Naming and security in a mobile, multihomed and multiple interfaces environement

Daniel Migault

THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Daniel Migault

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Nommage et Sécurité dans une environnement Mobile, Multihomé et à Interfaces Multiples

Soutenue le 26 Septembre 2012
devant le jury composé de :

Directeur de thèse :
- **Maryline Laurent Professeur Télécom Sud Paris**

Rapporteurs :
- **Abdelmadjid Bouabdallah Professeur Université Technologique de Compiègne**
- **Andrei Gurtov Professeur Université de Oulu**

Examinateurs :
- **Guy Pujolle Professeur Université Pierre et Marie Curie**
- **Stéphane Bortzmeyer Association Française pour le Nommage Internet en Coopération**
- **Nadia Boukhatem Professeur Paristech**

Thèse n°2012TELE0033

Ref : 2012TELE0033

Télécom Sud Paris / Université Pierre et Marie Curie

École Doctorale - EDITE

Naming and Security in a Mobile, Multihomed and Multiple Interfaces Environement

---

by Daniel Migault

Submitted in partial fulfillment of the requirements for the degree
of
Doctor of Philosophy at Télécom Sud Paris

September 26, 2012

Supervised by Maryline Laurent, Professor at Institut Mines-telecom
Certified by Andrei Gurtov, Professor at l'Université de Oulu
Abdelmadjid Bouabdallah, Professor at Université Technologique de Compiègne

Guy Pujolle, Professor at Pierre et Marie Curie
Stéphane Bortzmeyer, Association Française pour le Nommage Internet Coopératif
Nadia Boukhatem, Professor at Paristech

Contents

List of Figures

Summaries

English Summary

ISPs are concerned about providing and maintaining the level of security of its End User's communications. A communication is initiated by the End User with a name, and goes on by exchanging packets between two IP addresses. In this thesis, we focused our attention on two main points: (1) providing a secure Naming service, and (2) making IPsec communication resilient to IP address modification, addition or lost of an interface. We designed MOBIKE-X for that purpose and propose it as a standard at the IETF.

The first part is dedicated to the DNS to DNSSEC migration. Performance measurements show that DNSSEC migration for Resolving Platform requires up to 4.25 times more nodes. ISP's hardly cannot afford such increase and considering the differences between DNS and DNSSEC, we looked how to optimize the Resolving platform so to reduce the number of nodes. More specifically we are looking to optimize the operations that require most of the resources, that is to say: DNS Resolutions and DNS cache lookup.

The current architecture uses a load balancer that splits the DNS traffic among the nodes of the platform according to the IP addresses of the queries. This results in multiple parallel resolutions. In order to avoid the multiple resolutions, we started by splitting the traffic according to the FQDN rather than the IP addresses. This reduces the required resources by 30%, but these resources are non uniformly distributed among the nodes of the platform. In order to uniformly distribute these resources, we looked how the nodes can cooperate between each other, and use for that purpose a Distributed Hash Table architecture (DHT). Testing different DHT mechanisms shows that the proactive caching is the most efficient mechanism. Pro-active makes each node fill the other nodes with its most popular FQDNs. In fact the FQDNs' popularity of DNS traffic follows a power distribution, as a result, filling the cache with the most popular FQDNs results in caching a large part of the DNS traffic.

Another approach is to dissociate the pro-active caching mechanism from the DHT process. Pro-active caching is a light mechanism, easy to implement that can benefit from the Network Hardware Acceleration. More specifically, if the DNS query is requesting a FQDN cached in the Network Hardware Acceleration Card, then the response is provided and the DHT process is not even aware of it. Both architectures are likely to reduce at least the number of nodes by 4.

The second part is dedicated to IPsec configuration in a Mobile, Multihomed and Multiple Interface environment. MOBIKE [Ero06] is currently the protocol that deals with IPsec Mobility and Multihoming. However, MOBIKE only considers the Tunnel mode, and addresses only Mobile

Nodes with a single interface. MOBIKE-X [Dan09b] extends MOBIKE features to Transport mode and Multiple Interfaces, which thus provides also Soft Handover Mobility. The Tunnel mode is usually associated to a security gateway architecture, and the Transport mode is usually associated to end-to-end security, more or less like TLS.

A key application for MOBIKE-X is the capacity to offload traffic from Radio Access Network to WLAN. In fact WLAN does not have the same security properties as RAN, which requires the communication to be secured. One advantage provided by IPsec is that it makes possible to secure a communication without requiring the application to be modified. With MOBIKE-X, two architectures are thus possible. The first one consists in tunneling the traffic to a security gateway, using the IPsec Tunnel mode. The second one consists in providing end-to-end security with the Transport mode. On an architecture point of view, using end-to-end security avoid tunnel overhead, security gateway indirections, and decreases the network complexity. Performance measurements show that with Transport mode, Mobility operation interrupts the communication for $\approx 264\,ms$ which is between 9.3% and 15.6% faster than with the Tunnel mode.

French Summary

Une des problématiques majeures de sécurité pour les opérateur est de permettre à ses utilisateurs de maintenir la sécurité d'une communication même au travers d'un réseau qui ne soit pas de confiance. Nous avons pris le parti dans cette thèse de nous intéresser à deux problématiques : la sécurité du service de résolution de noms DNS et le maintient de la sécurité IPsec des communications, suite à une changement d'adresse IP, de l'utilisation d'une interface supplémentaire, ou de la perte d'une interface. Pour l'utilisateur, une communication est établie à partir d'un identifiant ou nom de domaine. Le système DNS permet d'associer à cet identifiant ou nom de domaine des adresses IP, qui vont permettre l'échange de paquets entre les deux nœuds. L'opérateur doit alors permettre à l'utilisateur de s'assurer que les adresses IP associées au nom de domaine sont légitimes, grâce à DNSSEC. Ensuite, nous avons pris le parti dans cette thèse, d'utiliser IPsec pour sécuriser la communication. Des mécanismes doivent également être mis en place afin de permettre à l'utilisateur de maintenir la sécurité de cette communication lorsque l'utilisateur change d'adresses IP, utilise une interface supplémentaire, ou perd la connectivité sur une de ces interfaces. Ceci est réalisé grâce au protocole MOBIKE-X que nous avons proposé à IETF.

La mise en place d'un service de résolution DNS Sécurisé (DNSSEC) nécessite d'augmenter la capacité des plateformes de résolution DNS, en multipliant jusqu'à 4.25 fois les ressources nécessaires. Les opérations qui nécessitent le plus de ressources sont la résolution DNSSEC et le nombreux cache lookup. Les architectures actuelles considèrent un load balancer qui réparti le trafic sur un ensemble de nœuds, en considérant les adresses IP des requêtes. La répartition du trafic est uniforme, mais de nombreuses résolutions simultanées sont réalisées par la plateforme. Pour éviter les résolutions parallèles, on répartit le trafic selon les noms de domaines. Cela réduit les ressources de 30%, mais la répartition est très inégale. Afin de palier à cette inégalité, on a choisit, dans cette thèse, d'organiser les nœuds de la plateforme en Distributed Hash Table (DHT) afin qu'ils puissent coopérer entre eux. En testant différentes organisation, on montre qu'un cache pro-actif est le mécanisme le plus efficace. Le cache pro-actif tire parti de la distribution des requêtes DNS. La distribution du trafic suit une loi de puissance. Ainsi, les 2000 Fully Qualified Domain Names (FQDNs) les plus populaires représentent environ 70% du trafic. Par conséquent, cacher ces 2000 FQDNs au sein de tous les nœuds de la plateforme de Résolution évite des résolutions inutiles.

Une autre alternative consiste à implémenter le processus de cache pro-actif en amont du processus DHT. Ainsi les requêtes concernant les FQDNs populaires cachés ne seront pas traitées par le processus DHT. L'avantage est qu'un tel processus peut tourner sur des cartes accélératrice, et ainsi réduire les ressources à fournir par les serveurs DHT. On montre qu'en considérant

les 2000 FQDNs les plus populaires, on divise par au moins 4 la taille de la plateforme de résolution.

La seconde partie est dédiée à la sécurité IPsec dans un contexte de Mobilité, de Multihoming et d'Interfaces Multiples. MOBIKE-X [Dan09b] est le protocole qui permet à la couche IPsec de gérer les opérations de Mobilité, de Multihoming, et d'interfaces Multiples. Si MOBIKE [Ero06] gère la Mobilité avec un Hard Handover pour le mode Tunnel et pour un terminal n'ayant qu'une unique interface, MOBIKE-X étend ces fonctionnalités au mode Transport, permet la gestion d'interfaces multiples ainsi que la Mobilité avec un Soft Handover. L'utilisation du mode Transport revient à une architecture où la communication est sécurisée de bout en bout, de la même manière qu'avec TLS.

MOBIKE-X permet aux ISP d'offloader les communications du Réseau Radio d'Accès vers des réseaux WLAN. L'intérêt d'IPsec est qu'il permet de sécuriser sans modifier l'application. IPsec propose deux modes: le mode Transport et le mode Tunnel. L'utilisation du mode Tunnel correspond à une architecture où le Nœud Mobile tunnelle l'ensemble du trafic vers un point d'entrée d'un réseau de confiance —en l'occurrence, celui de l'opérateur. Si les délais de mise à jour, dans le cas du mode Transport, sont 2.65 fois plus importants que dans le cas du mode Tunnel, en revanche, l'utilisation du mode Transport simplifie considérablement les opérations réseau, et permet au système d'être beaucoup plus réactif. Plus exactement, le temps d'interruption d'une communication d'environ $264\,ms$ est entre 9.3% et 15.6% plus rapide avec le mode Transport qu'avec le mode Tunnel.

Acknowledgments

This thesis would not have been possible without support that I received from many people and organizations.

I express my deepest gratitude to Professor Maryline Laurent that closely followed the ongoing work. Her knowledge and guidance made possible this thesis.
I would like to thank *Orange Labs* for providing time for academic research, for financing collaboration with *Télécom Sud Paris*, and for providing me the opportunity to be involved in IETF standardization activity. The IETF played a key role on providing guide lines and feed backs for this work both in the Naming and IPsec area. Among others, I would like to especially thank Tero Kivinen, Gabriel Montenegro, Stéphane Bortzmeyer, Francis Dupont, Miika Komu and Tobias Heer.

I had the opportunity to work with multiple colleagues and students that belonged to the Networking Security Team within Orange Labs. Among them are Jean Michel Combes, Stéphane Sénécal, Stanislas Francfort, Gilles Macario-Rat, Philippe Fouquart, Ryad Benadjila, Benoit Michau, Ludovic Eschard. I do not forget the support from PhD students in our lab such as Bodgan Marinoiu, Xavier Ferrer, Cédric Girard, Donglei Wang, Daniel Palomares, Emmanuel Herbert, Wei You, Ghada Arfaoui and Wolfgang Velasquez.

I would also like to thank my wife Christina, my family and my friends for their support.

Introduction

This thesis is focused on enhancing communication security. A communication is composed of packets exchanged between an Initiator and a Responder. The Initiator can be, for example, an End User or a device. It rarely uses network parameters like IP addresses to initiate the communication. Instead, it prefers using Names that provide a handsome identifiers for the Responder, both independent from the IP numbering and with a meaning that is easier to remember than IP addresses. On the other hand, packets remain exchanged between two IP addresses. Naming includes an architecture and a set of protocols, that make possible to bind a Name to various IP addresses. It works similarly to a phone directory, that makes you able to call *Daniel* rather than *+33670726958*. This is why Naming plays a crucial role in today's communications and can almost be included in the communication establishment phase. As such, in this thesis, we investigate how ISPs can provide a Secure Naming Service to both its End Users and for its internal devices and system configurations.

Once a communication has been established, we consider in this thesis a communication protected at the IP layer, that is to say, using the suite of IPsec protocols. The main reason we chose securing the IP datagrams from the Initiator IP address to the Responder IP address is that, ISP are willing to secure a communication over an untrusted network for example, without modifying the applications. IPsec completely achieves this goal. However, securing a communication from one IP address to another IP address, means that the IP address is expected to remain the same during a communication. This sounds like a reasonable assumption for static communication, but it is not true any more for Mobile communications. Moreover, communications that are more likely to require security are those of Mobiles Nodes being connected to multiple Networks. In this thesis, we assume that such Mobile Nodes, are likely to change the IP address of a given communication. Furthermore, with multiple interfaces, once a new interface is connected to a new Network, the Mobile Node is likely to benefit from this new IP address either to inform the Correspondent Node that the new IP address can be used in case the used IP address is not reachable —Multihoming —, or to simultaneously use both IP addresses —Multiple Interfaces. As such, in this thesis, we investigate how a Mobile Node can keep an IPsec protected communication over the previously mentioned changes of IP addresses. More specifically, we looked how the IPsec configuration can be updated so to provide an IPsec layer conform to IP address management.

As a result, we considered in this thesis the main two axes:

- How to deploy a Secure Naming Service
- How to provide a Seamless Network Security

How to Deploy a Secure Naming Service

Context Description

The Naming System provides an abstraction for IP addresses. Designed in the 80's, the Domain Name System [Moc87a, Moc87b] associated to a Fully Qualified Domain Name (FQDN) a set of parameters such as the IP address. This can be useful for End Users that are more likely to remember Names than IP addresses. Pressing *Daniel* dials automatically to proper phone number. Not only to the End Users, the use of Names rather than IP address is also very useful to manage networks. The reason is that FQDN makes reachability resilient to change of IP addresses. If configurations of devices like Boxes, phones provided by ISPs were based scripts or pieces of software using IP addresses, then any modification on the IP address plan would make these devices unable to configure themselves, and the only way would be to update or reconfigure all these pieces of software with the proper IP address. On the other hand, using FQDN only requires to modify the DNS, that centralizes the bindings. Another example is companies' Intranet, that are managed with DHCP. DHCP manages IP addresses resources, which results in providing different IP addresses for example to different servers. This makes access to these servers impossible with the IP address. However, DHCP updates the DNS, which provides access to the Server through a properly managed IP plan. At last DNS is also currently used for balancing load between servers, also called L7 load balancers. When an End User for example is requesting a connection to *www.myvideo.com*, *myvideo.com* may have multiple servers to handle all connections. In our case, the End User does not care which server it will be connected to, and the DNS can respond with the IP address of the server that matches some criteria such as lowest loaded server or closest server to the End Users. This is especially used for the Content Distribution Networks (CDN). All these examples show how we, End User and Network Administrators can take advantage of the separation of FQDN and IP addresses.

When designed in the 80's very few effort —or no effort at all —were provided to secure this system. Furthermore, at that time, people did not foresee how popular the DNS would become. So in the late 90's the IETF started designing a security extension for DNS: DNSSEC [EK97], published in 97. From this first version of DNSSEC, derived security extensions were used to secure transactions between servers (TSIG [VGEW00]), or administrative operations to secure question and response exchanges (SIG(0) [Eas00]). However, it took years to actively promote security mechanisms that secure the exchanges of any End Users. The second DNSSEC version [Eas99] in 99 and third version [AAL$^+$05a, AAL$^+$05c, AAL$^+$05b] in 2005 were especially focused on that point. From 2005 to 2008 the DNSSEC protocol was in a stable version. DNSSEC is an extension of DNS, but a major extension, and migration from DNS to DNSSEC has a considerable cost. DNSSEC introduces three security concepts for DNS:

- **Chain of Trust**: Starting from the Root zone, every zone is able to indicate securely which is the legitimate sub zone. The identifier used for the subzone is a public key, and the legitimate subzone is the one owning the corresponding private key. By signing with its private key, the subzone asserts it owns the private key. The key used for identification is called Key Signing Key (KSK).
- **Authentication and Integrity check of the DNS data**: When the End User receives a DNS response, DNSSEC adds a signature, to indicate who is providing the response, and to make the End User able to prove the response integrity. The signature is generated by the server using the Zone Signing Key (ZSK).
- **Proof-of-non-existence**: When a DNS query has no response, that is to say, the queried FQDN is not hosted on the zone. DNSSEC makes possible to prove the non-existence of the queried FQDN. The proof works in a similar way as one would prove a word does not exist in the dictionary. By ordering the zone file, the server responds something like "*the queried*

FQDN does not exist, otherwise it would be between those two FQDNs", with the associated two FQDNs. In some cases, it can be the hash of the FQDNs, instead of the FQDNs if the server does not want to send information about the hosted zone.

The DNS architecture is composed of Authoritative Servers, that host the zone files, and Resolving Servers that are in charge of resolving queries over the DNS architecture and send the response back to the client. For Authoritative Servers, DNSSEC makes the Servers sign the zone, which increases the size of the zone with additional data, and makes the responses larger than with DNS. For non existing FQDNs, DNSSEC involves hashing operation and comparison which is more costly than with DNSSEC. Furthermore, by signing the zone, DNSSEC cannot afford to publish a zone file with a mis-configuration. As a result, DNSSEC requires the existing DNS infrastructure to be upgraded for DNSSEC with more validation, and automatic key rollover management procedures. For Resolving Servers, DNSSEC introduces signature checks that make resolution much more costly than with DNS.

In this thesis, we measured the impact of DNSSEC on Authoritative Servers and Resolving Servers, and we measured that for Resolving Servers, the CPU resources must be increased by 500%. Given this increase, we designed an optimized Resolving Platform. The main principle of the architecture we propose in this thesis is to limit the number of resolutions. On multi node platform, we assign each FQDN a *Responsible node*, that is the only node likely to perform resolutions for that given FQDN. We show that we can reasonably reduce the involved resources.

Organization of the Material

As mentioned previously, this thesis measured how DNSSEC impacts the resources of a server. Then, we focus our effort on DNSSEC Resolving platforms, and propose different architectures that reduce the necessary resources. This section presents the goals and content of each chapter.

Chapter 1, presents the DNSSEC protocol, as well as the different actors that are involved in the DNSSEC migration. Then it presents the cost measurements of DNSSEC for Authoritative and Resolving Servers with various implementations.

This chapter introduces DNSSEC, provides input about its current development and deployment, and urges people that have not yet planned in their road map to migrate to DNSSEC to do so. The chapter also provides with measurements, an evaluation of the costs the DNS to DNSSEC migration represents. At the end of the chapter, we sum up the results with the example of an ISP willing to migrate its Authoritative and Resolving platforms. For Resolving Platform, a 500% of the resources make DNSSEC migration almost impossible with the current architecture. As such, optimizing the architecture of the platform is a challenging issue.

Chapter 2 proposes two architectures that optimize the resources involved on a DNSSEC Resolving Platform: one using a Load Balancer that load balances the traffic between the nodes of the platform according to the FQDN, and one that takes advantage of DHT nodes.

These architectures are designed to reduce the number of simultaneous DNSSEC resolutions. More specifically, current architectures designated as IP_{XOR} involves a Load Balancer that splits the DNS traffic between different nodes by XORing the IP addresses of the incoming DNS query. As a result, DNS queries for popular FQDNs are sent to all nodes of the platform, making each node of the platform perform a DNSSEC resolution. A first architecture considers specific Load Balancers that balance the traffic to the various nodes according to the hashing value over the FQDN rather than the IP addresses. By doing so, resolutions of FQDNs are provided by a single

node, which limit the number of resolutions. With such architectures, for very popular FQDNs, compared to IP_{XOR} where all nodes perform resolutions, the number of resolutions is limited by n where n is the number of nodes of the platform. Such architecture also reduces the cache length of the server, since a node only deals with $\frac{1}{n}$ of the global traffic. Simulations performed by replaying the DNS traffic show that Load Balancer balancing traffic according to the FQDN significantly reduces the resources of the platform —about 30%—, but presents a non uniform distribution of the resources among the nodes.

Replacing the Load Balancer can hardly be done by operational teams. Load Balancers are interconnecting the Platform to the CORE network, and under heavy traffic, failing Load Balancers may make the whole platform unreachable. For that purpose, we also investigate how the nodes of the platform can cooperate between each other and provide an architecture, that results in assigning every FQDN a specific Node. Distributed Hash Table (DHT) were designed for that purpose. The second architecture is derived from the DHT architecture we adapt to our purpose. In addition, DHT provides two other advantages over the Load Balancer alternative: DHT has been designed with auto-configuration mechanisms that ease Operation and Management (OAM) tasks, then, DHT comes with a bench of optimizations, caching or traffic redirection. This chapter tests multiple optimizations provided for DHT. One of the optimization is pro-active caching that has been designed for Zipf traffic distribution. In our case, the popularity distribution of the FQDNs looks like a Zipf distribution. This means that very few FQDNs, concentrate a large part of the traffic. As a result, each node considers the γ most popular FQDNs it is responsible for, and fills the nodes' cache of the other nodes of the platform. Caching these FQDNs in all nodes of the platform can be easily done, because, the number of FQDNs is quite small and would make every node being able to respond to a large part of the traffic. For the remaining traffic, the traditional DHT architecture is considered, that is to say, the node receiving the query checks forwards the query to the Responsible Node of the queried FQDN. The Responsible node is in charge of providing the response, which is then forwarded to the querying client.

Chapter 3 proposes a similar architecture as the DHT pro active caching architecture. The main difference is that we are taking advantage of Network Hardware Acceleration Cards that can handle some tasks of the Resolution on behalf of the Server. In our case, we started from the DHT pro-active caching architecture, where each node fill the other nodes' cache with its γ most popular FQDNs it is responsible for. Instead of making the DHT perform the pro-active caching process, we make it independent from the DHT and work on front end in the Network Hardware Acceleration Card. This means that a process runs in the card, intercepts all incoming DNS packets, and if the packet is querying one of the most popular FQDN, then the Card sends back the response. In that case, the DHT process is not aware of the query. If the queried FQDN is not one of the most popular FQDNs, then the Card forwards the query to the DHT process, that handles the resolution. If the node is the Responsible Node for the FQDN, then it sends the response, otherwise, it sends a query to the Responsible Node of that FQDN.

How to Provide a Seamless Network Security

Context Description

Before 2007, mobile phones were only carrying voice, and the mobile data only represented a small share of the global mobile traffic because, mobile data were expensive, applications were not so user friendly, and we believed there is no demand for these applications. In 2007, the iPhone provided a platform for applications connected to the Internet. The touch screen and designed applications were very user friendly, the screen was also attractive. This generates demands for applications, demand for mobile data, and today, one can hardly find a Mobile Phone and does not

have a WiFi Interface, that has not been designed to host applications. The demand is so high, that Cisco [Cis11] foresees the global demand for mobile data are expected to be 50 times greater than it is today. The Radio Access Network (RAN) will not be able to support this traffic at a reasonable price, which means that in the future:

- End Users with download limit are encouraged to use WLAN network when possible rather than RAN.
- ISPs must offload the traffic from RAN to WLAN when possible.

As a result, ISPs [NL11] and End Users are encouraged to use, when possible WLAN over RAN. A first step consists in configuring the Smartphone, so that when you are at home it uses the private WLAN instead of the RAN. In order to encourage End Users to offload by themselves their communication on WLAN, ISPs must provide multiple WLAN access Points. In Europe, DSL boxes deployed over various DSL End Users provide a good opportunity for a large WLAN coverage in major cities. In fact most European ISPs are developing WiFi Communities, which consist of sharing your WLAN Access Point with other End Users. Providing the infrastructure to encourage End Users to offload by themselves their traffic is a second step. However, in this thesis, we consider that ISPs are offloading the traffic of their End Users, to be able to manage RAN resources. This scenario is much restrictive because the ISP must provide the same security to the communication using the WLAN as on the RAN. Furthermore, the ISP must also provide the End User the same Quality of Service to its End User as with the RAN.

Security: RAN are trusted Networks whereas WLAN are untrusted. RAN are networks owned and managed by the ISP, which means that once the Mobile Node is securely attached to the antenna, the network behind is considered as a trusted network, and communication does not have to be protected. For this reason, communications on RAN are using Radio Layer security, at Layer 2, and any layers above are considered to be in a trusted network. While End Users are connected to a DSL Access Point, even though, the DSL User and the Mobile Node End User belong to the same ISP, there is no trust relationship between these two users, and the Mobile Node End User cannot assume the DSL End User is not listening to the IP communication. Similarly, when the End User is attached to a completely untrusted WLAN, like the one provided in a café, a bar, an hotel or an airport, the Mobile Node has no way to trust this network. As such, the ISP must secure the communication when it offloads the Mobile Node communications to WLAN.

Security can be handled by different layers, Radio Layer or Layer 2 is not sufficient, but one can consider the IP Layer, the Transport Layer or the Application Layer. Both Application Layer and Transport Layer Security require modification on the application source code. Application owed by the ISP can be adapted for these upper layers security, but third party applications cannot be secured this way. As a result, the only way to handle offload security remains using IPsec. This is the reason we considered IPsec security in this thesis.

QoS: RAN are reliable Networks whereas WLAN are unreliable. RAN are managed by the ISP and RAN is monitored and designed to be resilient to antenna fail over. This is not the case for WLAN based on DSL boxes for example. The boxes are not monitored —or monitored in a very light way —, and nothing prevents an End User for example to reboot or switch its box. As a result, Mobile Node must have mechanisms that prevent the communication to be interrupted when such events happen. Multihoming has been designed for that purpose. With Multihoming, the Mobile Node informs its Correspondent Node the IP addresses to which the Mobile Node may be reachable. Most Mobile Nodes have at least a WiFi and a Radio interface, and one can suppose Mobile Node in the future will have more WLAN Interfaces. With Multihoming, the Mobile provides the IP address of all its Interfaces and when it is no longer reachable on the running Interface, the Correspondent Node uses an Alternate Interface. Another mechanism may be the use of Multiple Interfaces. This means a Mobile Node may have a communication not only with

one IP address, but with Multiple IP addresses. Multiple Interfaces can ease the Multihoming decision, without completely switching from one Interface to the other, or can be used to split the flows for example.

QoS: RAN are managed Network whereas WLAN are unmanaged. At last, Mobile Nodes are not expected to remain attached to the same WLAN Access Point. As such, the Mobile Node must be able to change its attachment point, and so its IP address. This is called Mobility. As with Multihoming, and Multiple Interfaces, Mobile Nodes have connection managers that are expected to take the proper decision on which interface to use or when the handover should occur. The huge difference between RAN and WLAN is that such decision used to be taken by the Network, and are now taken by the Mobile Node.

QoS and Security. From the above section, we conclude that communications must be provided with mechanisms to handle Mobility, Multihoming and Multiple Interfaces. Such mechanisms are provided by protocols like Multi Path TCP [FRH+11] (MPTCP) or Stream Control Transmission Protocol [OY02] (SCTP). However, if communications are IPsec protected, modifications performed by Mobility, Multihoming, or Multiple Interfaces must not be only handled by the transport layer but also by the IPsec layer. IPsec can be seen as a firewall, whose rules have been defined according to the ongoing IP addresses used for the communications. When these IP addresses change, then the rules must be also updated. Any change or modification sounds more like creating a new rule rather than updating the already existing rules. The goal of this thesis is to design a protocol that were designed to update these rules. MOBIKE-X [Dan09b]. Currently IPsec only has MOBIKE [Ero06] that is providing Mobility and Multihoming mechanisms. However, MOBIKE is restricted to the IPsec Tunnel mode, and Mobile Node with a single interface. The purpose of MOBIKE-X is to extend these functionalities to the Transport mode and for Mobile Nodes with Multiple Interfaces.

IPsec is designed for two modes: Tunnel and Transport modes. Security architectures with the Tunnel mode most of the time involves a Security Gateway that is like a secure entry point of a trusted network. Security Gateway architecture come with a few issues. For example, they are more likely to be overloaded since they are dedicated to the whole offload traffic, the security gateway may add routing indirections, and the tunnel header may add some latencies as well as processing. For these reasons, in this thesis, we also consider End-to-End Security architecture. With End-to-End Security architectures, the Mobile node uses the IPsec Transport mode and is directly connected to the Service. Of course, for most of the services, the Security Gateway architecture may be enough, but services with Real Time Applications constraints like Voice Services, or game Services may benefit from this additional QoS.

Organization of the Material

Chapter 4 presents the protocol MOBIKE-X we design to enable the IPsec layer to handle Mobility, Multihoming and Multiple Interfaces. This chapter provides an in-depth description of the impact of Mobility, Multihoming and Multiple Interfaces on the IPsec configuration. Then, it describes the current protocols like IKEv2 [KHNE10] and MOBIKE [Ero06]. The reason we designed MOBIKE-X is that IKEv2 generates an exchange that is more or less like a new negotiation —CREATE_CHILD_SA exchange. This exchange is complex and not mandatory on the implementations. MOBIKE, on the other hand has been designed only for the Tunnel mode and Mobile Nodes with a single Interface.

Chapter 5 measures the performances for Mobility, Multihoming and Multihoming of our

MOBIKE-X. It compares the performances, when possible with MOBIKE and SCTP mobility over static IPsec configuration. When IPsec is used with the Transport mode, mobility of the communication must be handled by an additional protocol at the transport layer. In our case, we used SCTP. At first we evaluate the impact of IPsec over an SCTP mobility, that is to say we measure SCTP mobility over non IPsec protected links, and over IPsec protected links. This makes possible to measure the impact of IPsec and to compare how much the configuring IPsec stalls the communication. The IPsec impact is estimated for both Tunnel and Transport modes. From lab measurements, this chapter measures the differences over the different IPsec modes in a mobile environment, which helps deciding whether a communication should be secured with an End-to-End architecture or with a Security Gateway architecture. Then this chapter also points the enhancement provided by MOBIKE-X, even for the tunnel mode. In fact, by enabling the use of Multiple Interface, MOBIKE-X makes Soft Hand-over possible, as opposed to MOBIKE that only consider a single Interface.

Chapter 6 is focused on how ISP can offload their End Users communications from RAN to WLAN. This chapter starts by presenting the economical context of offloading. Then it details and compares the two possible ways to securely offload an application flow. One way is to consider End-to-End security, that is to say the communication is secured from the Mobile Node to the Service. The other way is to consider a Secure entry point to the trusted network of the ISP. The Mobile Node tunnels its communication via a secure tunnel to that Security Entry point. In both cases, MOBIKE-X provides advantages over existing protocols. It makes possible the end-to-end architecture, and provides soft handover to the security gateway architecture. Then, this chapter focuses on deployment issues, and explains how ISP can deploy offload. Two aspects are considered. The first one is how an ISP can reasonably deal with offload, without having to deploy a heavy infrastructure. For that purpose, we suggest to associate different offload strategies depending on the flows, and the nature of the service. Once a strategy has been chosen for each flow, the chapter provides input of combination between SCTP and MOBIKE-X so to ease deployment. Different possibilities are provided and depend especially on the nature of the service. The key point is how to move a non IPsec protected flow to an IPsec protected flow. In fact, this kind of mobility differs from regular MOBIKE mobility. The different alternatives exposed are using SCTP which requires applications to be ported on SCTP, using application resilience mechanisms, or using a specific configuration for MOBIKE-X. All variants described in this chapter are clearly analyzed and compared, to provide inputs for taking the proper decision. This chapter does not present a best solution for all cases. It provides input so it is possible to choose the solution that best fits the needs.

Part I

Providing Secure Naming Resolution

Introduction of Part 1

Description of the Work

The DNS(SEC) part is composed of three chapters: Chapter 1 provides experimental measurements and evaluates the cost of migrating from DNS to DNSSEC.Chapters 2 and 3 evaluate different architectures for resolution platforms.

DNSSEC is the DNS Security Extension designed by the IETF [AAL+05a, AAL+05c, AAL+05b]. Chapter 1 presents the main actors involved in the DNSSEC migration as well as the performance issues DNSSEC introduces. DNS and DNSSEC involve multiple actors such as Network Information Center that provides Domain Names, ISPs that performs DNS Resolutions on behalf of their end users, End Users that request the Naming Resolution Service, manufacturers that deploy platforms, software manufacturers that provide pieces of codes for End User, Resolution and Authoritative Platform. Deploying DNSSEC requires all these actors to jointly deploy DNSSEC. Some agree for deployment while others diagree, thus leading to a confused situation for the End User where DNS and DNSSEC are still used together. Chapter 1, in Section 1.2.2, provides a description of these actors and focuses on their motivations for deploying DNSSEC and their DNSSEC deployment status.

The second part of Chapter 1 is dedicated to performance measurements. DNSSEC introduces signatures checks that require more resources than it used to be with DNS. This section provides experimental measurements and compares DNSSEC to DNS. Moreover, measurements are provided for different implementations and for different configurations, that is to say for the Authoritative and Resolving configuration. The measurement platform and procedure are described in section 1.4, and measurements are provided in section 1.5.

DNSSEC deployment for large ISPs is an important issue since their DNS resolving platform must be enlarged by up to 5 times. There are mainly two expensive operations in DNS(SEC) resolutions: signature checks during the DNSSEC resolution and cache lookup when the caches are hosting a lot of FQDNs. In fact, caches are filled with multiple FQDNs that are never requested. In order to reduce the necessary resources, this chapter starts with the idea of imitating redundancy of FQDNS among the various cache of the nodes of the platform. To avoid simultaneous resolutions, this chapter assigns each FQDN a Responsible Node. For each FQDN, any time the FQDN is requested, the DNS query is forwarded to the Responsible Node. By doing so, at least for the Most Popular FQDNs, the number of resolutions is reduced by n, if n is the number of nodes of the platform. Then, depending on the popularity of the FQDN, it can be cached or not among the other nodes. Caching would avoid multiple forwarding, but in return, will make require

cache lookup.

In chapter 2, we consider two different approaches to assign for each FQDN a Responsible Node. One way is to use a hash function and a Load Balancer. The load balancer receives the DNS traffic of the End User, performs a hash function on the FQDN to determine which node the query must be forwarded to. In today's resolving platform, the traffic is split according to the IP addresses, which can be seen as assigning specific End User to each node. By splitting according to FQDNs, the Load Balancer assigns a unique Responsible node to each FQDN. Section 2.2 compares the different characteristics of IP addresses and FQDNs on a real live traffic capture. Section 2.4 runs simulations with real traffic, with different Load Balancing technics and compares how the resources are spread among the nodes of the platform. We show that such architecture requires up to 30% fewer nodes.

However, splitting according to the FQDNs still raises few issues: FQDNs have very different popularities, which makes the resources non uniformly spread among the nodes. We show in [MHS+a, XMSF11, FMS11, FMS12] how to overcome this issue, that is how to establish routing policies for FQDNs so that resources on the platform are uniformly distributed. This modifies a bit how the Load Balancer works, but provides efficient results. Another issue is that such Load Balancers are not widely deployed, and in addition Operational teams are not willing to modify the Load Balancers, as they are crucial elements connecting the Platform to the Core Network. For all those reasons, we look for an architecture that does not require modifications of the Load Balancers. In addition, Operations, Administration and Maintenance (OAM) is getting quite complex with the increasing number of nodes of the platform, and thus we are looking for an architecture that would provide auto-configuration facilities.

All those requirements make us investigate how could the Distributed Hash Table architecture (DHT) address these issues. In section 2.5, we describe different DHT architectures. With a live traffic capture and network measurements described in section 2.6, we evaluate their performances in section 2.7. We show that some architectures can be at least 4 times more efficient than the current resolving platform.

Chapter 3 proposes a second architecture solution: $PREFETCH_X$ is an architecture to overcome the ever increasing DNS traffic and the DNSSEC migration for large platforms, mainly hosted by ISPs. $PREFETCH_X$ consists in prefetching and caching the X most popular Fully Qualified Domain Names (FQDNs), and handling the remaining FQDNs by a Distributed Hash Table (DHT) architecture. Considering the popularity of each FQDN, we designate by $HEAD_X$ the most popular FQDNs and $TAIL_X$ the remaining FQDNS.

$PREFETCH_X$ balances the two different models proposed in chapter 2, that is the Load Balancer and a Distributed Hash Table (DHT) architectures that assigned to each FQDN a Responsible node. Then it proceeds differently for $HEAD_X$ and $TAIL_X$.

HEAD_X, the X most popular FQDNs are prefetched. Prefetching a FQDN means that the Responsible Node is in charge of populating all caches of the other nodes of the platform with the associated response of that FQDN. This reduces the impact of very popular FQDNs for the Responsible Node. In fact the DNS traffic is split by the Load Balancer between the nodes of the platform. The load balancer XORs the IP addresses which results in a uniformly distributed traffic among the nodes. Because, the Most Popular FQDNs are cached on all nodes of the platform and because the DNS traffic is uniformly distributed according to the IP addresses, their associated traffic is uniformly distributed between the nodes of the platform. Then the number of Most

Popular FQDNs is relatively small (a few thousands). This means a cache lookup to the cache associated to $HEAD_X$ is much smaller than the cache associated to $TAIL_X$, and so it makes the cache lookup much faster. Furthermore a lookup in cache of a few thousand lines can be implemented as an independent process from the DHT, and is a much lighter process. For these reasons, the process can be applied for any incoming DNS query, and forwarded to the DHT process only if the queried FQDN is not in $HEAD_X$. Because the process is light and independent, we can take advantage of Network Hardware Acceleration Cards and implement it as a front end process. Then, this process uses the embedded CPU of the card, keeping of the node's CPU resources for the DHT processing.

$\boldsymbol{T}AIL_X$ is composed of the FQDNs that are not in $HEAD_X$, and are handled by the DHT architecture. More specifically, queries concerning $TAIL_X$ are forwarded to the DHT process only if the Network Hardware Acceleration Card has not been able to respond. $TAIL_X$ takes advantages of the DHT architecture and avoids multiple resolutions, and responses are only provided from the Responsible Node. $TAIL_X$ is composed of less popular FQDNs, so there is a smaller probability, it is cached by the platform. The probability the FQDN is cached in a node is even smaller if the traffic is split among the nodes of the platform by XORing the IP addresses of the query. In this case, DHT avoids a resolution is performed over the Internet.

Section 3.3 details the goals and motivations for the $PREFETCH_X$ architecture as well as the design of $PREFETCH_X$. Live capture analysis estimates that $X = 2000$ FQDNs should be prefetched. Section 3.4 details how to derive X from a traffic analysis. X defines the Most Popular FQDNs. In addition, executing different theoretical models of DHT architectures concludes that *Pastry-SF*, derived from *Pastry*, improves significantly the performance over IP_{XOR}, —the currently deployed architecture. Section 3.5 computes the traffic $TAIL_X$ with different theoretical models defined in section 2.5. Results are provided in section 3.5 and shows which DHT architecture best fits our goals with $TAIL_X$. The choice of an architecture is driven from models evaluations, and we need to validate our models are valid. In section 3.6, an Experimental Platform based on FreePastry confirms our models are valid, and confirm our results are valid.

Notations & Abbreviations

In this part, we are using the following notations and abbreviations:

- AXFR: DNS full zone transfer opcode
- CHR: Cache Hit Rate
- OAM: Operations, Administration and Maintenance
- CPU^H: Occupancy time for a Cache Hit.
- CPU^R: Occupancy time for a Resolution over the Internet. It occurs when the response is not already cached.
- DHT: Distributed Hash Table
- DNS: Domain Name System
- DNSSEC: DNS SECurity extension
- FQDN: Fully Qualified Domain Name
- MPFQDN: Most Popular FQDNs
- NIC: Network Information Center
- RT: Response Time

Chapter 1

A Performance view on DNSSEC migration

This chapter provided inputs for CNSM'10 [MGL10] and results were presented at IEPG'79 [Mig10].

The mathematical expressions for measurements are provided in Annex A.1, and configuration files used are provided in Annex A.2.

1.1 Introduction

DNS [Moc87a, Moc87b] defined at the IETF, represents today's Naming System of the Internet and makes possible to start communications with names, and thus people to communicate through the Internet. Fully Qualified Domain Names (FQDN) are often more stable and easier to remember than IP addresses and people are more likely to deal with names than with IP addresses that define a network localization. On the other hand, DNS is not only used by end users, but also by the core network. Convergence between traditional telephone service (PSTN) and Voice over IP (VoIP) is expected to be done thanks to E.164 NUmber Mapping (ENUM) protocol which is based on the DNS [MD00, Fal00].

As a crucial element for making the Internet useable, the Internet Community is concerned about security issues on DNS. The Internet Engineering Task Force (IETF) started designing DNSSEC in January 97 [EK97], and a final version was issued in March 2005 [AAL+05a, AAL+05c, AAL+05b].

The DNS architecture is composed of *Clients* that send DNS queries to *Resolving Servers*. The *Resolving Servers* are responsible to find the responses of the *Clients*'s queries. Responses are hosted on *Authoritative Servers*. The reason *Clients* do not send their query directly to *Authoritative Servers* is that the DNS is a distributed and hierarchical database. This means that to get the Response, the *Resolving Servers* may have to send multiple queries to various *Authoritative Servers*, and has to interpret their responses. Another reason is that *Authoritative Servers* would not be able to handle all *Clients*'s DNS traffic.

DNSSEC is the security extension of DNS and provides resolvers with the mechanisms to authenticate the origin of the RRset, integrity protect RRsets, build a chain of trust and prove the

non-existence of the FQDN or a specific RRset. DNSSEC and DNS are compatible in the sense that a DNSSEC authoritative or resolving server can treat a DNS request. However DNSSEC comes with so many changes to the architecture, the servers and network security policies that it is better to consider it not as an extension of DNS but rather as a new protocol.

Complexity may be the major drawback of DNSSEC, and one of the main reasons for its slow adoption. Up to 2008, ISPs and regular firms were hardly considering DNSSEC adoption. In fact, in July 2008 Dan Kaminsky revealed a major flaw in the DNS specifications that makes it sensitive to cache poisoning attacks [Kam08b, Kam08a]. At that time DNSSEC was considered to be the long term solution to make DNS robust to cache poisoning attacks and it almost closed the debate about whether or not DNSSEC was worth being deployed.

This chapter intends to help those organizations to position themselves towards DNSSEC.

At first we show WHY organizations - and ISPs - should start their DNSSEC migration. More specifically, we detail the current position toward DNSSEC of major actors of the Internet community, and we show that DNSSEC is part of the Internet evolution. We also describe, for organizations, the benefit of migrating to DNSSEC. Then we show HOW organizations should handle DNSSEC migration. This includes considerations on how DNSSEC impacts the network as well as how platforms should be upgraded with regards to a performance point of view. This consideration on DNSSEC impacts is intended to help network administrators to plan and evaluate the cost within their own organization of the DNSSEC migration. Then we focus our concern on performance and consider how the DNS platform should be upgraded to DNSSEC. We present experimental measurements for various implementations to compare the cost of DNSSEC over DNS with various configurations. These are expected to help organizations define the DNSSEC architecture and the implementations that best fit their requirements as well as clarify how many servers should be added to the DNS platform, how much response time is increased, how many DNS update will we be able to be performed:.

The remainder of this chapter is organized as follows. Section 1.2 presents the current position towards DNSSEC of various actors in the Internet community, as well as considerations of DNSSEC deployment. Section 1.3 positions our experimental work. Next, sections 1.4 and 1.5 present our experiments. This includes a description of the testing environment, our methodology, DNSSEC impact on end user side with unitary naming resolution, DNSSEC impact on loaded servers considering maximum load and the response time for resolution and update operations. Based on those measurements section 1.6 illustrates how experimental measurements can help the upgrade of DNS platforms. Section 1.7 discusses these results and points that need to be looked at while considering DNSSEC migration.

This chapter presents the experimental measurements. Mathematical expression for these measurements are provided by Annex A. These expressions are more handy for modelizations.

1.2 DNSSEC Current Status

1.2.1 DNSSEC Brief Description

As mentioned before, DNSSEC [AAL+05a, AAL+05c, AAL+05b] provides mechanisms to authenticate the origin of the RRset, integrity protect RRsets, build a chain of trust and prove the non-existence of the FQDN.

Authentication and integrity protection are performed by the signatures (RRSIG). This means that each DNSSEC zone MUST be somehow assigned an identity – a Key Signing Key (KSK) – and a key that signs the DNS zone – Zone Signing Key (ZSK). The KSK has also cryptographic properties and is used both to identify a zone and to sign the ZSK so that ZSK are bound to the proper KSK. The chain of trust implies that once you trust one entry, you can securely trust the

subdomains. This trust delegation is performed through the Delegation Signer (DS) RRset, where a parent specifies the KSK of its child. The proof of non existence can be performed through two different mechanisms: NSEC [AAL+05a, AAL+05b, AAL+05c] and NSEC3 [LSAB08]. Both mechanisms are based on ordering all RRsets of a zone file as a dictionary. Each FQDN has a specific place in the zone file, and NSECx RRsets provide the link between the FQDNs. NSECx can prove an FQDN does not exist as it can indicate the FQDN is not at the right place. Furthermore, the response is signed by the authoritative server. By ordering and providing FQDN, NSEC enables zone walking [dns08], that is to say downloading the whole zone file even though the AXFR is disabled. NSEC3 addresses this problem by considering the hash of the FQDN instead of the FQDN itself. Since hash are one way function, providing the next hash does not provide any inputs on the FQDNs hosted in the zone.

1.2.2 DNSSEC Deployment

This section provides current DNSSEC deployment for the various actors of the Internet community. This includes Network Information Center (NIC), DNS software companies, OS implementers and ISPs.

1.2.2.1 Network Information Center (NIC)

NIC are part of the most influent actors in the DNS community, and were the early adopters of DNSSEC. [Ric09] provides Registries view on DNSSEC deployment, as well as DNSSEC deployment history. The .se and .cz were the first ccTLD to sign their zone. In the Internet SOCiety (ISOC) DNS panel in July 2009 James Galvin "*Our mission is to serve in the public interest, so securing our Top Level Domain (TLD) with DNSSEC became a top priority*" [Gal09]. Then, DNSSEC ranks first in US IT priorities for 2009 [Jac09], which results in .gov to be signed. In 2010 TLD that are known to have deployed DNSSEC are : .se (Sep 2005), .ru (Apr 2006), .pr (August 2006), .bg (Jan 2007), .br (Jun 2007), .org (March 2008), .cz (Sep 2008), .gov (Feb 2009), .na (Sep. 2009), .tm (Nov 2009), .li (Feb 2010), .ch (Feb 2010), .arpa (Feb 2010), .th (March 2010), .uk (Mar 2010), .enum (Mar 2010), .pm (Apr 2010), .edu (Apr 2010), .fr (Jul 2010), .re (Jul 2010), .nl (Aug 2010), .com (2011), .net (2011). Some of them also offer a DNSSEC delegation and provide facilities to sign their entries. DNSSEC deployment therefore varies from country to country (or TLD to TLD) but the trend is that most TLDs will implement DNSSEC [Wou10].

DNSSEC was first designed with the root signed in mind, thus providing a PKI-like infrastructure, with a single trust entry point. However it took time until the root zone was finally signed, which resulted in multiple isolated DNSSEC islands. As such, DNSSECed TLD and other zones published their own key in Trust Anchor Repository (TAR) and resolvers need to get the keys from TAR. This mechanism is known as DNS Lookaside Validation (DLV) [AW06, Wei07]. In June 3, 2009, [ICA09], ICANN announced that the root zone would be signed in 2009, and the Root zone has been signed on July 16 2009. Although it does not change much in terms of validation, politically it is a major step for DNSSEC and the naming system since it is able to provide a global chain of trust, with a single entry point.

The advantage of DLV mechanism is that it makes possible to deploy DNSSEC zones without waiting for the root zone to be signed. The drawbacks were that resolvers have to store multiple keys as Security Entry Point (SEP), and thus probably scared network administrators or network architects with this added complexity.

1.2.2.2 Software Implementers

DNS related software implementers were also heavily committed into the DNSSEC deployment. In 2010 DNSSEC is part of most DNS implementations - Internet Systems Consortium[1] (`BIND9`), NLnetLabs[2] (`NSD`[3] and `UNBOUND`[4]), Microsoft, Nominum (`ANS` and `CNS`), Secure64,... and powerdns[5] is actively implementing DNSSEC.

Administrative tools are also actively developed Opendnssec [6] is designed to manage security of Zones. Other pieces of software available are listed on [dnsb, dnsc].

1.2.2.3 OS Implementers

In November 2008, Microsoft announced how DNSSEC would be supported in windows 7 [Ses08]. Resolvers on Windows 7, clients do not perform the validation by themselves, but they let the trusted DNS resolving server deal with Trusted Anchor management and validation. For complete end-to-end security, communication between the resolver and the resolving server is expected to be secured using IPsec. Thus, to take the benefit of DNSSEC, server local policy must implement the DNSSEC signature check policy by default, and resolver must trust the resolving server. In other words, network administrators and ISPs have to deploy DNSSEC with signature check in their resolving servers. In February 2009, Microsoft implemented DNSSEC on Windows Server 2008 R2 [Mic09].

1.2.2.4 ISP

Among ISPs, Comcast [Com] is currently the only one that is publicly advocating DNSSEC adoption, and that, by the end of 2011, will sign its authoritative domains and proceed to DNSSEC validation[7] on its resolving servers [Gri10].

1.2.3 DNSSEC Impacts on the Network

1.2.3.1 DNSSEC Compliant Infrastructure

First of all DNSSEC is complex and operational teams need to become familiar with that protocol. Procedures are complex and need to be adapted to the operational environment with automatic procedures. "*If your data signatures don't validate, you're down! [...]* **You can't make a mistake!**" reports Kevin Oberman in [Obe09]. Then ISP naming infrastructure not only involves servers (resolving and authoritative) but also middle boxes located in end user's home. Deploying DNSSEC requires to validate DNSSEC compatibility across all the network equipments involved in the naming resolution as well compatibility with our services.

On servers' side, Comcast reports at NANOG45 [Gri09] that DNSSEC increases memory footprint between 5 and 9 times for the authoritative infrastructure and that the recursive infrastructure requires additional recursive clusters. For middle boxes, like residential Internet router and SOHO

[1] `http://www.isc.org`
[2] `http://www.nlnetlabs.nl`
[3] `http://www.nlnetlabs.nl/projects/nsd/`
[4] `http://www.unbound.net`
[5] `http://www.powerdns.com`
[6] `http://www.opendnssec.org`
[7] `http://blog.comcast.com/2010/02/dnssec.html`

firewall devices commonly used with broadband services, [BP08] shows that only 25% of the tested boxes in 2008 were fully DNSSEC compliant.

On the user point of view DNSSEC resolution on small devices may slow down web surfing and [LCGW09] shows that DNSSEC may not be compatible with the DNS redirect service provided by ISP. As a result, moving to DNSSEC without breaking connectivity for end users represents a great challenge.

1.2.3.2 Monitoring DNSSEC

DNSSEC introduces security to the traditional DNS service. However, DNSSEC also brings its own issues that can make resolutions impossible. One common reason is that DNSSEC packets are larger than regular DNS packets, and thus may be dropped by network devices. Resolvers advertise through the EDNS0 option [FM04] a larger packet size than the traditional DNS 512 bytes packet size. If the indicated packet size is larger than the one accepted by the network for an end-to-end connectivity, then smaller packet sizes are to be used. This operation is called the Path MTU walk (PMTU) [rep09]. [ORMZ08] monitors DNSSEC zones and traffic and shows that roughly 20% of the monitored zones suffer availability dispersion, and that PMTU walk is necessary for roughly 95% of the DNSSEC zones for 1.5% of the time. Finally, [ORMZ08] - maybe no longer up-to-date - shows that in 2008 97% of the DNSSEC zones were isolated -i.e. not attached to a TA-, and thus not verifiable. Then 9% of the authentication chain were broken which means that a DNSSEC resolution would consider the entire subzones as invalid. This usually happens when a zone is re-signed before updating the delegation. Then, due to the absence of revocation mechanisms, 19% of the zones had data that are still valid according to their signature expiration date, but that do no longer exist in the zone file.

1.2.4 ISP Position for Migrating to DNSSEC

This section considers network administrator's or ISP point of view, and describes their current position toward DNS. Then we mention how they are impacted by cache poisoning attacks, the problems encountered to have their infrastructure DNSSEC compliant and the new issues that come with DNSSEC.

1.2.4.1 ISP Consideration about DNS

ISPs aims at providing Internet connectivity and services to end users. DNS is only one component to provide this connectivity. Until now DNS architectures for authoritative and resolving servers were quite scalable, performed well, and were considered as an operational issue rather than a research issue. This at least explains why they were not that involved at the beginning of DNSSEC deployment and why DNSSEC seems new to them.

1.2.4.2 Cache Poisoning

With the Kaminsky Attack in July 2008, people became aware that their DNS architecture was sensitive to DNS cache poisoning. On the other hand ISP providing email facilities are facing phishing and pharming issues [Oll05] and DNS cache poisoning is one vector for such attacks. The AntiPhishing Working Group [APW] (APWG) shows that hijacking brand name is still an increasing issue, [Oll09] reports with concrete examples how valuables are FQDN for companies. As a result DNSSEC is attractive to protect companies brand, to protect Internet Services – ISP don't want for example their end user's email being redirected nor to provide corrupted DNS resolution

with corrupted cache. As Chris Griffiths from Comcast reports "*Current recursive infrastructure is not vulnerable but we cannot sit back and wait for the next big bug/exploit.*" [Gri09].

1.2.4.3 Position toward DNSSEC

ISP's position toward DNSSEC is balanced between the cost of DNSSEC migration and the impact of not upgrading their Naming System to DNSSEC. Costs for DNSSEC migration are high for organizations, since it impacts operational infrastructure, platform and network performances. However DNSSEC is being deployed by NIC, governmental institutions, and OS implementers, and end users ask for more security. As such DNSSEC is part of the Internet evolution. Delaying its migration may only make the cost higher in the future. In fact, today DNSSEC traffic is quite low, and is expected to increase with DNSEC deployment of major TLDs, end users OS, organizations... Increase of DNSSEC traffic will make the migration harder, and costs higher. On the other hand, not migrating to DNSSEC means that we keep the DNS organization into insecure islands on the Internet. This includes preventing end user from securing their naming resolution, accepting that end user private data may be redirected to an attacker web site, accepting that our domain name may be hijacked and our services made unavailable.

1.2.4.4 Elements Helping the migration

Migration to DNSSEC can be done in various way for resolving servers. With current DNS configuration, resolving servers only perform DNS resolution. A first DNSSEC configuration can make them perform DNSSEC with no validation when requested by the end user. Then this configuration can be extended to all incoming DNS queries. Finally servers can be set to proceed to DNSSEC validation. This chapter intends to provide input to evaluate the cost of each configuration.

Specifications [AAL+05b, AKB07, Con01, WG03] mention that DNSSEC clients and resolving servers exchange DNSSEC parameters through three bits : the DO bit (DNSSEC OK), the AD bit (Authentic Data) and the CD bit (Check Disable). [Con01] specifies that the DO bit indicates that the client is DNSSEC aware, which means it does understand DNSSEC response and DNSSEC RRsets. In other words, it does not specify who performs the signature check. [AKB07, AAL+05b, WG03] define that the AD bit indicates that the resolving server has checked the signatures according to the records mentioned in the Answer and Authoritative section. This is an indication, and should not be blindly trusted by the client. [AAL+05b] specifies that CD bit is set by the client to indicate that the signature check will be processed with the client local policy rather than the resolving server's policy.

1.3 Related Work

Most of the DNSSEC deployment guides are configuration oriented [Kol09, CR06, SL10, Cle08]. Some give key elements of the deployment, to help managers taking the decision to migrate to DNSSEC and to supervise the migration [Ait09, Sar10]. [Gra10] provides a checklist of key points to consider in the network before enabling DNSSEC on servers. However, all of them assume that the decision to migrate has been taken, and do not provide inputs so that administrators can plan DNSSEC migration or evaluate its costs. Performance impact has already been measured in various ways in previous studies. In 2005, O. Kolkman studied the resource requirements of DNSSEC for some of the root servers [Kol05]. The same year, B. Ager et al. conducted a study [ADF05, Age05] on two specific topics. First they observed how the size of the packets is impacted by switching to DNSSEC. Then they studied how memory and CPU usages are impacted, both on caching and authoritative servers, for a very specific situation. One year later, A. Guillard

performed a study [Gui06] focusing on authoritative servers only and testing impact on the zone and the bandwidth. [MM06] compares IPsec and DNSSEC. Tests were performed on the same platform, using different tools, and the DNSSEC version was not considering NSEC3 [LSAB08] but NSEC [AAL+05a, AAL+05c, AAL+05b].

Our work differs from previous work in that we study the impact of DNSSEC on authoritative servers, resolving servers and resolvers. Performance tests are realized with the DNSSEC NSEC3 option that was not available at the time of previous studies. Performance tests are performed with DNS and DNSSEC on various implementations. This makes possible to compare the cost of DNSSEC for a given comparison as well as to compare the performances of each implementation.

1.4 Testing Platform

In this chapter, we consider BIND 9.6.0-P1, UNBOUND 1.2.1 and NSD 3.2.1. Other DNSSEC implementations were available such as Microsoft DNS, power DNS, Simple DNS plus, Secure 64 and Nominum. We did not take them into consideration because DNSSEC-NSEC3 was partially implemented (power DNS), they required specific hardware (Secure64), they were not able to work on a Linux platform (Microsoft DNS), or we did not get the binaries.

NSD - authoritative server - and UNBOUND -recursive server- are both developed by the NLnet Labs whereas BIND9 is developed by the ISC. Next BIND version, v10, will also be split into different pieces of code for the authoritative and recursive server, which is expected to improve its performance. BIND and NSD have distinct designs. BIND loads its zone file whereas NSD compiles it so that any possible query is handled. As a consequence, updating a zone file requires recompiling the NSD zone file, and dynamic updates cannot be done with NSD. Nevertheless the data structures used by both implementations are more or less the same: red-black binary tree for authoritative data and hash table for cached data. In this chapter resolving servers are used without cache, except for section 1.5.5 and 1.6 where we look at the impact of caching.

1.4.1 Testing Environment

The hardware configuration used to do the test was based on very old servers and powerful end-clients. This choice has been made first of all to avoid speed limitation on the client's side. Secondly, using servers with limited performance, makes it easier to reach their limits without using multiple clients or being limited by network bandwidth. For our tests, we used Intel Pentium III (@ 1GHz 32 bits) CPU, 384MB of RAM for servers with Debian 5.0 (lenny), Linux kernel 2.6.24. To load the servers, we used an Intel Xeon E5420 (Quad-Core @ 2.5GHz 32bits) CPU, 3GB RAM with Ubuntu 8.10 (hardy) 32 bits version with Linux kernel 2.6.27. The tested BIND version was multithreaded, but with one CPU we used one thread.

The testing environment was designed to measure DNS and DNSSEC performances on resolving and authoritative servers. The two configurations are represented in figure 1.2. Time was measured using `Wireshark`. *Client Processing Time* is the time to initiate the query, forge the datagram, and send it to the outbound network interface, as well as the time to receive the response from the inbound interface back to the software. *ISP Network Latency* and *Internet Network Latency* are the time datagrams are on the wired network. *Cache processing Time* is the time a query is received on the inbound interface, processed, and a resolution is handled plus the time to forward the response from Internet interface to the client interface. *Authoritative Processing Time* is the time authoritative servers take to receive a query and send the response.

The data used for the tests were directly hosted on the authoritative server and we did not

consider any hierarchy in our Naming architecture, and the naming space was quite flat. When tests involved DNSSEC, all data in the authoritative server are signed with a single key, and this key is trusted by the resolving server. We choose the simplest scenario so that more complex models could be derived.
In this chapter, we consider different configurations for each authoritative and resolving servers.

- **Authoritative servers** can be configured with *DNS* or a *DNSSEC* server.
- **Resolving servers** can be configured with *DNS*, *DNSSEC* and *DNSSEC with validation*. The difference between *DNSSEC* and *DNSSEC with validation* is that with *DNSSEC* the Resolving server requests the additional data like signatures, chain of trust information but does not perform any signature checks. This can be done for example, if the client explicitly requests that the Resolving server does not perform the signature checks. With the *DNSSEC validation* signature check is performed by the resolving server. This configuration is obtained with `dnssec-enable yes` and `dnssec-validation yes` for BIND and `module-config "validator iterator"` for UNBOUND - Annex A.2 provides the various configuration files used for those tests.

1.4.2 Testing Tools

Tests were performed with different tools. `dnsperf` and `resperf` [dnsa] have been used to send requests or updates to DNS servers. Performance measurements have been made with `collectl` [col] and network latency measurements with `Wireshark` [Wir].

1.4.3 Testing Methodology

For our different tests, we used, when possible the median instead of the mean value. Figure 1.1 gives the distribution of measurements for a specific test, and shows that the median limits the weight of erroneous measurements. As such we consider it more representative of the data. However, for tests that involved `dnsperf`, we considered the mean value since this is the output of `dnsperf`.

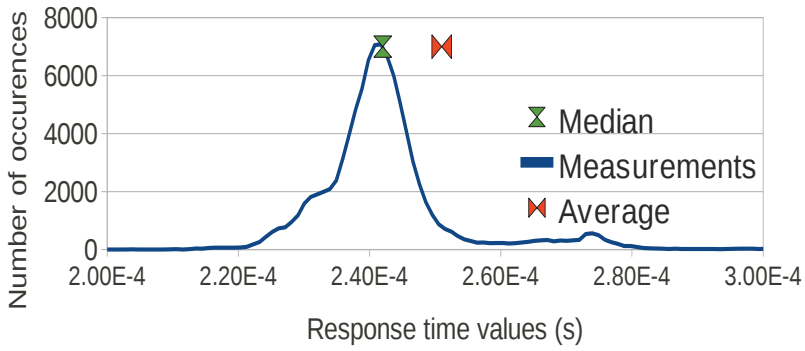
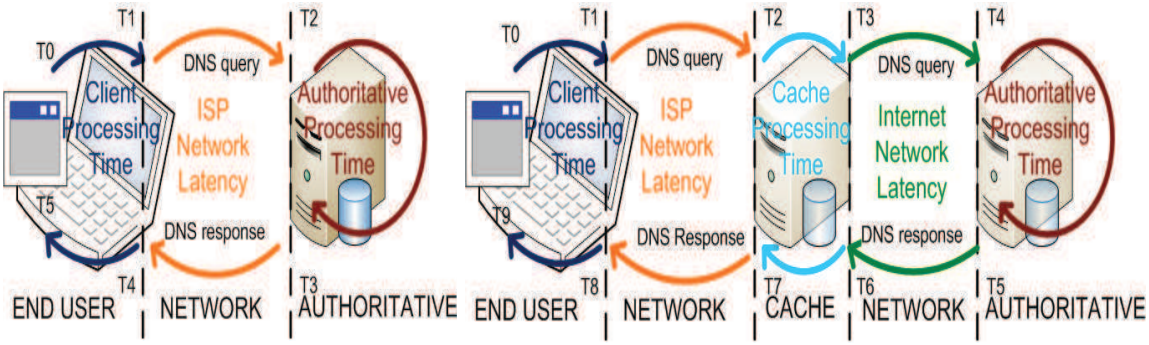


FIGURE 1.1: Experimental Measurement Distribution for a Resolution Time: Difference between Median and Mean Measured Value



(A) Authoritative (B) Resolving

FIGURE 1.2: DNS(SEC) Experimental Platform and Definition of the Measured Time

1.5 Experimental Work to Measure DNSSEC Impact on Servers and End Users

1.5.1 Impact of DNSSEC on Unitary Tests

Unitary tests measure the system performance without load considerations. Figure 1.3 provides time measurements with the same colors or grey level as in figure 1.2. For authoritative servers (figure 1.3a), implementation comparison provides that NSD always has better performance over BIND – 60% for DNS and 65% for DNSSEC. NSD also has lower network latency than BIND for DNS and DNSSEC – 8% for DNS and 7% for DNSSEC.

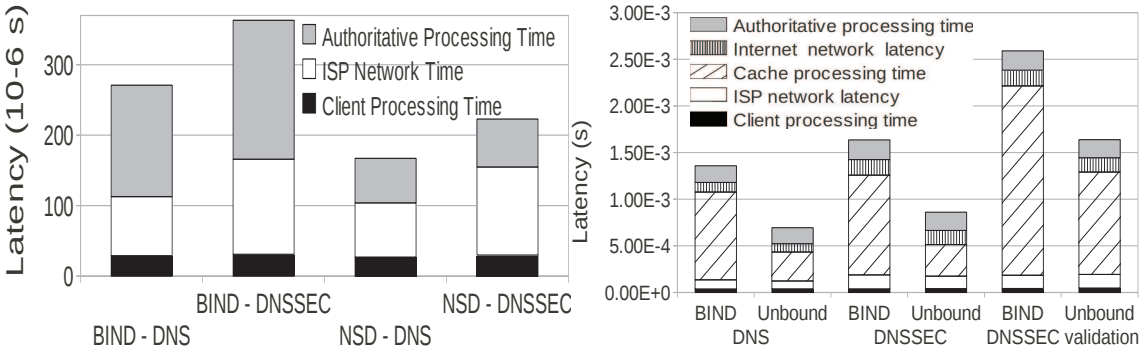
Protocol comparison shows that NSD is less impacted than BIND by DNSSEC – 8% for NSD and 25% for BIND. Network latency also increases by 60% with DNSSEC.

For resolving servers (figure 1.3b), the implementation comparison shows that UNBOUND lowers BIND performance by 67% for DNS, by 68% for DNSSEC without validation and by 46% for DNSSEC with validation. Migrating from DNS to DNSSEC with no validation adds an extra time of 9% for UNBOUND and 14% for BIND. On the other hand, migrating from DNS to DNSSEC with validation adds an extra 253% for UNBOUND and an extra 116% for BIND.

According to unitary tests, NSD is much more efficient than BIND. This can be partly explained by lighter source code for NSD and by the difference of their architecture. BIND10 should enhance its performances by splitting the code for authoritative and resolving servers. BIND10 will provide two distinct pieces of software, and we expect BIND10 to enhance measured BIND's performance. Also BIND and NSD have different architectures, BIND uses a table whereas NSD codes any possible responses. In other words, BIND presents a clear separation between running code and data, whereas NSD includes data and the running code. This is probably the cost of flexibility.

1.5.2 Impact of DNSSEC on Maximum Load

Figure 1.4 shows the CPU load of authoritative and resolving servers. For authoritative servers, considering the maximum load q_{max}, comparison of the different implementations shows that with DNS the maximum load handled by BIND corresponds to 43% of the maximum load handled by NSD. With DNSSEC the maximum load handled by BIND corresponds to 41% of NSD's maximum load. In other words, with the tested configuration NSD is able to deal with around 2.3 times more



(A) Authoritative with *DNS* and *DNSSEC* configurations

(B) Resolving with *DNS*, *DNSSEC* and *DNSSEC with validation* configurations

FIGURE 1.3: DNS(SEC) Experimental Measurements for Unitary Test Latency for BIND and NSD/UNBOUND implementations

traffic than BIND with DNS or DNSSEC.

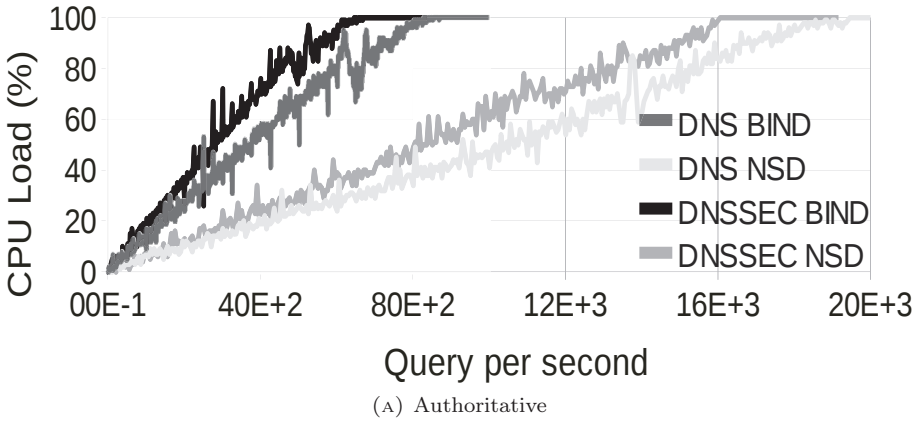
We also measured the cost for DNSSEC migration for each implementation. The maximum load with DNSSEC corresponds to 79% of the maximum load with DNS with BIND and 83% with NSD. In other words, with both implementations BIND and NSD, the cost of DNSSEC is estimated roughly at 30% of the DNS traffic.

For resolving servers, with DNS, the maximum query load handled by BIND corresponds to 28% of UNBOUND's maximum query load. With DNSSEC, resolving servers can proceed to a signature check (DNSSEC validation) or not. With DNSSEC without validation, the maximum load handled by BIND corresponds to 29% of UNBOUND's maximum load. With DNSSEC with validation, the maximum load handled by BIND corresponds to 55% of UNBOUND's maximum load. In other words, with validation UNBOUND is able to deal with around 3.4 times more traffic than BIND. With DNSSEC and validation UNBOUND deals with 1.8 times more traffic than BIND. Validation lowers the differences between BIND and UNBOUND. A possible explanation is that signature check is costly, and has equivalent performance on both implementations.

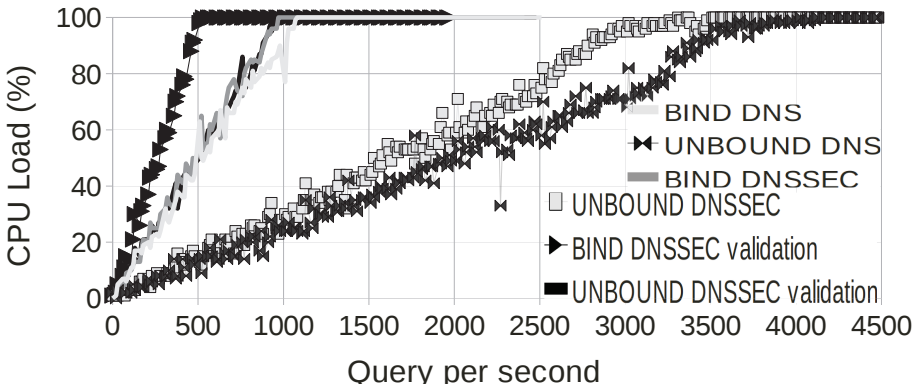
While comparing DNSSEC cost for a given implementation, we can see that BIND with DNSSEC (without validation) the maximum traffic load (without validation) corresponds to 90% of the maximum load with DNS. For BIND and DNSSEC with validation the maximum load corresponds to 49% of the maximum load with DNS. For UNBOUND with DNSSEC without validation, the maximum load corresponds to 86% of the maximum load with DNS. With DNSSEC with validation, the maximum load corresponds to 25% of the maximum load with DNS. In other words the cost of DNSSEC without validation represents between approximately 10% and 14% of the DNS traffic for both implementations. When validation is involved, the cost varies from 75% and 51% of the DNS traffic. The cost of DNSSEC with resolving server varies more across implementation than it does with authoritative servers. BIND has lower performance then UNBOUND, but seems less impacted then UNBOUND by DNSSEC.

1.5.3 Impact of DNSSEC on Network Latency & Response Time

The Response Time is the time is takes the client to get the response from the server, when the query has been sent by the client to the server. Response Time directly impacts the end user.



(A) Authoritative



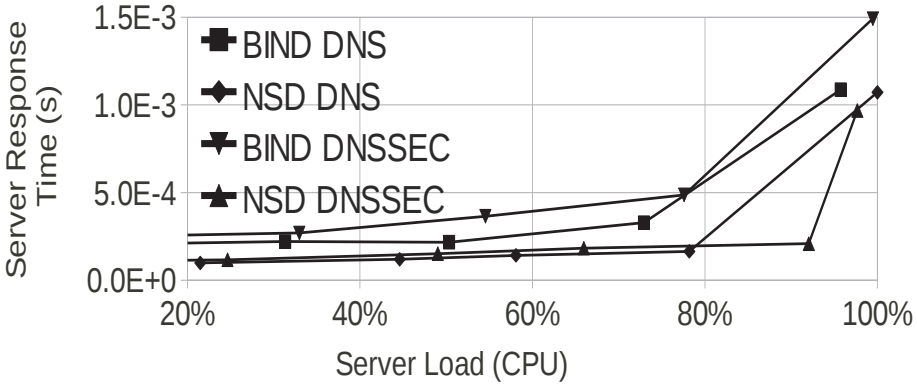
(B) Resolving

FIGURE 1.4: DNS(SEC) Experimental Measurements for CPU Load on Authoritative and Resolving Servers

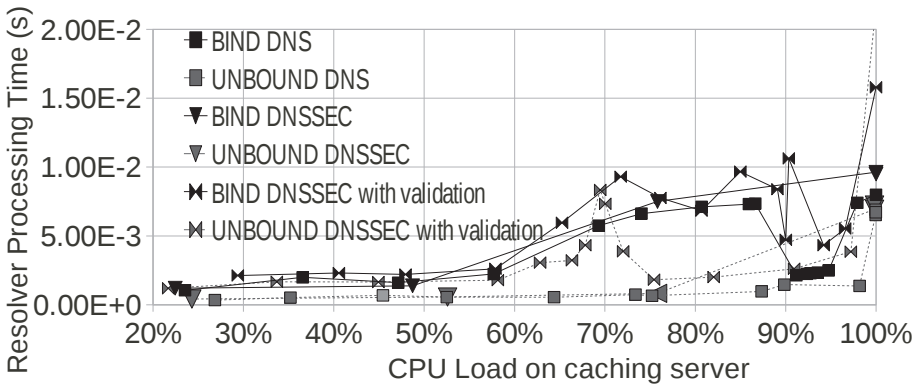
Figure 1.5 provides the server processing time of resolving and authoritative servers regarding the load. For authoritative servers and load below 40%, the response time is quite constant, and NSD response time is around 50% of BIND's response time with DNS and 45% with DNSSEC. Migration to DNSSEC increases response time of 20% for NSD and 10% for BIND. For resolving servers, and CPU time lower than 50%, the response time is quite stable. UNBOUND response time is around 35% of BIND's response time for DNS, 30% for DNSSEC and 75% for DNSSEC with validation. Migration from DNS to DNSSEC, either with BIND or UNBOUND, does not significantly change latency. With validation, migration increases response time by 35% for BIND and 215% for UNBOUND, compared to DNS.

1.5.4 Impact of DNSSEC on DNS Update Operations

Updates are performed using the `nsupdate` command on BIND only (NSD does not handle dynamic updates). Possible operations are : `add` or `delete`. We first compare costs of `add` and `delete` operations. Since `delete` must follow an `add`, to compare the respective cost of those operations, we actually compared $2.n$(`add`) to n(`add` + `delete`) operations. Figure 1.6 shows that $t_{add}^{DNS} = 0.217\,ms$ and $t_{delete}^{DNS} = 0.153\,ms$, so `add` requires 1.42 more time. With DNSSEC $t_{add}^{DNSSEC} = 48.07\,ms$ and $t_{delete}^{DNSSEC} = 11.55\,ms$, so `add` costs 4.16 more time. DNSSEC cost makes `add`



(A) Authoritative



(B) Resolving

FIGURE 1.5: DNS(SEC) Experimental Measurements for Response Time

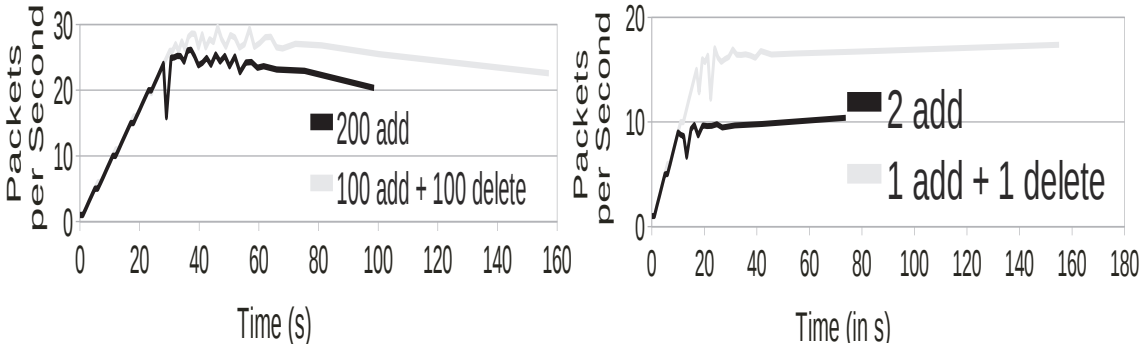
operation 221 times longer and `delete` operation 75 longer.

Tests are performed for one add operation, but `nsupdate` can perform multiple add operations at a time. Figure 1.7 shows that sending multiple updates per `nsupdate` query is more efficient, both with DNS and DNSSEC. From experimental measurements and the Least Squares method we derive a mathematical expression to express the rate of addition to a zone that can be done. This rate is expressed in equation 1.1 as a function of n_{add}, the number of `add` sent per `nsupdate` message.

$$\begin{aligned} Max\_add\_Rate_{DNS}(n_{add}) &= 5000 \times (1 - e^{-n_{add}/40}) \\ Max\_add\_Rate_{DNSSEC}(n_{add}) &= 24 - 10 \times e^{-n_{add}/2.5} \end{aligned} \tag{1.1}$$

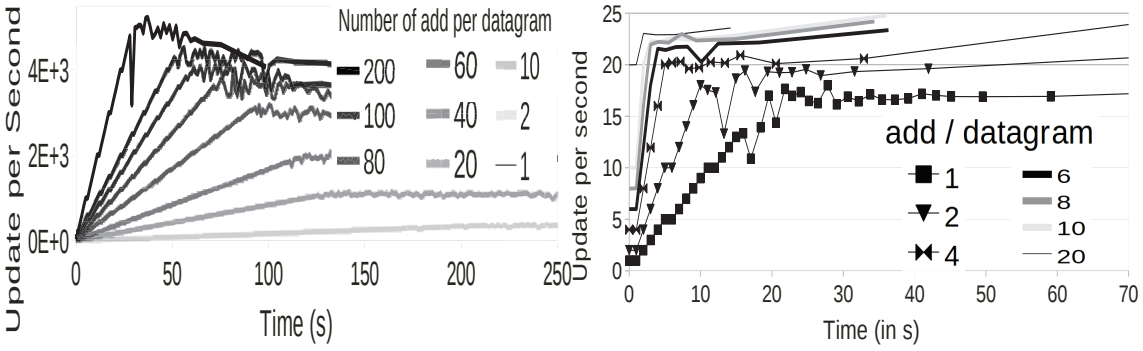
1.5.5 Impact of Cache Hit Rate

Resolving servers have caches and proceed to a resolution only when a cache miss occurs, and all previous tests consider a Cache Hit Rate (CHR) of 0%. To measure the CHR impact on resolving servers, we generate traffic with different CHR and for each traffic we figure the CPU time as a



(A) DNS (B) DNSSEC

FIGURE 1.6: DNS(SEC) Experimental Measurements for Update Rate: Rate Comparison between Delete and Add



(A) DNS (B) DNSSEC

FIGURE 1.7: DNS(SEC) Experimental Measurments for Update Rate with various number of add per nsupdate query

function of the query rate q. Then from the various curves, we computed the Added Query Ratio (AQR) $AQR(CHR) = \frac{q^{CPU}_{CHR} - q^{CPU}_{CHR=0}}{q^{CPU}_{CHR=0}}$. For a given CPU load on the platform, and a given query rate, AQR estimates how CHR increases the query rate q^{CPU}_{CHR} over the measured query rate defined for $CHR = 0$: $q^{CPU}_{CHR=0}$.

To generate a DNS traffic with a given CHR we consider two lists of FQDNs : $FQDN_l$ list with long TTL and $FQDN_s$ list with short TTL, then we load $FQDN_l$ and generate DNS traffic from the two lists as follows : $CHR \times FQDN_l + (1 - CHR)\, FQDN_s$. Figure 1.8 plots result from a CPU time fixed to 100% and shows that CHR is a major parameter on DNS platform performance. As expected, the more CPU time is required for a resolution, the more the CHR enhances performance. As a result, with $CPU = 100\%$, DNSSEC_VAL have an AQR that varies from 1149% to 1779%. DNSSEC and DNS has an AQR varying from 374% to 592%.

For a given implementation, the AQR has similar values with DNS and DNSSEC without validation. Although implementations have different performances with DNS and DNSSEC, the CHR impacts those performance in a similar manner. Comparison across the different implemen-

tations shows that UNBOUND has a greater AQR than BIND with DNSSEC_VAL. With DNS and DNSSEC, BIND has a greater AQR.

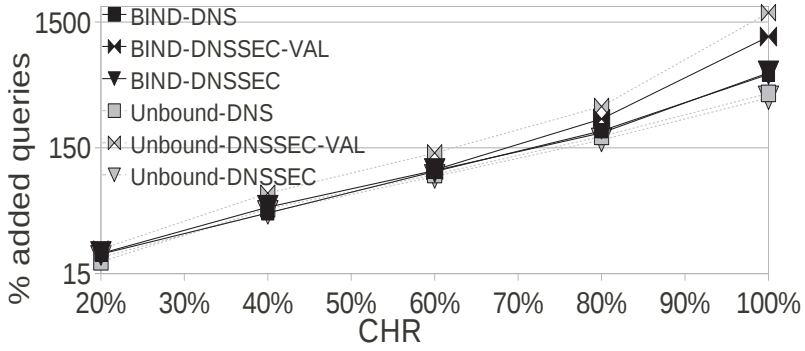


FIGURE 1.8: DNS(SEC) Experimental Measurements: Impact of Cache Hit Rate on Resolution Platform Performances

1.6 Application for ISPs: Estimation of Costs for Migrating from DNS to DNSSEC

In this section, we consider the experimental measurements of the previous sections to figure out how much it costs an ISP to migrate from DNS to DNSSEC. The measured daily query rate on our operational is $40,000\,q.s^{-1}$, with maximum query rates up to $120,000\,q.s^{-1}$. The measured CHR is $CHR = 0.7$. Thus we designed our platform for $40,000\,q.s^{1}$ with a CPU rate of $cpu_{max} = 20\%$.

Tables 1.1 and 1.2 provide, for different configurations, the required number of nodes for each platform as well as the response time for handling the whole traffic. For authoritative and resolving server platform, we provide an estimation for each implementation - `BIND NSD` and `UNBOUND` - considering different protocols - DNS, DNSSEC, or DNSSEC-validation. We also provide the Implementation Ratio (IR) as well as a Protocol Ratio (PR), to compare the implementations performances as well as the costs generated by the different protocols over DNS. Response time represents a good estimation on the end user side whereas the number of nodes provides a cost estimation on the ISP side. One should notice that tables 1.1 and 1.2 considers a platform composed of nodes we used in our experimentation that is to say Pentium IIIs, with 1GHz CPU and 384MB of RAM, which are definitely not the nodes we have in our operational platform. For this reason, we are more interested in ratio between implementations or between configurations. Such approximation must be cautiously considered, and performance tests must be done on the specific hardware used for the operational platform.

For authoritative platforms, migration to DNSSEC impacts more the end users than the ISP. The number of nodes of the platform increases by 33% whereas the response time increases by up to 485%. This latest cost deeply depends on the chosen implementation so ISPs must carefully choose the implementation that fits its required functionalities. For resolving platform costs due to cryptographic operations seems to be the performance bottleneck, thus the more efficient the implementation is the most costly is the migration to DNSSEC in term of number of nodes. In term of response time or the number of nodes, the cost of migrating to DNSSEC-validation can be up to 499%.

How these results can be explained according to our experimental measurements of figures 1.4 and 1.8. First these results are provided according to experimental measurements and equations related to these experimental measurements are provided in section A.1. This paragraph provided a subjective explanation of these results.

Figure 1.4 shows the CPU consumption on our experimental platform which considered a traffic with an associated $CHR = 0$. Then each DNS responses is composed of three fields ANSWER, AUTHORITY and ADDITIONAL, but only the ANSWER field required a signature check. With these hypothesis, experimental measurements show that DNSSEC resolutions requires 1.3 times more CPU than DNS resolution with BIND9 and that UNBOUND requires 3.15 times more CPU for DNSSEC resolutions than for DNS resolutions.

From this, we measured that real traffic has a CHR of 70%, and we considered that each DNS response field ANSWER, AUTHORITY and ADDITIONAL generates a signature check.

Considering 3 signature checks instead of one would make DNSSEC resolutions require 4 times more CPU for BIND9 and 9.45 times more CPU for UNBOUND. On the other hand, considering higher CHR values reduces this cost. In fact, with a CHR of 0%, each DNS query triggers a DNSSEC resolution that results in writing a new answer in the cache. Write operations cannot be done simultaneously by different process. Each time a write operation occurs, the cache is locked, meaning that other write process (that is to say parallel resolutions) are blocked. This results in multiple processes waiting to be active generating multiple interruptions. With a $CHR = 70\%$, 70% of the queries do not need resolutions, and are resolved by reading the cache. These read operations are faster operations, can be performed simultaneously and do not require the cache to be locked. Overall it makes the server much more efficient by reducing the number of interruptions. Finally this high CHR decreases the DNSSEC cost overhead from 4 to 2 with BIND9 and from 9.45 to 5 for UNBOUND.

Node	DNS	DNSSEC	DNSSEC-validation
BIND9	80 (1*)	87 (1.09*)	160 (2.00*)
UNBOUND	20 (1*)	24 (1.20*)	85 (4.25*)
$\frac{\text{UNBOUND}}{\text{BIND9}}$ (IR**)	0.25	0.28	0.53

(A) Number of nodes

Response Time (μs)	DNS	DNSSEC	DNSSEC-validation
BIND9	1402 (1*)	1161 (0.80*)	2300 (1.63*)
UNBOUND	401 (1*)	366 (0.92*)	2000 (4.99*)
$\frac{\text{UNBOUND}}{\text{BIND9}}$ (IR**)	0.20	0.20	0.87

(B) Response Time (μs)

TABLE 1.1: Example of DNSSEC Impact on Resolving Platform for ISPs– (*) PR), (**) IR —query rate $q = 40,000$, Cache Hit Rate $CHR = 0.7$, $CPU = 20\%$

1.7 Conclusion

DNSSEC is deployed to make the Internet more reliable. DNSSEC deployment needs that all Internet actors move forward to DNSSEC, which has already been started by the registries and the software industry. The road to DNSSEC is still long and ISPs as well as other network administrators will have to dive soon into it. This chapter presents keys points to consider when migrating

Node	DNS	DNSSEC
BIND9	29 (1.00*)	38 (1.31*)
NSD	9 (1.00*)	12 (1.33*)
$\frac{\text{UNBOUND}}{\text{BIND9}}$ (IR**)	0.31	0.32

(A) Number of nodes

Response Time (μs)	DNS	DNSSEC
BIND9	239.38 (1.00*)	1161.80 (4.85*)
NSD	92.27 (1.00*)	130.42 (1.41*)
$\frac{UNBOUND}{BIND9}$ (IR**)	0.39	4.50

(B) Response Time (μs)

TABLE 1.2: Example of DNSSEC Impact on Authoritative Platform for ISPs – (*) PR, (**) IR)

to DNSSEC, so to ease planning this migration.

At first one must be aware that DNSSEC is not a trivial option. People must plan this migration and consider DNSSEC as a new protocol with its own issues, its own engineering rules... rather than an option of DNS. However DNS is still compliant to DNSSEC which eases the transition, and migration should be much faster than IPv4 to IPv6. Then people should not underestimate the change on the operational procedures. This includes the signing procedures for authoritative servers, but also monitoring both traffic and deployed DNSSEC zones - at least at the beginning, so to avoid false positives. Then, DNSSEC deployment on resolving infrastructure should be done step by step, and opt-in trial is probably the most relevant thing to start with. More specifically, opt-in trial involves people that are aware of the technology, or that are willing to use it, and ready to debug encountered corner cases. At last, whatever difficult DNSSEC migration is now, DNSSEC is on its way to be deployed and migration will become even harder in the future. For this reason one should start now and small.

Future work should include distributed loader to check large operational platform with real DNS traffic. Traffic monitoring and traffic analysis should also anticipate DNSSEC issues, and provide input for new platform architecture. In fact, with high CPU occupancy resolution DHT-based architectures may outperform traditional resolving servers architecture.

Chapters 2 and 3 are both dedicated to optimize a Resolution Platform. Platforms are designed to avoid simultaneous resolutions by different nodes of the platform. Thus we assign for each FQDN a Responsible Node. DHT architectures are evaluated and we take advantage of the law power distribution of the FQDNs.

Chapter 2

Overcoming DNSSEC Performance Issues with FQDN Load Balancer and Cache Sharing

This chapter reflects some of the work we performed on this topic. It has provided inputs for [MHS+b, MHS+a, XMSF11, ML11, FMS11, FMS12]

2.1 Introduction

DNS [Moc87a, Moc87b] is the protocol that makes possible communication based on names, by binding an IP address to a Fully Qualified Domain Name (FQDN). As such end users rely on a DNS resolving platform to determine the IP address of the target, which makes DNS resolution platform a critical element of the Internet.

Most of the time, when an end user types *www.facebook.com* in its browser, the browser sends a DNS query to the DNS resolving platform of the ISP, which sends back the IP address of *www.facebook.com*, so that the browser initiates an http connection to the web server. As a result, ISP DNS resolving server have become strategic points over the Internet reached by numerous end users and that define where they will be connected to. In July 2008, Dan Kaminsky showed that DNS was sensible to cache poisoning attacks [Kam08b, Kam08a], and that the long term solution was DNSSEC [Hig09]. DNSSEC [AAL+05a, AAL+05c, AAL+05b], was designed by the IETF so the end user can check if the received DNS responses are legitimate through electronic signatures. As a consequence, DNSSEC resolution costs a lot more than a DNS resolution. Note the Top Level Domain (TLDs) have already deployed DNSSEC, making the DNS infrastructure DNSSEC ready. Then, end users Operation System like Window7 is also DNSSEC ready, which makes ISPs upgrading their DNS resolving platform to DNSSEC. Lab tests and experimental DNSSEC trials by major ISPs (Orange, Comcast) [MGL10, Gri09] show that DNSSEC deployment on their resolving servers requires between 4 and 5 times more resources than with DNS. Thus, for a DNS Resolving Platform - like it is the case for Orange - currently composed of 18 nodes, migration to DNSSEC with the current architecture would require 90 nodes. Administrating 20 nodes versus 90 comes with management as well as organizational issues, that makes DNSSEC hard to be considered by some ISPs. As such, the primary goal of this chapter is to provide an architecture for a resolving platform that optimizes resources required for the DNSSEC resolutions. In addition to the migration performance issue, scalability of the resolving platform is also a priority as traffic keeps

increasing by 8% per month for the last five years. Moreover, Content Delivery Networks (CDNs) are more and more popular among ISPs. In fact ISP are heavily involved in deploying CDNs to deliver content to their end users, in their home network. CDNs usually load balance the traffic on their content servers by using the DNS with FQDN whose Time To Live (TTL) is in the order of 30 s. Both CDN development as well as short TTL are expected to increase drastically the DNS traffic.

As such, this chapter looks for alternative architectures adapted to the upcoming DNS(SEC) traffic. ISPs have different architectures for their DNS resolution platform. Some have a decentralized architecture where the nodes of the platform are located in the Access Network. Others have a centralized platform composed of one or few clusters, which responds to all end users queries, like Orange having 18-node clusters. In this chapter we consider the latest architecture where load balancers are in charge of splitting the DNS queries among the n different nodes. For each incoming query, the load balancer XORs the IP source and destination of the query, and then performs a modulo n operation over the 24 least significant bits [alt03, Rad10]. This traditional architecture is referred in this chapter as IP_{XOR}.

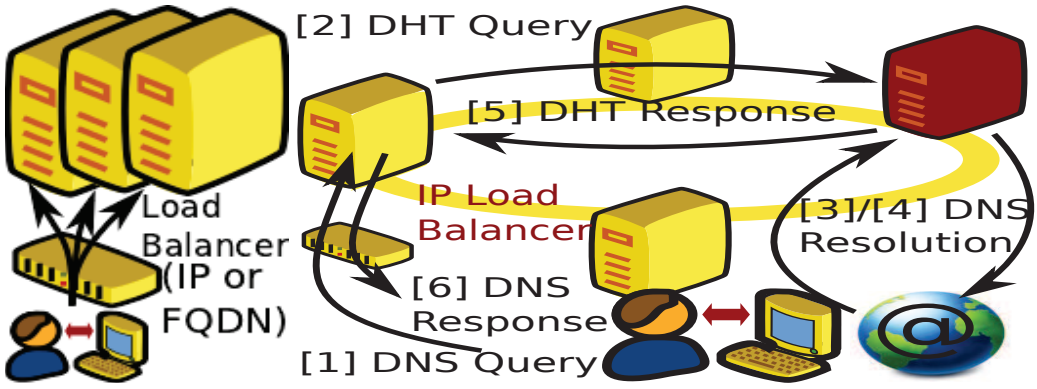
This chapter considers an alternative architecture *FQDN* where load balancers split the DNS queries according to the SHA1 value of the FQDN as represented in figure 2.1a. Compared to IP_{XOR}, we expect *FQDN* to be more efficient, more scalable and to better protect the end users' privacy. Firstly *FQDN* is expected to be more efficient as it implicitly assigns each FQDN to a single *Home* node that is responsible for performing the resolution for that FQDN. Secondly, the number of resolutions is expected to decrease, thus reducing the CPU consumption. Note that the efficiency of the DNS resolution platform is measured according to the global CPU consumption and the distribution of the CPU load among nodes. An optimal platform has the CPU load uniformly distributed among the nodes.

In fact if you consider that τ is the maximum number of tasks a node can perform. If tasks are uniformly distributed, then a n node platform is designed $n\tau$ tasks. On the other hand, if one node always performs twice as the other nodes then the n node platform can only perform $n - 1\frac{\tau}{2} + \tau$ tasks. In our case, tasks are either resolution (R) or Cache Hit lookup (CH). Each FQDN is associated a Time To Live (TTL) that indicates how long the FQDN is stored into the cache. With a traffic analysis of DNS traffic we considered the distribution of Queries, and TTL, so to evaluate whether distributing tasks on a per FQDN basis may introduce a bias compared to a per IPs distribution. This chapter shows that *FQDN* improves the IP_{XOR} platform efficiency by 30%, thus making *FQDN* more scalable. Moreover, a newly added node in IP_{XOR} has to perform resolutions that includes the most popular FQDNs, even if the other nodes of the platform have already resolved them. Adding a node in *FQDN* does not generate redundant resolutions. Finally, *FQDN* better protects the end user's privacy as the traffic is load balanced according to the FQDN and not the IP address. Additionally, if a routing table is established based on a traffic analysis, the privacy of end users is preserved for *FQDN*, but not for IP_{XOR} as statistics on IP addresses are needed.

Even though *FQDN* offers multiple advantages over IP_{XOR}, *FQDN* presents two major drawbacks. First load balancers based on FQDN are unusual on the market. Second, FQDN Load Balancer must be deployed in conjunction of other equipments whose purpose is to define proper balancing rules according to the running traffic. Third, replacing the load balancer with another one in the current platforms requires modifying the core network, and thus making network architects more reluctant for a transition from IP_{XOR} to *FQDN*. To overcome these issues, we consider a multi-node platform where each FQDN has been assigned a *Home* node, and each node is able to find the *Home* node of any FQDN. This provides the main *FQDN* principle which is splitting FQDN naming space among different nodes. Routing queries to their *Home* is performed by the nodes themselves rather than the load balancer. Contrary to *FQDN*, platform nodes have two functions: resolving DNS queries and routing queries to the *Home* node. In this chapter, such architectures are modeled with Pastry [RD01a] as represented in figure 2.1b.

In this chapter, Pastry is used for a maximum of few hundred nodes, while Pastry was origi-

nally designed for very large platforms of thousands or billions nodes. As such, we do not consider the routing discovery mechanisms provided by Pastry, but we take advantage of Pastry auto-configuration capacity, its robustness to DoS, and its cache sharing mechanisms. By comparing different cache sharing mechanisms, we show that active cache sharing increases the platform efficiency by more than 3.5.



(A) $FQDN, IP_{SHA1}, IP_{XOR}$ (B) DHT (Pastry)

FIGURE 2.1: DNS Resolution Platform Architectures base don Load Balancers and on DHT

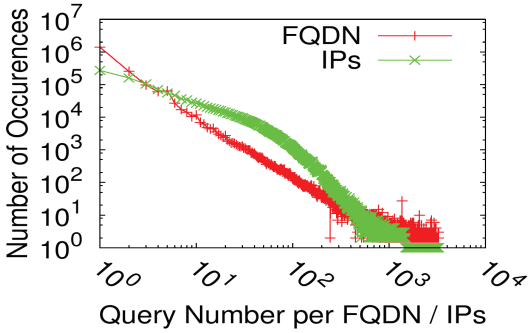
Note that this chapter considers how multiple nodes can cooperate and share their cache, so to improve the efficiency of the DNSSEC resolution platform. The efficiency of the proposed platform results from the combination of the following features: FQDN load balancing, Cache sharing and active caching. This chapter points the key features for efficiency and propose one way to implement them. Other alternatives to implement those features or a subset of them may exist, for example by taking advantage of multi-cores environment and hardware card specificities or by taking advantage of existing products such as Global Traffic Manager [FN10] where cache may be pre-populated with the very most popular FQDN.

The chapter is organized as follows. Section 2.2 compares characteristics of FQDN and IP addresses of DNS queries based on real DNS traffic captures. It shows that using FQDN for load balancing makes sense and can lead to further considerations. Section 2.3 positions our works. Section 2.4 simulates, with real DNS traffic, different types of load balancers that consider - like IP_{XOR} - the IP addresses of the DNS queries, or - like *FQDN* - the FQDN. *FQDN* is shown to be more efficient by up to 30%. Multiple motivations make us considering DHT based architectures for our platform rather than a single hardware FQDN load balancer. Section 2.5 describes how we model various DHT Pastry based architectures - with different cache sharing mechanisms. Section 2.7 gives simulation results about the different modelized DHT architectures and shows that the proactive cache sharing improves the platform efficiency by 3.5 compared to the traditional architectures.

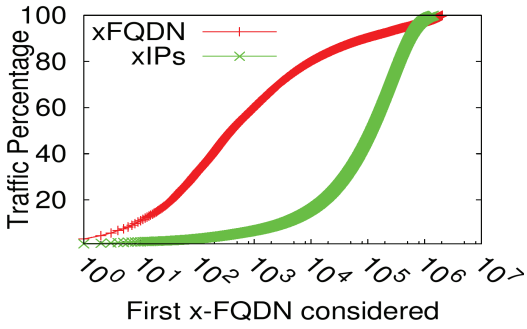
2.2 FQDN or IPs as Load Balancer Criteria

This section compares two load balancing strategies for splitting DNS queries among the nodes: the IP addresses of the queries as currently used in DNS platforms, and the queried FQDNs. It shows that it makes sense to consider FQDNs instead of IP addresses. For that purpose, we consider a 10 minute live DNS traffic capture at the rushing hours (19 : 33) on our 18 node platform.

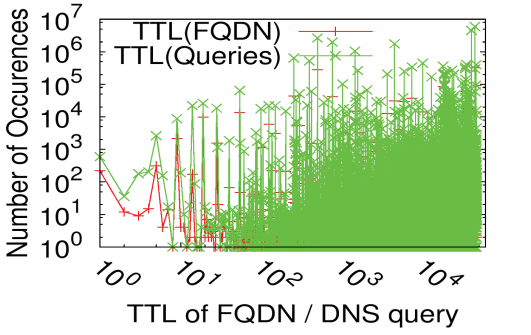
Figure 2.2a (resp. figure 2.2b) illustrates the distribution (resp. the cumulative distribution) of the queries associated to the FQDNs or IP addresses of the queries. The cumulative distribution shows the percentage of traffic represented by the x first most popular FQDNs. This distribution is also known as a Zipf distribution [BCF+99, JSBM01] : There are few very popular FQDNs whereas most of them are almost never requested. However, we did not find any mathematical proof for it. It suggests that FQDN and IPs have similarities, but one can also notice that difference between FQDN popularity is more important than end user querying ability. TTL is the time a FQDN stays in a cache, so TTL, in conjunction with the FQDN popularity influences the Cache Hit Rate (CHR). Figure 2.1c represents TTL distributions among the FQDN space (FQDN) - i.e. for a given TTL value how many different FQDN we have -, and among the DNS query space (IP) - i.e. for a given TTL value, how many DNS response, whatever the FQDN is, we have. It shows that TTL distribution in FQDN space(FQDN) and TTL distribution in DNS query space (IP) are similar. Thus $FQDN$ does not introduce major bias on the CHR over IP_{XOR}. Then, figure 2.1d plots for each FQDN the TTL value as a function of its popularity or rank. The larger the popularity of a FQDN is, the larger the TTL variation is. Popular FQDN have TTL similarities, which may not be considered for less popular FQDNs.



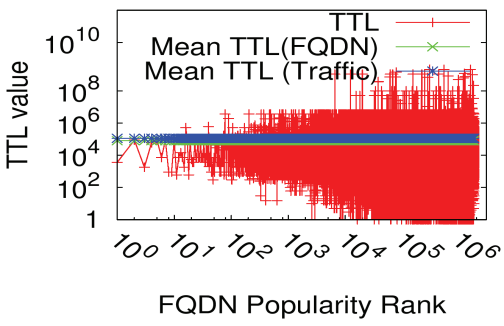
(A) Query Distribution



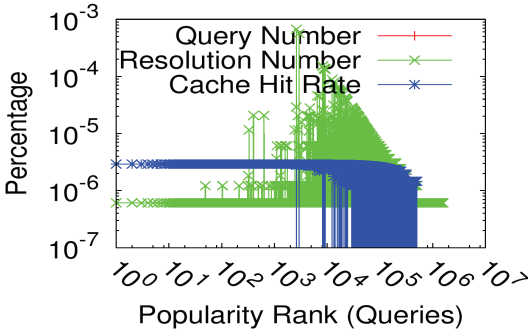
(B) Cumulative Traffic Distribution



(C) TTL Distribution



(D) TTL vs. FQDN Rank



(E) Query/Response/CHR vs. FQDN Rank

FIGURE 2.0: Live DNS Traffic Analysis: Comparison of FQDN and IP addresses Distributions

2.2.1 Measured DNS(SEC) CPU^R and CPU^H

When a node receives a query, then it performs a cache lookup. If the response is in the cache then the resolving node sends back the response. If the response is not in its cache, the node performs a resolution and then inserts the response in the cache. The difference between the two cache operations is that cache lookup only requires read access to the binary tree, whereas the insertion requires write access, and so lock the binary tree for some time. This explains the difference between the CPU Time when a resolution is performed (CPU^R) and when a cache lookup is performed (CPU^H). Those values are summed up in table 2.1b. Equation 2.1 expresses the CPU load as a function of CPU^R, CPU^H, and the Maximum Load (ML). From figure 2.1f and [MGL10] we derive a mean CPU per request for different values of the Cache Hit Rate (CHR) (table 2.1a). More specifically, with $CHR = 0$, any query triggers a resolution. This means that for any query a cache lookup is performed, followed by a resolution. With $CHR = 100\%$, any query triggers a cache lookup. With equation 2.1 we derive for each implementation (BIND9 and UNBOUND) and protocol (DNS, DNSSEC, DNSSEC_SIG) CPU^R and CPU^H in table 2.1b. DNS is the traditional DNS protocol, DNSSEC carries DNSSEC responses, but do not consider the signature check. DNSSEC_SIG on the other hand considers the signature check.

$$CPU_{max} = ML(CPU_H + (1 - CHR)CPU^R) \tag{2.1}$$

To evaluate the cost of a three signature check response from a single signature check response, we considered that CPU^R is composed of two tasks: one that consists in handling the packet and inserting the data into the cache, and the other one that consists in checking the signature. [MGL10] shows that DNSSEC without validation has a 10% overhead on maximum load. Thus we approximate $CPU^R_{DNSSEC\_NO\_SIG} \approx 1.10 CPU^R_{DNS}$. Then we considered $CPU^R_{DNSSEC} = CPU^R_{DNSSEC\_NO\_SIG} + CPU^R_{SIG\_CHECK}$. We derive $CPU^R_{SIG\_CHEK}$ (0.07654052 % CPU for BIND9 and 0.073271687(% CPU for UNBOUND) and then $CPU^R_{DNSSEC,3\,SIG\_CHECK}$ we summed up in table 2.1b.

Cost	DNS	DNSSEC	DNSSEC_SIG
BIND9			
$CHR = 0\%$	0.1030927	–	0.188679245283
$CHR = 80\%$	0.0347222	–	0.053191489361
$CHR = 100\%$	0.0148809	0.0151057	–
UNBOUND			
$CHR = 0\%$	0.02688172	–	0.10204081
$CHR = 80\%$	0.00984251	–	0.02433090
$CHR = 100\%$	0.00566251	0.005427997	–

(A) Measured % CPU per Request

	BIND9	UNBOUND
Single signature check		
CPU^H_{DNS}	0.014880952	0.005662514
CPU^H_{DNSSEC}	0.015105740	0.005427997
CPU^R_{DNS}	0.088211799	0.021219210
$CPU^R_{DNSSEC\_NO\_SIG}$	–	–
CPU^R_{DNSSEC}	0.173573505	0.096612818
$\frac{CPU^R_{DNSSEC}}{CPU^H_{DNS}}$	11.66413983	17.06182412
$\frac{CPU^R_{DNSSEC}}{CPU^R_{DNS}}$	1.967690342	4.553082701
Three signature checks		
CPU^R_{DNSSEC}	0.317833377	0.241034271
$\frac{CPU^R_{DNSSEC}}{CPU^H_{DNSSEC}}$	19.41673043	38.69695766
$\frac{CPU^R_{DNSSEC}}{CPU^R_{DNS}}$	3.603071024	11.35924810

(B) CPU^H and CPU^R

TABLE 2.1: Deriving CPU^H CPU^R in various DNS(SEC) configurations from Experimental Costs Measurements

2.3 Position of our Work

This chapter shows that using a DHT based architecture with proactive cache mechanisms can improve DNS resolving platform by 2.2. Proactive caching takes advantage of the Zipf distribution of the FQDN in the DNS traffic. Thus we position our work toward previous work first on DNS traffic analysis. Then we consider work that optimizes the tasks performed by the resolving nodes, that is to say optimization of Caching mechanisms and Resolution mechanisms. Note that those previous work do not consider DNSSEC, and is mainly concerned by optimizing the latency rather than the CPU Time. At last we consider work that provides a Distributed Hash Table for the Naming service. Here again DNSSEC was not deployed, thus only DNS was considered. Most mechanisms are optimizing the latency of the architecture as well as its robustness. Then DHT was considered as an unlimited database, and most of the work is considering a DHT architecture to host the zone files rather than for a resolving platform. Multiple mechanisms have been designed so to optimize the DHT architecture. The one we are more interested in is the Active Caching that takes advantage of the Zipf distribution of the DNS traffic.

2.3.1 DNS Traffic Analysis

[BCF+99] analyzes web traffic distribution and shows it follows the Zipf law function. Zipf law function considers that given the popularity rank i of a page, the probability that a request concerns a page of rank i is $P_i = \frac{\Omega}{i^\alpha}$. To check these distribution properties, it builds a model, assuming requests are independent, and derives a few asymptotic results that can be measured with the real captures. First based on its model, it derives an analytical expression of the Cache Hit rate for an infinite cache with finite request stream as well as a for a finite cache size with an infinite request stream. Then still from the models, it derives the distribution of inter-arrival time. Considering the infinite cache with finite request stream leads to the expression of the Cache Hit Rate as a function of the number of Request. For the finite cache size with an infinite request stream, this leads to the expression of the CHR as a function of the cache size. The distribution of inter-arrival time leads to the expression of the probability that the next request for a given page of rank i is k requests later. Zipf-like functions are characterized by the alpha parameters, and we then have three different ways to derive the alpha parameter, and check how their value is similar. The model matches asymptotic results.

Our chapter considers the DNS traffic, not web traffic. However, as mentioned in the conclusion, Zipf-like function seems inherent to any web access streams, and thus DNS should follow this distribution. Similarly, the model is expected to match properly asymptotic results, but finer predictions may have to consider inter-dependence between the data. The caching models are not valid anymore since DNS data are cached according to the TTL value, and TTL are values associated to the data, as opposed to a local cache policy.

2.3.2 DNS Cache Optimization

[DBI03] describes a multi-layer cache architecture where multiple local DNS resolving servers check their cache. If a cache miss occurs, the resolving server performs a cache lookup to a centralized cache. If the requested data is still not in the centralized cache then local DNS servers perform a resolution and update the centralized cache. This technique reduces the number of resolutions up to 26% for corporate environment. If this technique would be applied, it also expected to decrease the load of Top Level Domain (TLD) authoritative servers by 40%. [DBI03] aims at reducing the number of resolutions. However, this is intended in order to reduce the Resolution Time or latency of the end user rather than to reduce the load of the platform. More specifically, ISPs have to deal with heavy traffic, and it is not evident that the centralized database could support the multiple parallel read or update operations.

[JSBM01] shows how important caching is for the DNS. It starts by analyzing data collected from university campus of MIT and KAIST. It shows the log distribution of the Cumulative Distribution Function (CDF) for latency. It shows that only a small number of requests that have a low and very large latency resolution. It then shows how the number of referrals (NS) and how NS Cache Hit (CH) or Cache Miss (CM) affects the CDF of latency. It also analyses unanswered queries, as well as negative responses. In the section "Effectiveness of Caching", the paper intends to answer to the following questions:

- How useful is-it to share DNS cache among many client machines?
- How TTL impacts the caching effectiveness of the architecture?

It shows the domain name popularity (i.e. the number of queries associated to a FQDN) follows a Zipf law. It also plots TTL distribution as well as TTL distribution weighted by popularity. Then the paper studies how Cache Hit rate is impacted by considering smaller groups of end users, and shows that most of the benefits are obtained with as few as 10 or 20 end users, which makes cache sharing not so efficient. To show how TTL impacts the Internet architecture, the paper

considers a single TTL value for all FQDN and looks at the inter-arrival time. The inter-arrival time log distribution shows TTL values have few impacts on the CHR. However TTL value of NS records should not be significantly decreased. [JSBM01] shows that considering large group of end user does not significantly the CHR properties compared to small groups of 10 or 20 persons. This can be explained by the power law distribution of FQDN popularity. End users request the most popular FQDN, and others are only requested by a single user. This suggests that however numerous is the group of end user, name servers will resolve the same most popular FQDN, and that all the other names are only resolved by single end users, thus leading to a resolution. This shows that in order to optimize resources of the platform, the traffic should be split according the FQDN rather than by IP addresses. Dedicated servers would treat very popular FQDN the others the isolated requests.

[JBB03] looks at the inter-arrival time of DNS queries and shows the distribution of CHR(TTL) follows a Pareto distribution law. As a result the CHR increases with TTL reaching 80% for a TTL = 15 *minutes*, and with very little increase of CHR for larger TTL. [JBB03] shows that TTL has an optimum value (15 *min* that balances CHR maximization and refreshment of the information. However, CDN, and many popular web sites do use TTL values that are far below that value 30 seconds , and the goal of TTL for wed administrators is more to load the traffic then to optimize DNS cache. As such TTL may not be chosen adequately so to benefit from CHR. It is true that there is a close relation between TTL and CHR, however resolution platform undergo such relation rather than being able to take advantage of it.

2.3.3 DNS Resolution Optimization

[PPPW04b, PPPW04a] describes CoDNS and looks how cooperation between resolvers can improve the DNS resolution. The main idea is that when one resolver cannot perform a DNS resolution, it asks its neighbors to perform the resolution on its behalf. This provides significant improvements because, different resolvers have different policies on which DNS servers to query as well as different locations and thus different network congestions. The papers start by considering the different distribution of DNS responses time for various DNS resolvers. From the Cumulative Distribution Function (CDF) it derives criteria that help defining when a resolution takes too much time. It then points out what on the resolver side can make resolution fail - among them are packet loss, Name server overloading, Resource competition, maintenance problems:. It also shows that over a large distribution of nodes, at any time we have above 90% of the resolvers that are healthy. It then measures the performances of CoDNS vs. the traditional resolution. More specifically, it shows that with even only two resolvers involved in the CoDNS, delays in resolution is significantly improved (between 27 and 82%).

The goal of CoDNS is to improve DNS resolution and improve its reliability. Resolution are performed by multiple nodes, since we saw that there is a high probability that bad performance resolution do not occurs simultaneously on two different nodes. Our work considers an architecture point of view and defines how functions (routing/ resolution) should be distributed among the nodes. CoDNS is a way to improve the resolution function. The goal of our architecture is too minimize the signature operations between the nodes of the platform, and to load balance the traffic so that every node performs the same tasks. CoDNS should be considered in the next step and future work should consider how CoDNS could improve the resolving function of the platform we design in this chapter.

2.3.4 DNS service over DHT overlay

Most popular DHT protocols are Chord [SMK+01], Pastry [RD01a], Tapestry [ZHS+04] and CAN [RFH+01], where nodes are self-organizing, leave and join the ring with no extra burden for the administrators. Pieces of data are randomly spread among nodes, thus resulting in a greater robustness against DDoS attacks [CMM02] describes a DNS service based on Chord [SMK+01] and DHash [DKK+01]. DNS was run over DHT so to get rid of painful name server administration, and inherit good load balancing and robustness from DHT architecture. The paper reports an experiment with 1000 nodes in the Chord ring serving as authoritative server. Replication was turned off, which means that the information stored in a node is not replicated on k other nodes. However, this architecture considers passive caching, which means that when a node performs a lookup, it stores the answer. DHash is block driven, which means that files in our case DNS Responses are not hosted by one node, but blocks of the file may be distributed over multiple nodes. DHash as well as caching mechanisms provide a well load balanced traffic over the nodes. However, even though DHash and PAST have different design, since the size of CFS blocks of DHash are up tens of kilobytes, in our case, DHash and PAST have more or less the same results. The major issue of the DNS over Chord architecture is that the latency is much too high.

[Mas06, RHM09] analyze how DHT could enhance the robustness of the Naming System. The robustness of both Chord and DNS considers *Data failure rate*, *Path failure rate* and *Path length*. The DNS efficiency was proved to be linked to the popularity of its zone and the number of labels of the domain name, whereas the DHT efficiency is related to the popularity of its RRsets. In fact, DHT main drawback is its heavy routing algorithms. DHT is also more robust to orchestrated attacks and could achieve the same availability of the current DNS with added mechanisms like proactive caching - Beehive [RS04].

Our chapter differs from those papers since the DHT ring is used for hosting authoritative data, whereas our architecture uses the DHT Pastry [RD01a] as a way to define which is the node that performs the resolution. As such we do not use caching of the data. The way the data is stored into the DHT also differs from DHash to PAST. In DHash, blocks of the files are spread over the DHT nodes, whereas in PAST the whole file is hosted on the node. Our architecture is also expected to be at maximum at around one or two hundred nodes, whereas the DHT considered in [CMM02] considers a 1000 node experiment which is mentioned as being a restricted number.

2.3.5 Alternate Naming Architecture

[WBS04] proposes the *Semantic Free Referencing*, a Naming Architecture to avoid the Web being constrained by the DNS. The increase of demands for the Web burden the DNS, which results in slowing the Web. For this reason [WBS04] proposes to name Web objects with a naming system that differs from the current DNS: the *Semantic Free Referencing*. The purpose of this architecture is to use references that are free of semantic —unlike FQDNs for example. This makes the design of the architecture only concerned on the technical aspects of the resolution, trademarks copyright are out of scope. Then *SFR* also addresses the issue of Web object migration. With current DNS and HTTP infrastructure, Web object migration requires redirections either on the DNS server or on the HTTP server. These configurations are only available to the owner of the DNS zone files or web servers, but not the owner of the web content. Similar issues are found for replication. Thus, *SFR* consists of web object binary identifiers. Unlike FQDNs, these identifiers have no human meanings and are names for web objects, instead of servers. This granularity provides much more independence for each Web Object. These binary identifiers indicates the current location, or redirect to another binary identifier.

These binary objects are hosted on a global DHT platforms. Note that these DHT platform *are not not relying on flaky personal machines connected via cable modems!*. Although there are work around to enable navigation on a domain while being disconnected from the Internet, the

basic architecture requires a connectivity and connection on the global DHT platform. Because all information are shared on a common global platform, the architecture is called *fate of share*. Information provided by the identifier can be checked by the client, by providing public key and signatures. This makes possible to prove the ownership of the content. Similarly, when the owner of the identifier updates the locations proof-of-ownership uses the public.

The common point with our work is that the architecture is based on DHT. We use Pastry, they are using Chord and DHash, but in both cases, the architecture may be adapted to one or the other protocol. More specifically, in both cases, the platform is well administrated by a single entity. Pastry provides neighbor discovery protocols, but we are not taking advantage of it. We choose Pastry mostly because we have a Java based implementation, then Pastry hosts the whole content on the same node, whereas DHash may split a file across multiple nodes. Our work differ from [WBS04] because, we look at improving the performances of the DNS Resolving platform whereas [WBS04] proposes an alternate naming infrastructure. Our motivation for improving the platform is the migration to DNSSEC, whereas DNSSEC is not mentioned in the paper. As mentioned in the paper, DNS(SEC) is still expected to co-exist with the new naming infrastructure, and our paper looks to improve the performances of resolving platforms.

2.3.6 Active Caching for DHT architectures

Beehive [RS04] replicates data that follows Zipf-like distribution, to achieve O(1) lookup performance over a structure DHT such as Chord [SMK+01] or Pastry [RD01a]. Latency of structured DHT is in $O(ln(n))$, and Zipf data's popularity is $i^{-\alpha}$. Thus replicating the most popular data exponentially reduces latency. Beehive describes how to choose the minimum subset of data that should be replicated at different level so to achieve a 0(1) lookup. Simulations are provided with DNS traffic, and performances of Pastry, Passive-Caching Pastry and Beehive are compared. With a platform composed of 1024 nodes, PC-Pastry devises Pastry latency by 1.5 whereas Beehive divides Pastry latency by 2.3. However Beehive takes 2.3 times more than PC-Pastry to calibrate. The total number of objects transferred by PC-Pastry is proportional to time, whereas it looks more like a log function for Beehive. This shows that Beehive reduces the number of transferred objects. Because data have a Zipf-like distribution, the number of stored object is quite small. It also shows that Beehive is quite adaptive to changes in traffic.

[RS07] provides results based on Beehive for multiple applications with Zipf-like popularity distribution, more specifically DNS, WEB and gnutella. For all those traffics it plots the latency evolution with Pastry and Beehive. It shows that Beehive behaves in a similar way with all those traffics, and that it can reduce the latency up to 0.5 hops whereas Pastry incurs a 2.5 hop latency.

[RS04, RS07] both deal with DNS traffic. Although the main objective of the papers is to reduce latency by replicating the most requested data to multiple nodes. The proposed architectures are based on DHT which assigns FQDN to specific node as in our chapter. However, the architecture is composed of more than thousands of nodes, and designed to replace authoritative servers rather than resolving servers. Some of our architectures are also based on Pastry, however, we only consider the self organization properties of Pastry. Our platform is of small size (less than two hundred nodes) and we are not concerned with the number of hops, since all nodes are neighbors. Furthermore, our platform is not used only for storage, but performs DNS or DNSSEC resolution. In fact Beehive has been designed for authoritative servers. It takes a large benefit over passive caching by considering updates of the data, rather than the TTL value. As a resolving platform we can only rely on TTL values so to consider the data valid. So Beehive has not been designed for a resolving platform whose information validity is based on TTL. However, the replication principle could be one alternative to reduce the number of resolution to be performed. In fact by assigning a FQDN to a single node reduces the number of resolution. Since on resolving platform updates are based on TTL, the number of updates might generate some non negligible traffic among the

nodes as well as some non negligible computing resources to update caches of each node. Beehive principle is to propagate the requested data where the queries can be. As such, Active Caching may be a more efficient alternative to the Passive Caching mechanisms of the DHT.

[CS02, LCC+02] provides replication and search strategies for unstructured Peer-to-Peer networks. Such architectures are out of scope of our study. In fact our architecture is considering DHT in a totally controlled environment.

2.4 Traffic-based Load Balancer Simulation

This section evaluates the efficiency of different load balancing strategies from the same live DNS capture described in section 2.2. It shows that balancing traffic on the platform according to FQDN rather than IP address requires 30% less resources.

Evaluation of each strategy considers the distribution of the queries, the resolution, the Cache Hit Rate (CHR) and the CPU Time (CPU) on the 18 nodes of the platform. The efficiency of the different strategies is measured by plotting the distribution of CPU of the different nodes of the platform, which shows the global CPU of the platform, as well as the dispersion of CPU among the nodes. This section provides two ways to figure out the distribution of CPU of the nodes. One is derived from the CHR distribution, the other is computed from a simulation.

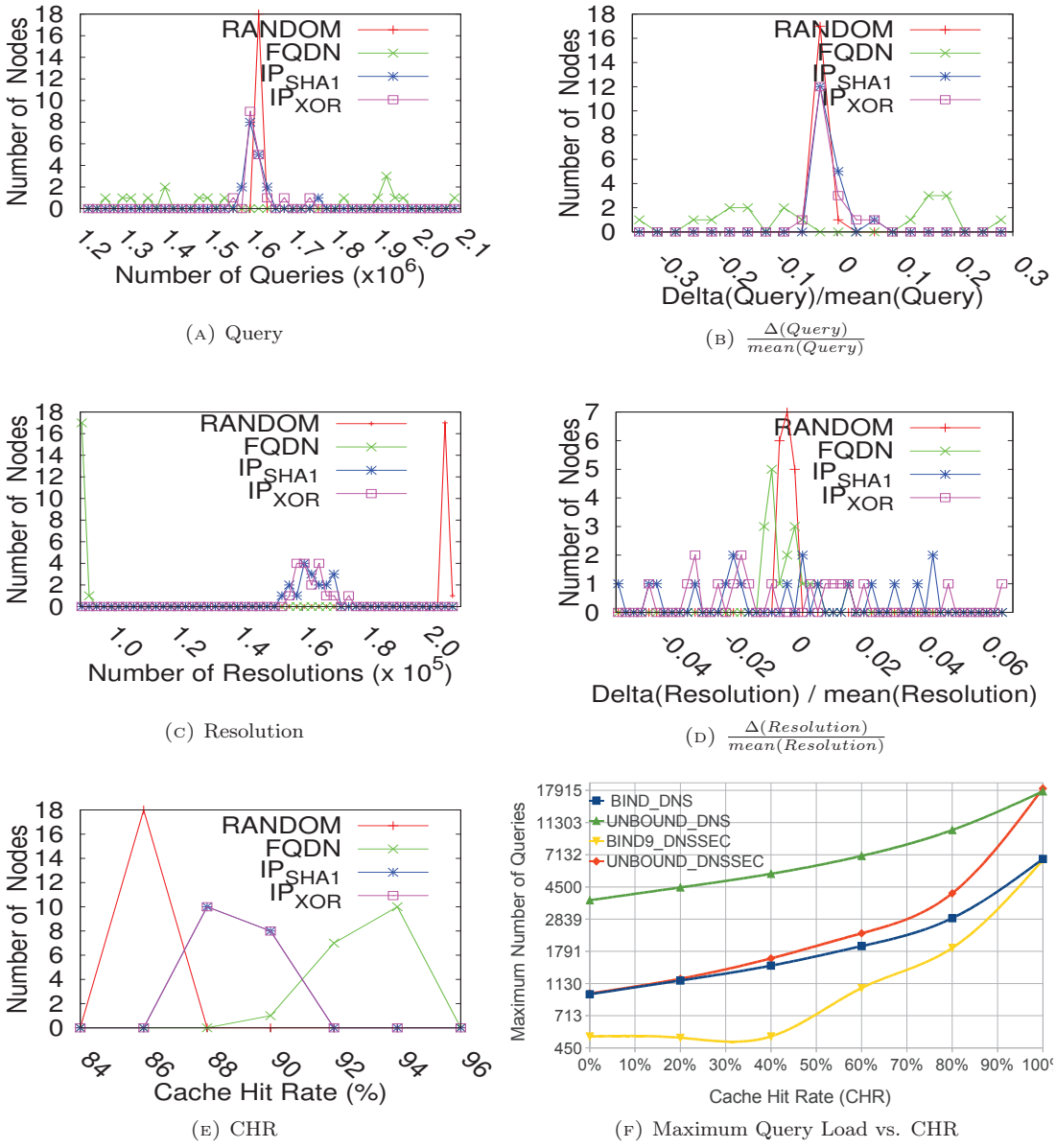
To compare the efficiency of different load balancing strategies from the CHR distribution associated to each strategy, we need a link between the CHR and the CPU. For that purpose we use an experimental benchmarking from [MGL10] with Intel Pentium III (@ 1GHz 32 bits) CPU, 384MB of RAM for servers with Debian 5.0 (lenny), Linux kernel 2.6.24. Node CPU is measured for traffic with various CHR. For clarity, experimental measurements provided by [MGL10] are replotted in figure 2.1f. It shows the maximum number of queries accepted by the server for a traffic with a given CHR. CHR influences the maximum number of accepted queries because, when a Cache Hit occurs, the node performs a cache lookup and only reads the data. On the other hand, when a Cache Miss occurs, an insertion is performed which requires to lock the cache. The advantage of this method is that it only relies on a platform benchmark (figure 2.1f) and a traffic analysis - the distribution of the CHR, which differs from a simulation.

Another way is to directly compute CPU for the different strategies with an estimation of CPU^H the occupancy time for a cache Hit and CPU^R, the occupancy time to perform a Resolution. CPU^H and CPU^R are derived from figure 2.1f, by considering specific values of $CHR = 0\%$ and $CHR = 100\%$.

Finally we compare the scalability by measuring the queries, the resolutions and the CHR associated to each node according to the number of nodes of the platform.

The studied load balancing strategies are based on IP_{XOR} and *FQDN* as mentioned in section 2.1. We also introduce two other strategies: IP_{SHA1} where splitting is done over the SHA1 value of both IP source and destination of the queries, and *Random* where the node selection is randomly performed. IP_{SHA1} vs IP_{XOR} shows the efficiency of SHA1 vs XOR functions.

2.4.1 Cache Hit Rate (CHR) Simulation



(A) Query

(B) $\frac{\Delta(Query)}{mean(Query)}$

(C) Resolution

(D) $\frac{\Delta(Resolution)}{mean(Resolution)}$

(E) CHR

(F) Maximum Query Load vs. CHR

FIGURE 2.1: Live Traffic Analysis: Query, Resolution CHR Distribution

Figure 2.1 represents the distributions of the Queries, (resp. Resolutions and CHR), that is to say, for a given number of Queries, (resp. Resolution of CHR value), the number of nodes that receive this amount of Queries (resp. perform this amount of Resolution, or have this CHR value). Figures 2.1a and 2.1b show the query distribution among the nodes of the platform. *Random* provides the best distribution, followed by IP_{SHA1} and IP_{XOR}, and then by *FQDN*. IP_{XOR} and IP_{SHA1} have almost the same distribution, thus validating this hashing function for load balancing

traffic. On the other hand, *FQDN* presents a huge dispersion of the queries, which tends to show that FQDNs do not provide enough randomness, i.e. some FQDNs unbalance the traffic.

On the other hand, Figures 2.1c and 2.1d show that *FQDN* distribution of resolution outperforms the other distributions, with a mean value of around 56% for IP_{XOR} or IP_{SHA1} and 44% for the random distribution. Furthermore the relative dispersion is much smaller with *FQDN* than with IP_{SHA1} or IP_{XOR} but *Random* still provides smaller dispersion.

For a DNS resolution platform, one distribution performs better if the dispersion of the queries balances the dispersion of the resolutions. With DNSSEC, unlike DNS, a resolution over the Internet requires signature validation. Thus CPU of the servers is largely influenced by the number of resolutions the node performs. Furthermore, figure 2.1e shows that *FQDN* has a mean CHR of 93% whereas IP_{XOR} and IP_{SHA1} provide 88% and *Random* 86%.

To estimate the platform's efficiency when raising CHR by 5% we use [MGL10] that runs an experimental platform and measures the Maximum Load for DNS(SEC) resolution platform versus CHR. Results are summed up in figure 2.1f which shows that with a CHR of 93% rather than 88%, the servers can deal with 1.49% more traffic with UNBOUND (resp. 1.25% with BIND9). As a result *FQDN* architecture is likely to reduce the number of nodes by up to 28%. In our case, our 18 node DNS platform running on BIND9 would become a 90 node platform with DNSSEC migration. Thus increasing the CHR by 5% with the *FQDN* architecture reduces the number of nodes of the DNSSEC platform by 24 nodes.

Note that this is a first estimation since [MGL10] considers a single signature check, and CHR is provided for a 18 node platform. In fact a resolution may involve more signature checks and 3 signatures may be more appropriated (ANSWER, AUTHORITATIVE and ADDITIONAL). On the other hand a DNSSEC platform is 5 times larger which decreases the CHR.

Table 2.2 reports the Maximum Load, i.e. the Maximum number of queries the resolving node

Architecture CHR	*Random* 86%	IP_{XOR} or IP_{SHA1} 88%	*FQDN* 93%
DNS - BIND9	3600 [1.11]	4000	5000 [0.80]
DNS - UNBOUND	12000 [1.06]	12700	14500 [0.87]
DNSSEC - BIND9	2730 [1.13]	3100	4250 [0.72]
DNSSEC - UNBOUND	6400 [1.16]	7450	10000 [0.75]

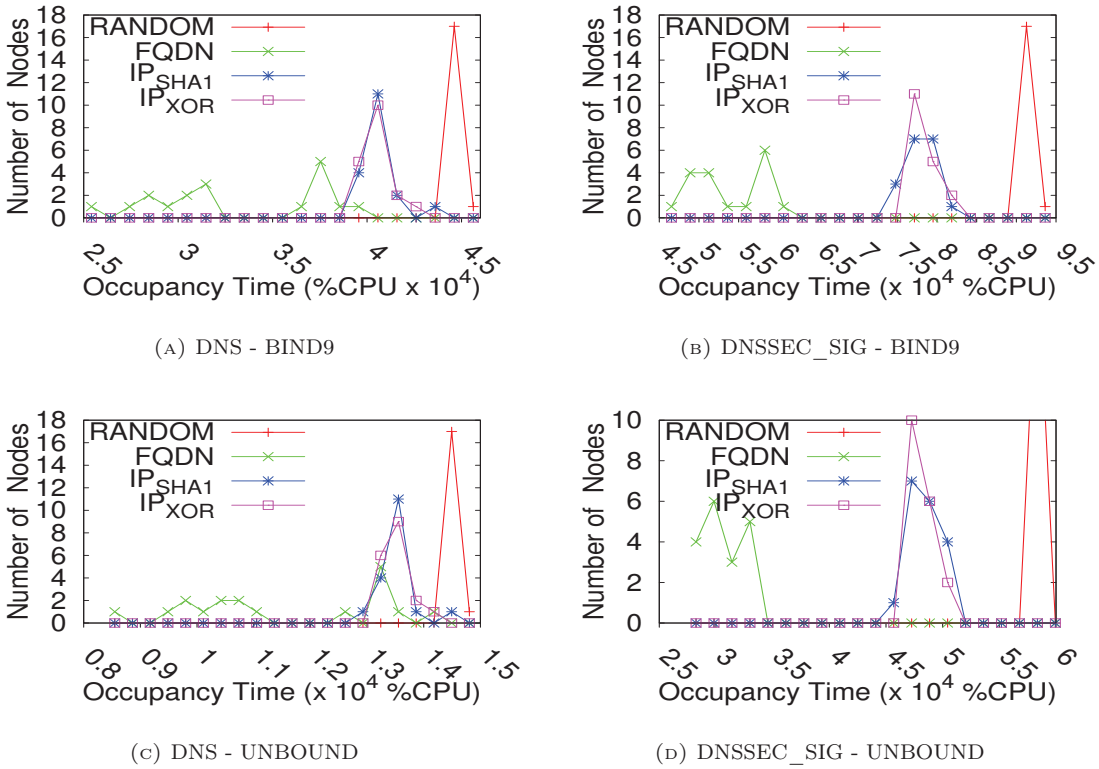
TABLE 2.2: Deriving for Measurements how a 5% Cache Hit Rate increase impact the Maximum Load of a DNS(SEC) Resolving Platform —($queries.s^1$) —and number of nodes ratio —$[\frac{n^{Arch}}{n^{XOR}} = \frac{ML^{XOR}}{ML^{Arch}}]$ —

2.4.2 CPU Time Simulation : CPU^R, CPU^H

This section estimates CPU by considering CPU^H and CPU^R (%CPU) for BIND9 (0.015 $\%CPU$, 0.317 $\%CPU$) and UNBOUND (0.005, 0.241). For both BIND9 and UNBOUND, IP_{XOR} and IP_{SHA1} are very similar, and perform better than *Random*. *FQDN*, on the other hand provides a bi-cluster distribution: the low CPU and high CPU groups. For DNS, the low CPU group has a high variance, and the mean CPU of the high CPU group almost equals CPU of IP_{SHA1} with UNBOUND. With BIND9, the mean CPU is even greater and almost equal to *Random*. With DNSSEC results are better, there are still two clusters but they have smaller variance, which means that the CPU is more uniformly distributed. The mean CPU value of both groups are closer to each other than in the case of DNS, and in any case much lower than with *Random* or IP_{SHA1} /

IP_{XOR}.

FQDN seems promising since it offers a mean CPU lower than any of the other architectures. As in section 2.4.1, *FQDN* happens to be 1.117 more efficient for BIND9 and DNS, respectively 1.172 for UNBOUND and DNS, 1.342 for BIND9 and DNSSEC and 1.409 for UNBOUND and DNSSEC. However its major drawback is that it presents a non uniform distribution.



(A) DNS - BIND9

(B) DNSSEC_SIG - BIND9

(C) DNS - UNBOUND

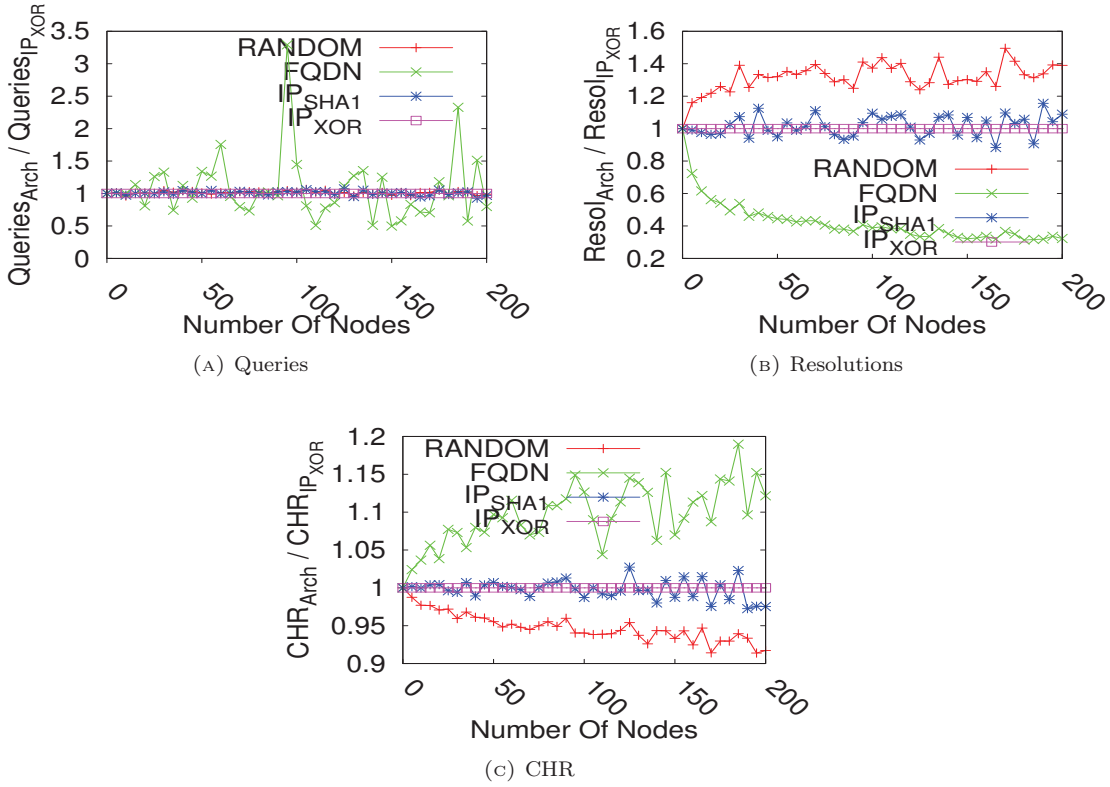
(D) DNSSEC_SIG - UNBOUND

FIGURE 2.2: Deriving CPU Time for Resolving Platform from real DNS traffic capture with various DNS(SEC) configurations and various implementations

2.4.3 Scalability Simulation

Figure 2.3 compares the number of queries, resolutions and CHR of $node_i$ versus n the number of nodes of the platform. It depicts the ratio with IP_{XOR}. For each n value, $node_i$ is randomly chosen, and the traffic is the same. For a 90 node DNSSEC platform, figure 2.3b shows that *Random* performs 30% more resolutions than IP_{XOR}, *FQDN* performs 61% less resolutions than IP_{XOR}, and IP_{XOR} requires only 4.8% more than IP_{SHA1}. Figure 2.3b shows that CHR of $node_i$ is 4.47% lower with *Random*, (resp. 1.09% IP_{XOR}), and larger by 8.79% with *FQDN*. Figures 2.3b and 2.3c also show that the difference between *FQDN* and IP_{SHA1} get larger when n increases, and that IP_{XOR} provides results very close to IP_{SHA1}. Figure 2.3a shows that the distribution of queries is not optimized with *FQDN*. The peak for $n = 95$ and $n = 185$ probably results from the fact that $node_i$ is *Home* for a range of popular FQDNs.

As a result, figure 2.3 shows that *FQDN* provides a much scalable architecture than IP_{SHA1}. However, queries are better load balanced by considering the IP addresses.



(A) Queries

(B) Resolutions

(C) CHR

FIGURE 2.3: Scalability vs. Number of Nodes —Queries, Resolutions and Cache Hit Rate (CHR) ratio are expressed as a ratio with the IP_{XOR} architecture.

2.4.4 Conclusion on Load Balancer

FQDN improves the platform's efficiency by roughly 30% compared to IP_{SHA1}. However, *FQDN* does not present a uniform distribution of the CPU among the nodes. This mostly results from the difference between FQDN's popularity which is much higher than user's query ability. In that sense, *FQDN* distribution can not rely on a hash function (SHA1) and we have to check that two popular FQDNs cannot be assigned to the same *Home*. Establishing such a distribution is out of scope of the chapter, however we have several solutions in mind. For instance, for a given hash function, we can use a salt, concatenate the salt to the FQDN to be hashed and consider $hash(FQDN||salt)$. The salt is incremented until it splits properly the most popular FQDNs. Another alternative is to build a routing table for the most requested FQDNs [XMSF11, FMS11, FMS12] whereas others are load balanced with the hash function. With Mixed Integer Programming (MIP) and K-mean techniques, one computes an almost uniform distribution where each node deals with the same amount of queries and resolutions. In the next sections, we assume such a distribution has been established and check how *FQDN* principles can be deployed through Pastry based architectures. When building the routing table, there is a balance between the number of FQDN to consider, the algorithm to define the most efficient routing table, and the resources required to compute it. [XMSF11] provides a good problem description as well as guide lines on how to compute the routing table by using MIP or K-mean techniques. The computed distribution is such that each node has to deal with the same number of queries and the same number of resolution.

Current work consists on finding a more efficient algorithm to build the routing table.

2.5 Modelization of Pastry based Architectures

This section computes IP_{SHA1}, *FQDN* and various Pastry based architectures. We refer to Pastry as its architecture is widely known by the community, but other DHT protocol may be considered. The way we use Pastry [RD01a] differs from what it has originally been designed for. First of all, the Pastry nodes constitute the platform. As such they belong to the same administrator, are located in the data center, every node knows all the other nodes. This is plausible since platform do not expect to be larger than a few hundred nodes. Then, another major difference is that we do not assert that the node ID is always derived from the data by a simple hash (SHA1) function, but we may apply a specific distribution known by all nodes. We used Pastry because we want to take advantage of the auto configuration mechanisms, but we do not consider the routing algorithm. As such, a light Pastry may be developed. Another advantage of Pastry (or DHT) is the robustness to DoS attacks [Mas06, RHM09, CMM02], which may balance the sensitivity of DoS attacks introduced by DNSSEC. In fact, resolution in DNSSEC involves signature checks, and thus costs much more than DNS resolutions. Then NSEC3, and the proof of non-existence, adds another hash to be performed for both authoritative and resolving servers, and requires hosted hashed data to be ordered, which makes operation heavier - especially for authoritative servers.

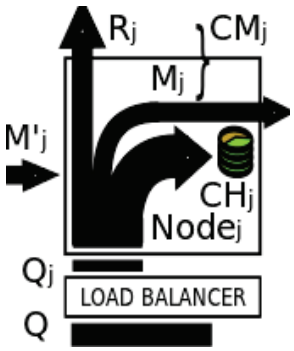


FIGURE 2.4: Traffic Description going through a Node of the DHT Platform n_j

From figure 2.4, we note the following traffic flows:

- Q: the number of queries sent by all end users, and we note q the query rate —queries per second $q.s^{-}1$.
- Q_j: traffic received by node n_j ($q_j = L_j.q$).
- I_j: DNS resolutions performed by n_j over the Internet.
- $M_j^{'}$ (resp. M_j) : the incoming (resp. outgoing) exchanges between n_j and the other nodes of the platform.

The different objects we consider are:

- $p(type, IPs, fqdn, t)$: a DNS packet where *type* indicates if p is a query or a response, IPs the IP_{source}, $IP_{destination}$, $fqdn$ the FQDN and t the time at which p is sent.
- $fqdn(name, ttl, rank)$: a FQDN where *name* designates the FQDN, *ttl* its associated Time-To-Live (TTL) and *rank* its rank which reflects its popularity. Note that *ranks* is based on the popularity of the FQDN, i.e. the query rate value associated to that FQDN. However each FQDN has a distinct rank, which means that when two FQDNs have the same associated request rate, their associated *ranks* are r and $r + 1$.

The following architecture components are:

- lb : a Load Balancer that splits the traffic among the different nodes of the platform.
- n_j : $node_j$ of the n node platform.

We note :

- $\mathcal{F} = \{f_r \,|\, \exists p \in Q,\, p.fqdn = f_r,\, r \in \mathcal{R} = card(\mathcal{F})\}$. $\mathcal{F}$ is the ordered list of all FQDNs concerned by the queries of the End Users Q. FQDNs are ranked according to their popularities.
- $\mathcal{P} = \{IPs_i \,|\, \exists p \in Q, p.IPs = IPs_i\}$. $\mathcal{P}$ is the list of IP addresses involved in the traffic between the End Users and the Platform.
- $H(f_{r,r\in\mathcal{R}}) = \{n_j \,|\, n_j \text{ is } Home \text{ for } f_r,\, j \in [1, ..., n]\}$ defines the set of the nodes that perform resolution for FQDN f_r.
- $\mathcal{C}_j$ is the cache of n_j.
- CPU, RT defines the CPU Time and the Response Time (RT). RT depends on CPU, and CPU depends on the action performed (cache lookup CPU^H, cache insertion CPU^R, query forwarding CPU^{FWD}...) and the considered protocol (DNS, DNSSEC).
- l_I, l_{ISP} and l_{plt} represents the network latency on the ISP's network, over the Internet and on the platform.

The different probabilities we are considering for each node are :

- $L_j = P(p \in Q_j | p \in Q)$: the probability that n_j receives a packet p of Q.
- $H_{r,j} = P(n_j \in H(f_r))$: the probability that n_j is Home for f_r.
- $\Phi(r) = P(fqdn = f_r, r \in \mathcal{R})$.
- $CM = P(p.fqdn \notin \bigcup_{j=0}^{n-1} \mathcal{C}_j \cap p \in Q)$ the platform probability for a Cache Miss, i.e. the response is not stored in any cache node $\mathcal{C}_j$ of the platform. Respectively $CH = P(p.fqdn \in \bigcup_{j=0}^{n-1} \mathcal{C}_j | \cap \in Q)$ designates the Cache Hit of the platform.
- $CM_j = P(p.fqdn \notin \mathcal{C}_j \cap p \in Q_j)$: the probability a packet addressed to n_j has not its response in $\mathcal{C}_j$. Respectively $CH_j = P(p.fqdn \in \mathcal{C}_j \cap p \in Q_j)$ represents the Cache Hit of the node.
- R_j :the probability a packet p in Q_j triggers a resolution over the Internet on n_j.
- M_j : the probability a packet p in Q_j triggers a management operation by n_j (like being forwarded to another node).

Note that we have $CM + CH = 1$, $CM_j + CH_j = 1$ and $CM_j = M_j + R_j$, so an architecture is fully characterized by CM, R_j and M_j.

2.5.1 Single Node Model: τ_r

Let us consider the case where a DNS traffic Q is addressed to a single node.

$$\begin{aligned} Q &= \{p(query, IPs, fqdn),\, fqdn \in \mathcal{F}, IPs \in \mathcal{P}\} \\ CM_o &= P(p.fqdn \notin \mathcal{C} | p \in Q) \end{aligned}$$

With the following assumptions :

$$\begin{cases} (p.fqdn \in \mathcal{F}) \Leftrightarrow (\exists r \in \mathcal{R}, p.fqdn = fqdn_r), \\ P(fqdn = fqdn_r \cap fqdn = fqdn_s, \forall r, s \in \mathcal{R}^2, r \neq s) = 0 \end{cases}$$

$$\begin{aligned} CM_o &= P(p.fqdn \notin \mathcal{C} | p.fqdn \in \mathcal{F}) \\ &= \sum_{r \in \mathcal{R}} P(p.fqdn \notin \mathcal{C} | p.fqdn = fqdn_r) \\ &= \sum_{r \in \mathcal{R}} P(fqdn_r \notin \mathcal{C}).P(p.fqdn = fqdn_r) \\ &= \sum_{r \in \mathcal{R}} cm_{r,p,t}.\Phi(r) \\ &\quad \text{with } cm_{r,p,t} = P(p.fqdn_r \notin \mathcal{C}) \end{aligned}$$

Under a constant query rate q, we want to express $q\tau_r$ the number of packets to consider between t and $t - p.fqdn_r.ttl$. τ depends on the $fqdn_r.ttl$, but if we want to rely on our measurements, especially for $\Phi(r)$, we have to consider that measurements have been performed during T seconds. In this simplified model, we considered as in [JSBM01] a constant TTL for the FQDNs. When a packet arrives at t, to test if the request is in the cache, we need to consider the previous $q \times fqdn_r.ttl$ if $t > fqdn_r.ttl$ or $q \times t$ if $t < fqdn_r.ttl$.

Let $\mathcal{P}'_p$ be the set of packets we have to consider to express the Cache Miss for a given packet. Typically, that's the packets that came between t the time we receive p and $t - p.fqdn.ttl$.

$$\mathcal{P}'_p = \{p' | p' \in \mathcal{P} \text{ and } p' .t < t - p.fqdn.ttl\}$$

Assuming all queries are independent, we have :

$$\begin{aligned} cm_{r,p,t} &= P(p'.fqdn \neq fqdn_r, \forall p' \in \mathcal{P}'_p) \\ &= (1 - \Phi(r))^{card(\mathcal{P}'_{p,r})} \end{aligned}$$

Let assume that we have a constant query rate q. We thus can derive :

$$card(\mathcal{P}'_{p,r}) = q.min(p.t, fqdn_r.ttl)$$

$$card(\mathcal{P}'_{p,r}) = \begin{cases} q.fqdn_r.ttl, \text{ when } t > fqdn_r.ttl \\ q.t, \text{ when } t < fqdn_r.ttl \end{cases}$$

Considering τ_r the associated frequency of $fqdn_r$, $\tau_r = \frac{1}{q}\langle card(\mathcal{P}'_{p,r})\rangle$. Then if we derive the mean value over time, we have :

$$\begin{cases} \tau_r = \frac{1}{T}\left[\frac{1}{2}.ttl_r^2 + ttl_r(T - ttl_r)\right], \text{ when } fqdn_r.ttl < T \\ \tau_r = \frac{T}{2}, \text{ when } t < fqdn_r.ttl \end{cases}$$

Then,

$$\langle cm_{r,p,t}\rangle = (1 - \Phi(r))^{q.\tau_r}$$

And Finally :

$$\begin{aligned} CM_o &= \sum_{r \in \mathcal{R}} (1 - \Phi(r))^{q.\tau_r}.\Phi(r) \\ R_o &= CM \\ M_o &= 0 \end{aligned} \tag{2.2}$$

We then derive :

$$\begin{aligned} CPU(q) &= q.CPU^H + q.CM.CPU^R \\ RT(q) &= l_{ISP} + CH.RT^H(_{CPU_j}) + \\ &\quad CM_{traf.}.(RT^R(_{CPU_j}) + l_I) \end{aligned} \tag{2.3}$$

2.5.2 IP_{SHA1} Architecture

In the IP_{SHA1} architecture traffic is balanced, as represented in figure 2.5a, according to $p.IPs$. If we assume that FQDN and IP addresses are independent, then :

$$P(p.fqdn = p.fqdn_r | p \in Q_j) = \frac{P((p.fqdn = p.fqdn_r | p \in Q).P(p \in Q_j)}{P(p \in Q_j)}$$

Then,

$$\begin{aligned} CM_j &= P(p.fqdn \notin \mathcal{C}_j | p \in Q_j) \\ &= \sum_{r \in \mathcal{R}} P(p.fqdn = fqdn_r \cap p.fqdn \notin \mathcal{C}_j | p \in Q_j) \end{aligned}$$

From equation 2.2 considering that $p.fqdn = fqdn_r$ and $p \in Q_j$ are independent :

$$\begin{aligned} CM &= P(p.fqdn \notin \bigcup_{j=0}^{n-1} \mathcal{C}_j | p \in Q) \\ &= \sum_{j=0}^{n-1} P(p.fqdn \notin \mathcal{C}_j | p \in Q_j) \\ &= \sum_{j=0}^{n-1} L_j.CM_j \\ &= CM_j \\ CM_j &= \sum_{r \in \mathcal{R}} \Phi(r).(1 - \Phi(r))^{L_j.q\tau_r} \\ R_j &= CM_j \\ M_j &= 0 \end{aligned} \tag{2.4}$$

Finally :

$$\begin{aligned} CPU(q_j) &= q_j.CPU^H + q_j.CM_j.CPU^R \\ RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + \\ &\quad CM_j.(RT^R(_{CPU_j}) + l_I) \end{aligned} \tag{2.5}$$

2.5.3 *FQDN* Architecture

In *FQDN*, each FQDN is *Homed* by one specific node, which performs all resolution over the Internet for that FQDN. In this section we assume that load balancers redirect each $fqdn_r$ to its *Home*, as presented in figure 2.5a. How FQDN are assigned to the nodes is out of scope of this section. However, we show in [XMSF11] that using MIP or K-mean techniques results in a distribution such that each node has to deal with the same number of queries and the same number of resolution.

We define $\mathcal{F}_j$ the set of FQDN n_j is *Home* for and $\mathcal{R}_j$ their associated ranks.

$$\begin{aligned} \mathcal{F}_j &= \{fqdn | H(fqdn) = n_j, fqdn \in \mathcal{F}\} \\ \mathcal{R}_j &= \{r | fqdn_r \in \mathcal{F}_j, i \in \mathcal{R}\} \end{aligned}$$

General partition properties are :

$$\begin{cases} \mathcal{F} = \bigcup_{j=0}^{n} \mathcal{F}_j \\ \mathcal{F}_i \cap \mathcal{F}_j = \emptyset \, \forall (i,j) \in [0,...,n-1]^2, i \neq j \end{cases}$$

The specificities of our partition are :

$$\begin{cases} \text{Each node } n_j \text{ receives the same amount of queries.} \\ \text{Each node } n_j \text{ performs the same number of resolutions.} \\ \text{Each node } n_j \text{ is } \textit{home} \text{ of the same number of FQDN.} \end{cases}$$

Since $\{\mathcal{F}\}_j$ is a partition, the contribution of $fqdn \in \mathcal{F}_j$ to CM_j/CH_j is the same as for CM_o/CH_o. Using the distribution's properties, we have $(CH_j, CM_j) = (CH_i, CM_i), \forall (i,j) \in [0,...,n-1]^2$. Then CH_o (resp. CM_o) is the mean of the CH_j (resp. CM_j) and finally $(CH_j, CM_j) = (CH_o, CM_o) \, \forall j \in [0,...,n-1]$.

$$\begin{aligned} CM &= CM_o \\ CM_j &= CM_o \\ R_j &= CM_j \\ M_j &= 0 \end{aligned} \tag{2.6}$$

Finally :

$$\begin{aligned} CPU(q_j) &= q_j.CPU^H + q_j.CM.CPU^R \\ RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + \\ &\quad CM_j.(CPU^M(_{CPU_j}) + l_I) \end{aligned} \tag{2.7}$$

Here follows demonstration for :

$$(CH_j, CM_j) = (CH_{traf.}, CM_{traf.}) \, \forall j \in [0,...,n-1]$$

$$\begin{aligned} CH_{traf.} &= P(p.fqdn \in \mathcal{C} | p \in Q) \\ &= \sum_{j=0}^{n-1} P(p.fqdn \in \mathcal{C} | p \in Q_j).P(p \in Q_j) \\ &= \sum_{j=0}^{n-1} P(p.fqdn \in \mathcal{C}_j | p \in Q_j).P(p \in Q_j) \\ &= \sum_{j=0}^{n-1} L_j.CH_j \end{aligned}$$

With the distribution we consider,
$L_j = L$ and $CH_j = CH_{node} \, \forall j \in [0, \ldots, (n-1)]$

Then :

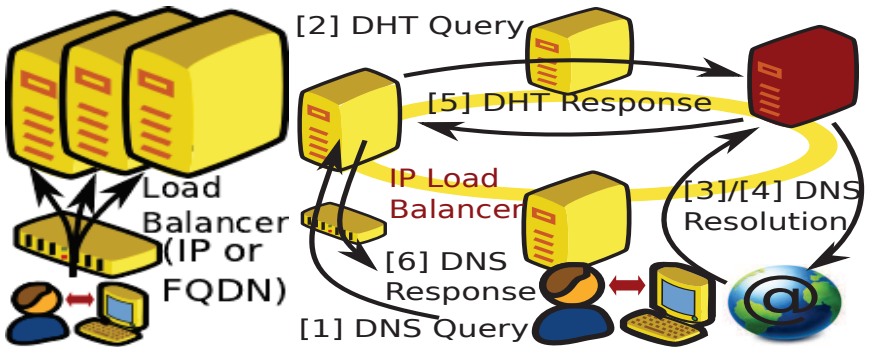
$$\begin{aligned} CH &= \sum_{j=0}^{n-1} L.CH_{node} \\ CH &= CH_{node} \end{aligned}$$

2.5.4 *Pastry*-based architecture (no cache, no replication)

Pastry-based architectures are a combination of the *FQDN* (section 2.5.3) and the IP_{SHA1} (section 2.5.2) architecture. Load balancing functions according to the IPs is combined with the use of *Home* nodes for each FQDN. The main goal of such architectures is to deploy a set of nodes that are self-organizing without modifying the current architecture – the heavy loaded load balancer architecture – while minimizing the numbers of resolutions performed by the platform. With *Pastry* based solutions, all the intelligence is placed on the platform's nodes. All nodes route queries to their *Home* node and resolve the queries they are *Home* for. The challenge is to find out how managing the traffic balances the number of avoided DNS(SEC) resolutions. There are various ways to manage the incoming traffic. In this section, we consider the following mechanisms :

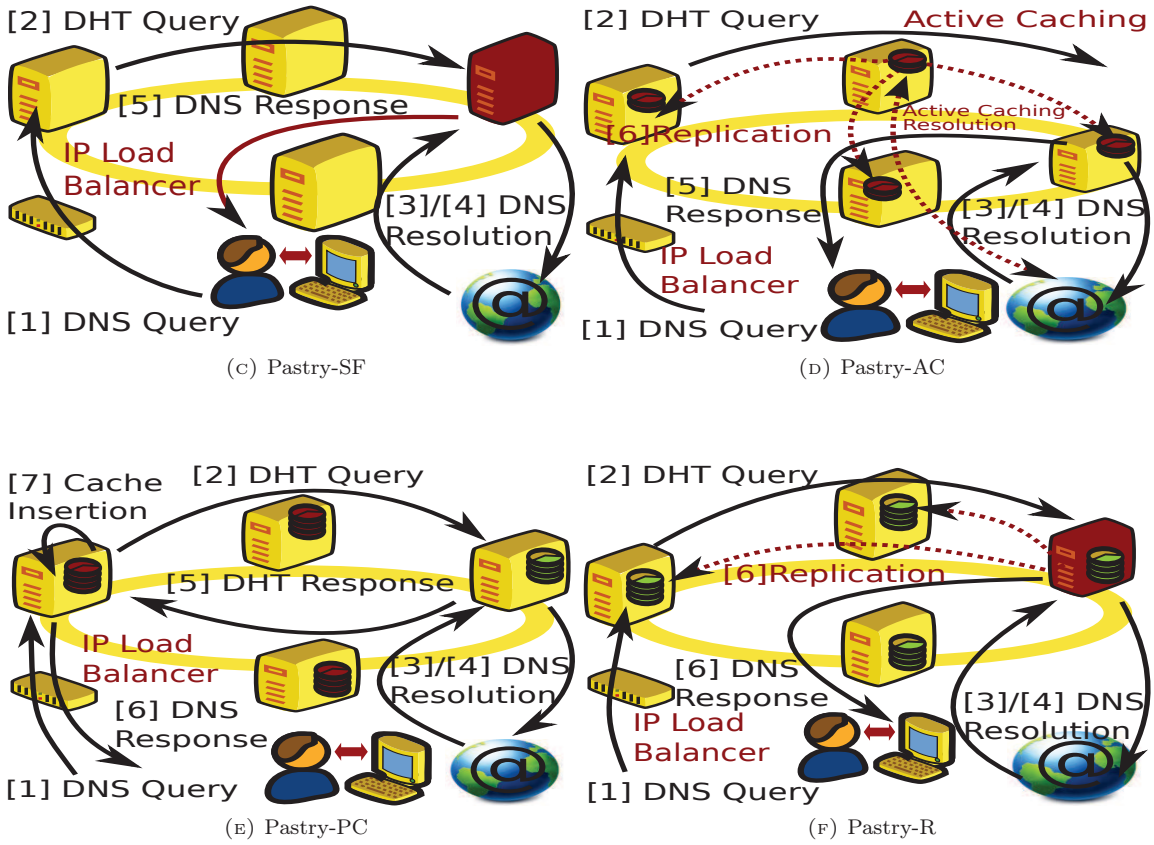
- No cache no Replication (*Pastry*) : When a node receives a query from an end user and doesn't have the response in its cache, then its sends a query to the *Home* node, and then forwards the response to the end user.
- Stateless Forwarding (*Pastry-SF*) : When the *Home* node receives a query from another *Pastry* node, it sends directly the response to the end user, rather than to the *Pastry* node.
- Passive Caching (*Pastry-PC*) : works like No mechanism except that the querying node keep the response in its cache.
- Replication (*Pastry-R*) : When the *Home* node performs a resolution, it provides a copy of the response to k neighbors. This mechanism is mainly used for robustness in case a node fails.
- Active Caching (*Pastry-AC*) : takes advantage of the law power distributions. In our case, it means that a *Home* node choose to update the other node's caches with its γ most popular FQDN. It takes advantage that a small portion of FQDN provides response to a large part of the traffic.

Pastry, *Pastry-SF*, *Pastry-PC*, *Pastry-R* and *Pastry-AC* are respectively illustrated by figures 2.5a, 2.5b, 2.4c, 2.4d, 2.3e and 2.3f.



(A) IP_{SHA1}: *FQDN*

(B) *Pastry*



(c) Pastry-SF (d) Pastry-AC

(e) Pastry-PC (f) Pastry-R

FIGURE 2.3: Description of DHT Architecture and their Principles

$$\begin{aligned} CM &= CM_o \\ R_j &= H_{r,j} \sum_{r \in \mathcal{R}} \Phi(r).(1-\Phi(r))^{q\tau_r} \\ M_j &= \sum_{r \in \mathcal{R}} (1 - H_{r,j})\Phi(r) \end{aligned} \tag{2.8}$$

Finally :

$$\begin{aligned} CPU(q_j) &= q_j.(CH_j + M_j.CH).CPU^H + \\ &\quad q_j.M_j.CPU^{FWD} + \\ &\quad q_j.(R_j + M_j.CM).CPU^R \\ RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + \\ &\quad R_j.(RT^M(_{CPU_j}) + l_I) + \\ &\quad M_j.(l_{plt} + CH_{traf.}.RT^H(_{CPU_j}) + \\ &\qquad CM_{traf.}.(RT^M(_{CPU_j}) + l_I)) \end{aligned} \tag{2.9}$$

$CPU^{FWD} \approx CPU^H$ since we consider a routing table lookup on a small table, forwarding the query to the *Home* node and then forwarding the response to the end user. CPU^H considers reading the DNS query, but this might not be necessary, and redirection may be based on reading hashing a fixed number of bits at a defined position, as routers do with IP addresses.

2.5.5 *Pastry-Stateless Forwarding* (*Pastry-SF*): (no cache, no replication)

With SF-Pastry n_j sends the response directly to the end user rather than to the Pastry node that has forwarded the query. This means that n_j does not have to handle response of packets from M_j. CM_j, R_j and M_j probabilities are the same as in equation 2.8. Finally:

$$\begin{aligned} CPU(q_j) &= q_j.(CH_j + M_j.CH).CPU^H + \\ &\quad q_j.M_j.CPU^{FWD'} + \\ &\quad q_j.(R_j + M_j.CM).CPU^R \\ RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + \\ &\quad R_j.(RT^M(_{CPU_j}) + l_I) + \\ &\quad M_j.(l_{plt}^{query} + CH_{traf.}RT^H(_{CPU_j}) + \\ &\qquad CM_{traf.}(RT^M(_{CPU_j}) + l_I)) \end{aligned} \tag{2.10}$$

$CPU^{FWD'} \approx \frac{CPU^H}{2}$. $CPU^{FWD'}$ is similar to CPU^{FWD} to the extent that no response is received and responses are larger than queries.

2.5.6 *Pastry-Active Caching* (*Pastry-AC*) (no replication)

Active Caching [RS04] takes advantage of the Zipf distribution of the FQDNs. Each node informs all other nodes of the γ most popular *Homed* FQDN. In our model, nodes responsible for the γ FQDNs are proactive, which means that no cache miss occurs for those FQDNs. If we consider that all FQDN have the same TTL (as in [JSBM01]), then during TTL n_j sends γ responses to all $n-1$ nodes and receives the γ most popular FQDN from all $n-1$ nodes. As a consequence a DNS query with rank greater than $n.\gamma$ follows the *Stateless Forwarding* Pastry resolution procedure.

$$\begin{aligned} CM &= \sum_{r\in\mathcal{R}} P((p.fqdn = fqdn_r,\, r > n.\gamma) \cap \\ &\qquad p.fqdn \notin \bigcup_{j=0}^{n-1} \mathcal{C}_j | p \in Q) \\ &= \sum_{r\in\mathcal{R}}^{r>n.\gamma} \Phi(r)(1-\Phi(r))^{q.\tau_r} \\ R_j &= \sum_{r>n.\gamma} H_{r,j}\Phi(r)(1-\Phi(r))^{q.\tau_r} \\ M_j &= \sum_{r>n.\gamma} (1-H_{r,j})\Phi(r) \end{aligned} \tag{2.11}$$

Finally:

$$\begin{aligned} CPU(q_j) &= q_j.(CM_j + M_j.CH)CPU^H + \\ &\quad q_j.(R_j + M_j.CM).CPU^R + \\ &\quad q_j.M_j.CPU^{FWD'} + \\ &\quad \frac{\gamma}{\tau_r}.CPU^{R_{AC}} + \\ &\quad CPU^{FWD_{AC}}(_{2.\frac{\gamma.(n-1)}{\tau_r}}) \\ RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + R_j.RT^M(_{CPU_j}) + \\ &\quad M_j.(l_{plt} + CH.RT^H(_{CPU_j}) + \\ &\qquad CM.(RT^M(_{CPU_j}) + l_I)) \end{aligned} \tag{2.12}$$

$CPU_j^{FWD'} \approx \frac{CPU^H}{2}$ as in equation 2.10. $CPU_j^{R_{AC}} \approx CPU^H$. With $CPU^{R_{DNS}}$ the CPU of a DNS resolution (as opposed to DNSSEC), $CPU_j^{FWD_{AC}} \approx n.CPU_{DNS}^R$, since cache updates are sent in one block, and the updated Active Cache are small. Optimization may also consider different levels of caches, so to improve cache lookup and interactions between $\mathcal{C}_j$ and the *Active Cache*.

2.5.7 *Pastry-Passive Caching* (*Pastry-PC*) : (cache, no replication)

With Passive Caching, node proceeds as in the regular Pastry, but keeps the response in its cache. Thus it can respond directly to the end user, even though it is not a *Home* for that FQDN. Note that with passive caching, if FQDN is in the cache of one of the other nodes of the platform, then it is in the cache of its *Home* node.

$$\begin{aligned} CM &= P(p.fqdn \notin \bigcup_{j=0}^{n-1} \mathcal{C}_j | p \in Q) \\ &= CM_{traf.} \\ R_j &= \sum_{r \in \mathcal{R}} P(p.fqdn = f_r \cap H(f_r) = n_j \cap f_r \notin \mathcal{C}_j) \\ &\quad \sum_{r \in \mathcal{R}} \Phi(r).H_{r,j}.(1-\Phi(r))^{q.\tau_r} \\ M_j &= \sum_{r \in \mathcal{R}} P(p.fqdn = f_r \cap H(f_r) \neq n_j \cap f_r \notin \mathcal{C}_j) \\ &\quad \sum_{r \in \mathcal{R}} (1-H_{r,j}).\Phi(r).(1-\Phi(r))^{q.L_j.\tau_r} \end{aligned} \tag{2.13}$$

Finally :

$$\begin{aligned} CPU(q) &= q_j.(R_j + M_j.CM).CPU^R + \\ &\quad q_j.(CH_j + M_j.CH).CPU^H + \\ &\quad q_j.M_j.CPU^{FWD''} \\ RT(q) &= l_{ISP} + R_j.RT^M(_{CPU_j}) + CH_j.RT^H(_{CPU_j}) + \\ &\quad M_j.(l_{plt} + CH.RT^H(_{CPU_j}) + \\ &\qquad CM.(RT^M(_{CPU_j}) + l_I)) \end{aligned} \tag{2.14}$$

$CPU^{FWD''} \approx CPU^{R}_{DNS}$ since insertion is performed on a large cache.

2.5.8 *Pastry-Replication* (*Pastry-R*): (no cache, replication)

Replication with no cache works like the regular Pastry architecture with no cache except that when a resolution is performed the response is replicated on k neighbors. This mechanism is close to the active caching mechanism to the extent that all FQDNs will be replicated, whereas in active caching we only care about the most popular FQDNs, then replication occurs only to k nodes, whereas Active Caching replicates the responses on all nodes. When n_j does not have the response in its cache, it proceeds as in Pastry (cf. section 2.5.4). Let $\mathcal{N}^k_j = \{l \,|\, n_l \text{ is neighbor of } n_j,\ l \neq j\}$.

$$\begin{aligned}
CM &= CM_o \\
R_j &= \sum_{r\in\mathcal{R}} P(p.fqdn = f_r \cap H(f_r) = n_j \cap f_r \notin \mathcal{C}_j) \\
&= \sum_{r\in\mathcal{R}} \Phi(r).H_{r,j}.(1-\Phi(r))^{q.\tau_r} \\
M_j &= \sum_{r\in\mathcal{R}} P((p.fqdn = f_r \cap \\
&\qquad H(f_r) \in \mathcal{N}^k_j \cap f_r \notin \mathcal{C}_j) \cup \\
&\qquad (p.fqdn = f_r \cap H(f_r) \notin \{\mathcal{N}^k_j, n_j\})) \\
&= \sum_{r\in\mathcal{R}} \Phi(r).k.H_{r,j}.(1-\Phi(r))^{q.\tau_r} + \\
&\qquad \Phi(r).(1-(k+1).H_{r,j})
\end{aligned} \tag{2.15}$$

$$\begin{aligned}
PC^{FWD} &= P(\text{emitting or receiving a FQDN from peer}) \\
&= 2kR_j
\end{aligned} \tag{2.16}$$

Finally :

$$\begin{aligned}
CPU(q_j) &= q_j.(CH_j + M_j.CH).CPU^H + \\
&\quad q_j.(R_j + M_j.CM).CPU^R + \\
&\quad q_j.M_j.CPU^{FWD'} + \\
&\quad q_j.2.k.R_j.CPU^{FWD_{PC}} \\
RT(q_j) &= l_{ISP} + CH_j.RT^H(_{CPU_j}) + \\
&\quad R_j.(RT^M(_{CPU_j}) + l_I) + \\
&\quad M_j.(l_{plt} + CM.(RT^M(_{CPU_j}) + l_I) + \\
&\qquad CH.RT^H(_{CPU_j}))
\end{aligned} \tag{2.17}$$

$CPU^{FWD'}_j \approx \frac{CPU^H}{2}$ (cf. equation 2.10). $CPU^{FWD_{PC}}_j \approx \frac{1}{2}CPU^R_{DNS}$ since cache update is performed, but no response is sent nor cache lookup performed.

2.6 Configuration for Simulation

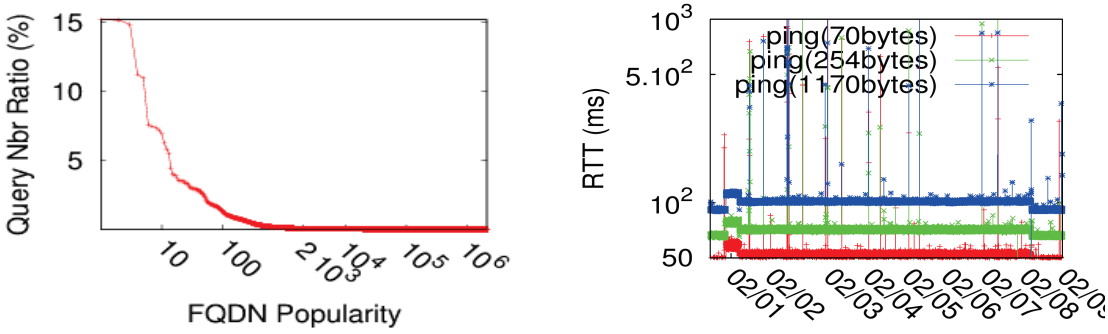
Numerical application of the models requires that we define $\Phi(r)$, the experimental popularity distribution, the occupancy time CPU required for a Cache Hit (CPU^H) and for a resolution

(CPU^R). The values considered for CPU^R and CPU^H are derived from figure 2.1f, and thus only consider small cache size. This is a limitation of our model, and results provided for large cache should be carefully considered (*Pastry-AC* with large γ, *Pastry-PC*, and *Pastry-R*). Response Time needs to evaluate the latency to reach the ISP DNS platform (l_ISP), the latency over the Internet (l_I) and between the nodes of the platform.

Figure 2.4a shows the distribution of the FQDN popularity Φ computed from a 10 minute traffic of $35,105,176$ queries and $2,063,864$ FQDNs. Note that we did not consider the Zipf modelization and instead considered the measured popularity distribution function so to avoid errors due to traffic estimations. Figure 2.4b and table 2.3 show latency measurements within Orange network with different size of packet to compute DNS queries and DNS/DNSSEC responses. Tests are also summed up in table 2.3 for tests performed every 30 sec from January 31 11 : 16 pm to February 9 23 : 33. Daily peaks are due to the rekeying of the PPP session. We derive latency values $l_{ISP}^{DNS} = 64\,ms$ and $l_{ISP}^{DNSSEC} = 79\,ms$. We also measured $l_I = 223\,ms$ for the Internet and on the platform $l_{plt}^{query} = 0.4\,ms$ and $l_{plt}^{response} = 0.6\,ms$.

RTT (ms)	Bytes	Min	Average	Max	Mdev
DNS - Query	98	46.028	54.676	2810.981	67.424
DNS - Response	282	64.063	73.335	2849.930	67.795
DNSSEC - Response	1198	89.159	103.391	2906.601	69.171

TABLE 2.3: Maximum Load versus Cache Hit Rate



(A) Popularity (B) Ping

FIGURE 2.4: Traffic and Network Measured Characteristics: FQDN Popularity Distribution and Network Latency

2.7 Simulation of Pastry Based Architectures

This section compares the different architectures modeled in section 2.5. Simulations are run with the data of section 2.6 of our 18 node DNS platform and the experimental measured ranking of FQDN versus their popularity ($\Phi(r)$) of our 10 minute traffic live capture. We consider that DNSSEC requires 90 nodes. This section also compares each architecture' sensibility to criteria, as the addition of nodes, the variation of the ratio between the cost of a resolution and cache hit $\frac{CPU^R}{CPU^H}$ and the variation of the TTL. Platform's efficiency is measured by the node CPU, and figures plot for each architecture - IP_{SHA1}, *FQDN*, *Pastry*, *Pastry-SF*, *Pastry-AC*, *Pastry-PC* and *Pastry-R* - the CPU ratio $\frac{CPU^{Arch}}{CPU^{IPs}}$. We considered IP_{SHA1} instead of IP_{XOR} as in section 2.4

because IP_{XOR} has not been modeled, and IP_{XOR}'s efficiency almost equals IP_{SHA1}' efficiency (cf. section 2.4).

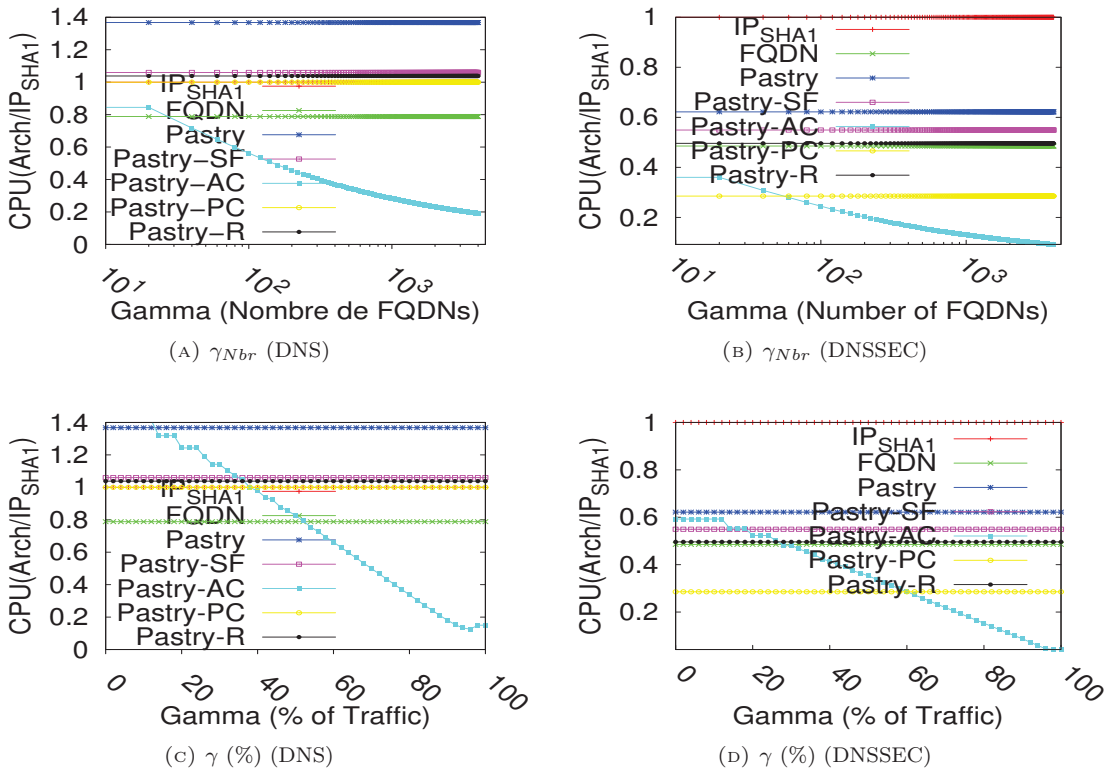
2.7.1 γ, *Neighbors*

Pastry-AC and *Pastry-R* can have different values for γ the number of FQDNs actively cached or k the number of neighbors the node replicates the responses on. This section shows how they influence the node's performance and give estimation for them.

Figure 2.5 shows for DNS and DNSSEC how γ influences the performances. γ is expressed both as the number of FQDN the node is considering and as well as the percentage of the global DNS traffic. Correspondence between traffic percentage and Number of FQDN is provided by figure 2.2b. Note that, that as mentioned in section 2.6 our model does not properly consider large amount of FQDN.

Figure 2.5 shows the $\frac{CPU^{arch.}}{CPU^{IPs}}$ for the different architectures versus γ. With γ as small as 40 FQDNs, makes *Pastry-AC* 1.39 (resp. 1.09) times more efficient than IP_{SHA1} (resp. *FQDN*) with DNS. With DNSSEC, this makes *Pastry-AC* 1.96 (resp. 1.64) times more efficient than IP_{SHA1} (resp. *FQDN*). Figure 2.2b shows that 40 FQDNs per node for a 18 node platform makes 56% of the traffic is cached by the platform. Similarly $\gamma = 200$ represents 72% of the traffic and makes *Pastry-AC* 2.18 (resp. 1.72) times more efficient than IP_{SHA1} (resp. *FQDN*) for DNS. For DNSSEC, it makes *Pastry-AC* 3.04 (resp. 2.54) times more efficient than IP_{SHA1} (resp. *FQDN*). $\gamma = 200$ may still provide a good compromise between a small table each node is responsible for and a consequent amount of the DNS traffic stored in cache, however, in the next section we consider $\gamma = 100$. $\gamma = 100$ makes *Pastry-AC* more efficient than *Pastry-PC*, and keeps the size of the cache small.

In fact Figure 2.5 also shows that for DNS DHT based architecture have little or no advantage over IP_{SHA1}. In fact IP_{SHA1} is 1.36 time more efficient than *Pastry*, respectively 1.06 times more efficient than *Pastry-SF*, equivalent to *Pastry-PC* and 1.04 more efficient than *Pastry-R* with $k = 3$ neighbors. Then, *FQDN* is 1.27 more efficient than IP_{SHA1}. As a result, *FQDN* is much more efficient than IP_{SHA1}, however, DHT based architecture generates management traffic can not compensate the costs of the DNS resolutions. On the other hand, DHT based architectures are much more efficient than IP_{SHA1} with DNSSEC. *Pastry* (respectively *Pastry-SF*, *Pastry-PC*, *Pastry-R*) is 1.6 times more efficient than IP_{SHA1} (respectively 1.82, 3.5, 2.01). *FQDN* is 2.06 times more efficient than IP_{SHA1}. The difference between *FQDN* and DHT based architecture is essentially constituted by management traffic. This management traffic seems negligible with DNSSEC but not with DNS. *Pastry-AC* and *Pastry-PC* are taking advantage of the Zipf distribution of the FQDN popularity (cf. figure 2.2b). The difference is that *Pastry-AC* provides to other node's cache only the most popular FQDN, whereas *Pastry-PC* provides all requested FQDN. Our model does not make any difference between large and small cache, but this consideration makes our preference for *Pastry-AC*. However, *Pastry-PC* is much easier to implement the *Pastry-AC*, since no popularity tables are required. In the remaining of this chapter we consider $\gamma = 100$.



(A) γ_{Nbr} (DNS)

(B) γ_{Nbr} (DNSSEC)

(C) γ (%) (DNS)

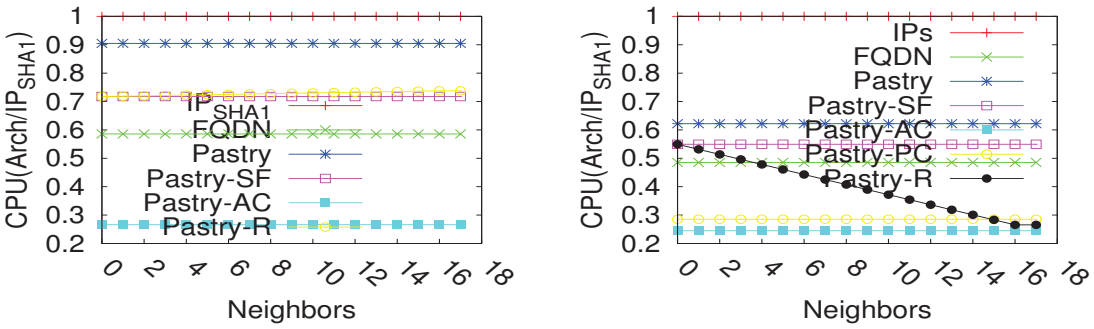
(D) γ (%) (DNSSEC)

FIGURE 2.5: Evaluation of the Impact of γ on CPU

Figures 2.6b shows how much neighbors should be considered so to maximize *Pastry-R*. *Pastry-R* can be seen as *Pastry-SF* combined with cache replication mechanism. Any resolution is followed by a broadcast of the response on k Neighbors. This improves CHR on each node and presents robustness against node failover. Only traffic with high value of CHR makes *Pastry-R* worth being considered, otherwise, the platform would be easily overloaded. However Zipf distribution presents a heavy tail (cf. figure 2.4) so most FQDN does not need to be cached, since they won't be queried twice. Caching unnecessary data results in an increase of the cache size with presents more power consumption operations. As such cache lookup for a cache of length l has a complexity of $O(\log l)$ and sorting in $O(l \log l)$, which makes local cache operation less efficient. *Pastry-AC* improves *Pastry-R* on both points, by considering only the most popular FQDNs. Furthermore *Pastry-AC* implements a cache hierarchy that leaves cache updates for popular FQDN aside the resolution process. This is why we prefer *Pastry-AC* to *Pastry-R*. The only advantage of *Pastry-R* over *Pastry-AC* is that *Pastry-R* does not have to evaluate FQDN popularity.

Figure 2.6 does not consider the overhead of larger cache, but shows that with DNS the number of neighbors does not influence much, and that *Pastry-R* is very close to *Pastry-SF*. This means that the cost due to replication balances the Cache Miss and management traffic with *Pastry-SF*, which does not provide *Pastry-R* any advantage over *Pastry-SF*. DNSSEC makes CPU^R much more costly than CPU^H, and the replication mechanism makes *Pastry-R* almost as efficient as *Pastry-AC* - with approximation of our model. *Pastry-R* presents the advantage of being completely deterministic compared to *Pastry-AC*, however, We favor *Pastry-AC* for cache efficiency, and management traffic optimization. In the remaining of this chapter we consider $Neighbors = 3$ which makes *Pastry* 1.707 more efficient than IP_{SHA1} with DNS (resp. 2.15 with DNSSEC). Figure 2.6b shows

that *Pastry-SF* is almost equivalent to *Pastry-R* with DNS. This means that high CHR of the DNS traffic makes replication for caching equivalent to forwarding without caching. With DNSSEC and large values for k, that is to say replication is performed on almost all nodes of the platform, *Pastry-R* is almost as efficient as *Pastry-AC* or *Pastry-PC*. This means that providing the request to the querying node on a per-query base is similar to proactively providing the query to all nodes of the platform. For DNSSEC figure 2.6b shows that for *FQDN* and *Pastry-R* are equivalent for k between 3 and 4. In the remaining of this chapter we consider $k = 3$.



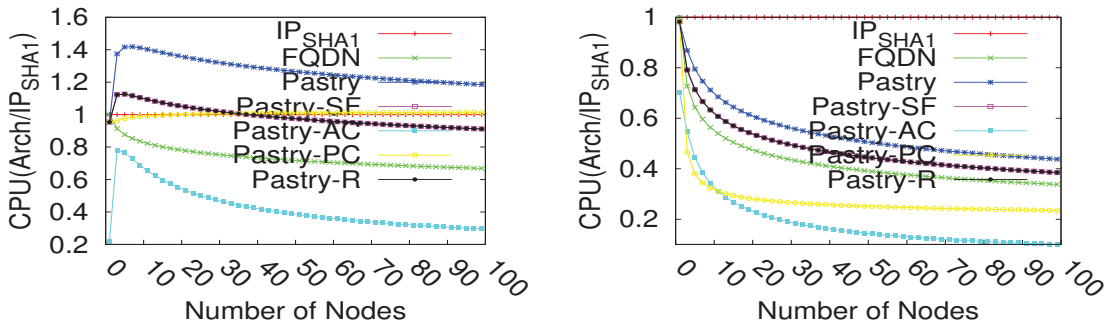
(A) DNS (B) DNSSEC

FIGURE 2.6: Evaluation of the Impact of Neighbors on CPU

Simulations are run with $n = 18$, $\gamma = 100$ FQDN per nodes, Neighbors $k = 3$. All tests below figure out how stable are the architectures when parameters vary. The considered parameters are: Number of nodes of the platform, the relative cost $\frac{CPU^R}{CPU^H}$, the TTL, and the Query Rate.

2.7.2 *CPU* & Scalability

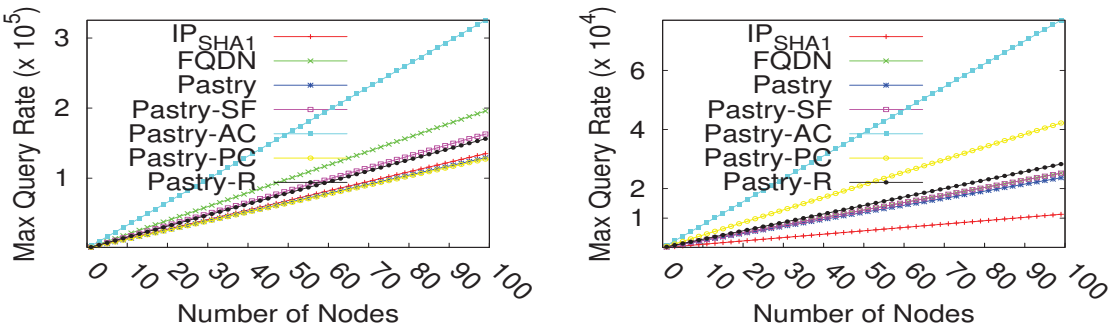
Figure 2.7 compares a randomly chosen node's efficiency when node number varies from 1 to 100. The considered traffic is a DNS traffic for 18 nodes, and we have seen that migration to DNSSEC may involve 5 times more nodes. As such, it is valuable to consider nodes for 20 to 100 for a given traffic. With DNS, and a 18 node platform, *Pastry-AC* (resp. *FQDN* performs 1.8 (resp. 1.3) times better than *IPs*. *Pastry-SF* happens to perform better than IP_{SHA1} for a number of nodes greater than 40. *Pastry* never outperforms IP_{SHA1}. However, *Pastry-SF* and *Pastry-R* are more efficient for a number of nodes lower than 49. Pastry architectures clearly show how management tasks within the platform balances the resolutions tasks. When the number of nodes grows, then IP_{SHA1} performs more resolutions, which are costly tasks, whereas *Pastry** increases management tasks that are of lower costs. On the other hand, with DNSSEC resolutions are much more costly, thus making Pastry architectures more efficient than IP_{SHA1}. *Pastry*, *Pastry-SF* and *FQDN* differ in the way End Users queries are handled by the requested node. With *FQDN*, the query is sent to the Responsible Node, which means there are no interactions with the other nodes when a cache miss occurs. With*Pastry*, the requested node may not be the Responsible Node, and in that case, the requested node has a complete query/response exchange with the Responsible Node. With *Pastry-SF*, the exchange with the Responsible Node is reduced, and only the query is forwarded to Responsible Node. The Responsible Node is expected to send the response directly to the End User. Considering DNSSEC and $n = 90$ *Pastry-AC* (resp. *FQDN*) performs about 9.5 (resp. 2.38) times better than IP_{SHA1}.



(A) DNS (B) DNSSEC

FIGURE 2.7: Evaluation of the impact of the number of Nodes of the platform on CPU. CPU is presented as a ratio of the CPU required by an architecture vs the CPU required by IP_{SHA1}

Another representation of the CPU consumption is to represent the maximum query rate for each architecture as a function of the number of nodes.



(A) DNS (B) DNSSEC

FIGURE 2.8: Evaluation of the Maximum Query Rate over various Architecture

2.7.3 *CPU* & $\frac{CPU^R}{CPU^H}$

Figure 2.9 compares architecture's efficiency versus the ratio $\frac{CPU^R}{CPU^H}$. The higher $\frac{CPU^R}{CPU^H}$ is, the more *Pastry-AC*, *FQDN*, *Pastry-SF* and *Pastry* are efficient compared to *IP*. *FQDN* and *Pastry-AC* are always more efficient than IP_{SHA1}, but *Pastry* (resp. *Pastry-SF*) are more efficient than IP_{SHA1} only for $\frac{CPU^R}{CPU^H} > 5$ (resp. $\frac{CPU^R}{CPU^H} > 10.5$). In our case we have for a single signature check $\frac{CPU^R}{CPU^H} = 11.490$ for BIND9 (resp. 17.79 for UNBOUND). If one considers that roughly 3 signature checks may be required - one for the ANSWER, the AUTHORITATIVE and the ADDITIONAL section -, then $\frac{CPU^R}{CPU^H} = 19.41$ for BIND9 (resp. 38.69 for UNBOUND) with 3 signature check per response. This makes *Pastry* and *Pastry-SF* more efficient than IP_{SHA1}.

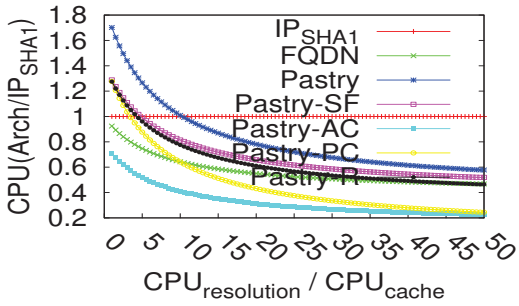
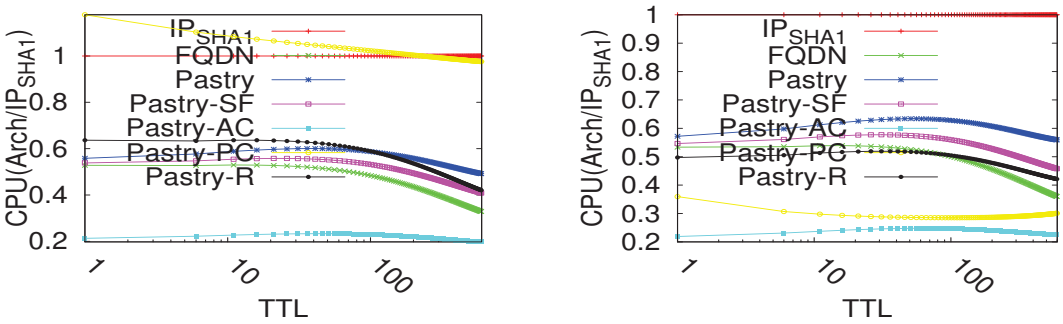


FIGURE 2.9: Evaluation of the Impact of $\frac{CPU^R}{CPU^H}$ on CPU

2.7.4 Evaluation of TTL impact over CPU

Sensitivity to TTL makes us considering how CDN may impact different architectures with low TTL values. CDNs are expected to increase queries of most popular FQDN as well as to reduce their TTL. On the other hand, our model considers a single TTL value for all FQDNs, and we do not consider more queries. Figure 2.10 stresses that what matters is the number of queries received during TTL. The higher this ratio is, the more caching architectures like *Pastry-PC* or *Pastry-R* take advantage of it. With DNS, *Pastry-PC* spends more resources on requesting, caching and resolving queries from others than only resolving queries for itself. Thus cache sharing is inefficient with *Pastry-PC* when TTL becomes lower. With DNSSEC, requesting a node on the platform avoids signature checks, and makes cache sharing efficient even for *Pastry-PC*. With large TTL value, all requested FQDNs are cached on the platform, or on the nodes. Responses cached on the node increases availability and makes *Pastry-AC* and *Pastry-PC* more efficient.



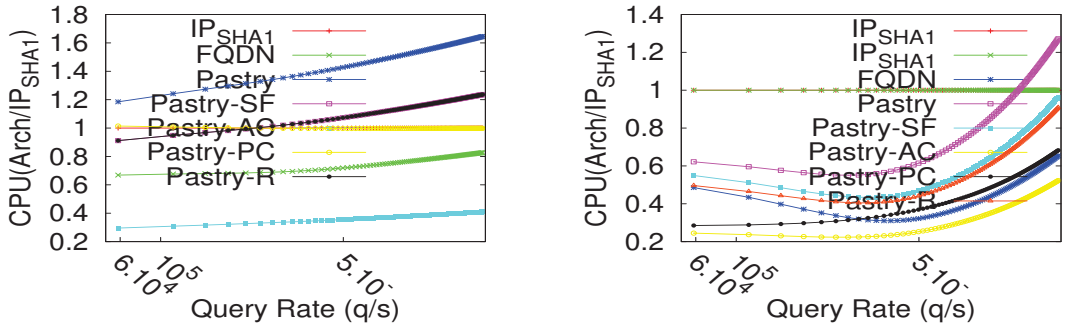
(A) DNS (B) DNSSEC

FIGURE 2.10: Evaluation of the Impact of TTL on CPU

2.7.5 *CPU & Query Rate*

Figure 2.8 shows how architectures are affected by small variations over the traffic the platform has been designed for. Since the number of FQDN is fixed, increasing the number of queries increases the CHR, after a while there is no more cache insertion to be performed. In other words, our

model does not consider a simultaneous growth of the traffic as well as the number of different FQDNs. When everything is cached Pastry architectures' management tasks are non negligible and constitutes the main traffic.



(A) DNS (B) DNSSEC

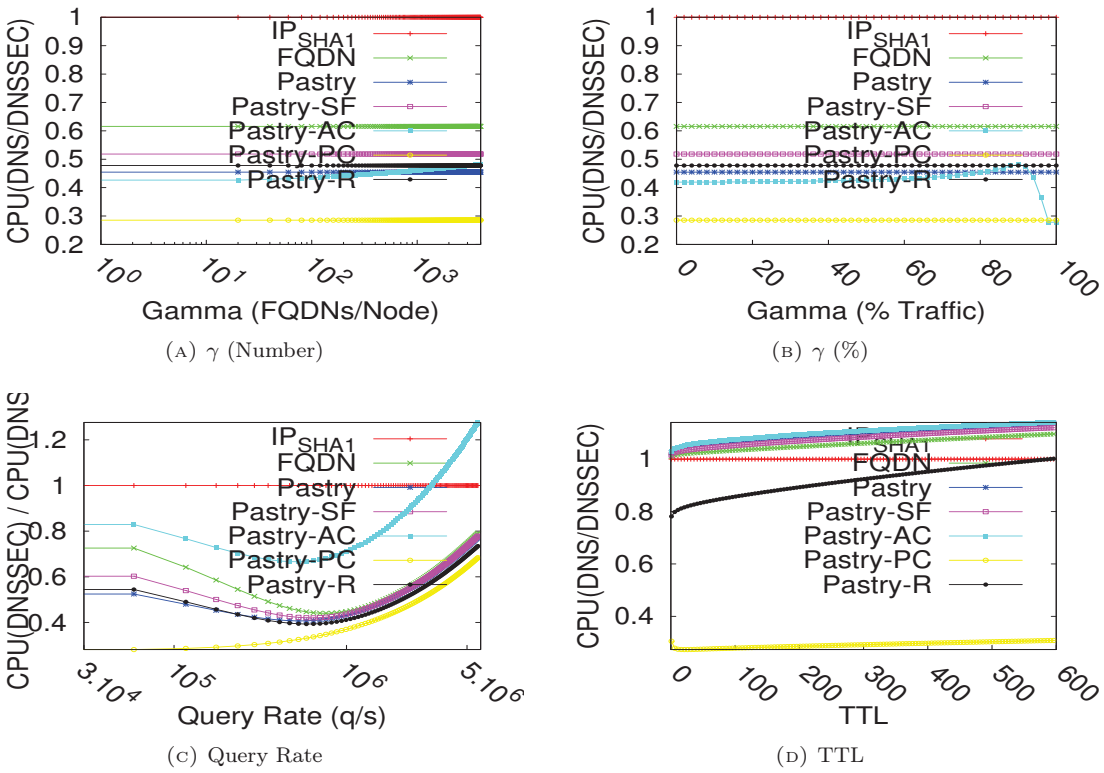
FIGURE 2.11: Evaluation of the Impact Query Rate on CPU

2.8 DNSSEC Migration

2.8.1 Analysis on CPU Time

This section shows the impact of the DNSSEC migration versus DNS for different parameters and compares it to the IP_{SHA1}.

Figure 2.13a shows that *Pastry-R* with a replication to less than 15 nodes out of the 18 nodes presents the most expensive cost for migrating from DNS to DNSSEC with signature.



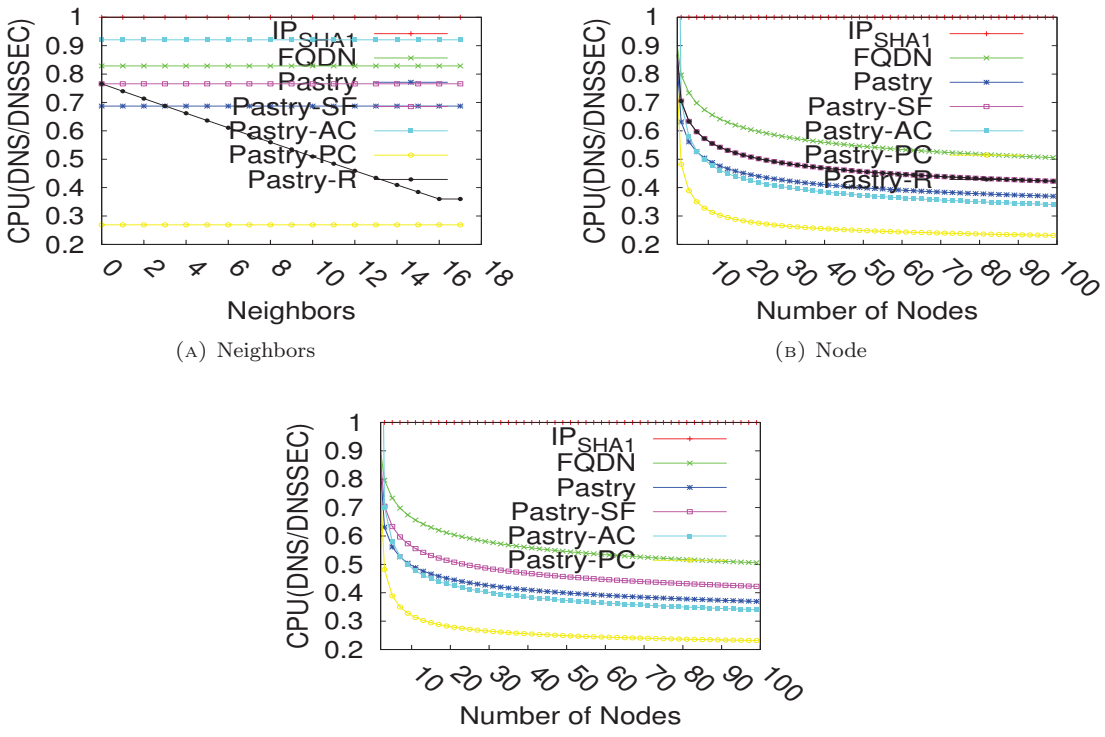
(A) γ (Number)

(B) γ (%)

(C) Query Rate

(D) TTL

FIGURE 2.12: Comparison between the various DHT architectures: Impact of Traffic Parameters (γ Query Rate and TTL) on the Platform CPU Time $\frac{CPU_{DNSSEC}}{CPU_{DNS}}$



(A) Neighbors

(B) Node

(C) Node (no *Pastry-R*)

FIGURE 2.13: Comparison between the various DHT architectures: Impact of Platform and Implementation Parameters (Neighbor, Node and $\frac{C_R}{C_H}$) on the Platform CPU Time $\frac{CPU_{DNSSEC}}{CPU_{DNS}}$

2.8.2 Analysis on Response Time (RT)

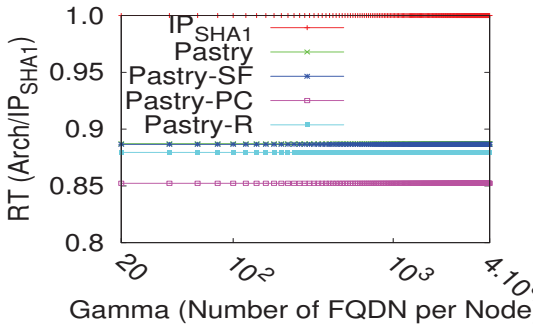
This section presents the same results as in section 2.8.1 but considers the Response Time (RT) instead of CPU Time. RT reflects the Quality of Service provided to the End User, whereas CPU is considered in the design of the platform. CPU and RT are not uncorrelated, in the sense that servers with large CPU will provide responses with smaller RT. However the difference between RT provided by an idle server and a loaded server may be negligible compared to the network latency.

In this section since we considered the CPU load to estimate the RT on the platform, we need to provide different configurations for DNSSEC and DNS. [MGL10] shows a close relation between RT and CPU of the platform. If we want to take advantage of such measurement, we need to have realistic configurations for both DNS and DNSSEC. In that sense, simulation with DNS considers a 18 node platform whereas simulation for DNSSEC considers a platform of 90 nodes. The main goal is to find an estimation of CPU on each node for each architecture that is between 0% and 100%. We chose $n = 90$ since migration of the IP_{SHA1} to DNSSEC requires the number of nodes to be multiplied by 5.

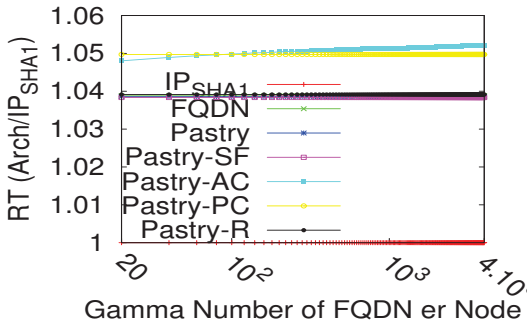
First of all comparing the different architecture based on the RT do not provide differences that are over 10%, which may not be sufficient to compete with the differences provided by comparison of CPU. CPU shows that *Pastry-AC* can be up to 350% more efficient than the traditional architecture. This means that adopting *Pastry-AC* may reduce a 90 node platform to a 28 node

platform. Balancing the number of servers to a 10% improvement in term of QoS may not be reasonable. On the other hand our first conclusion from figures 2.14 and 2.15 is that none of the proposed architecture may be rejected because of bad QoS.

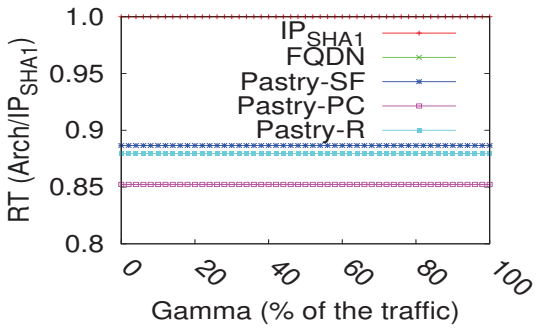
Then [MGL10] shows that the CPU consumption for both UNBOUND and BIND9 implementation presents a peak for a given load. This means that with higher load we can measure lower CPU load. Although no plausible explanation has been provided to explain such phenomenon, but they might slightly impact the Response time of the platform.



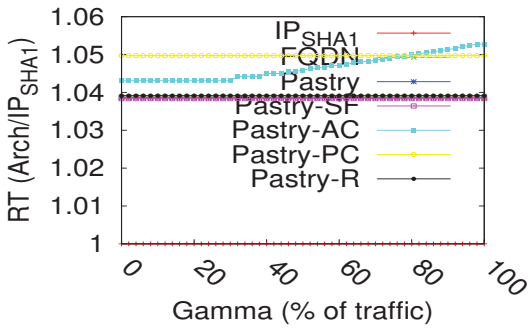
(A) γ (Number) - DNS



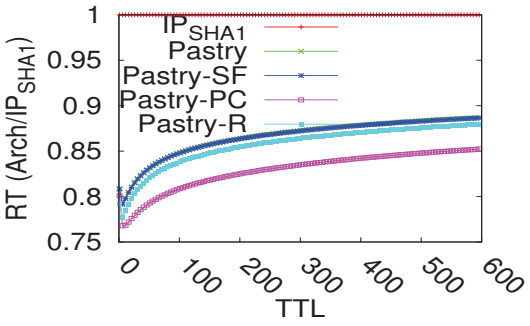
(B) γ (Number) - DNSSEC



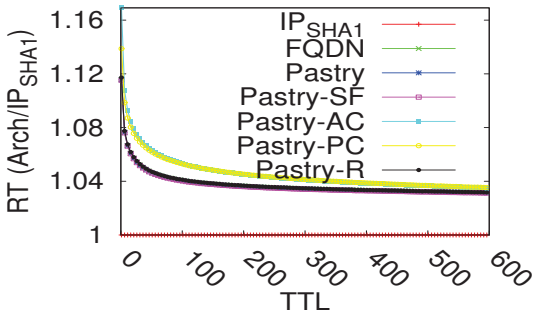
(C) γ (%) - DNS



(D) γ (%) - DNSSEC

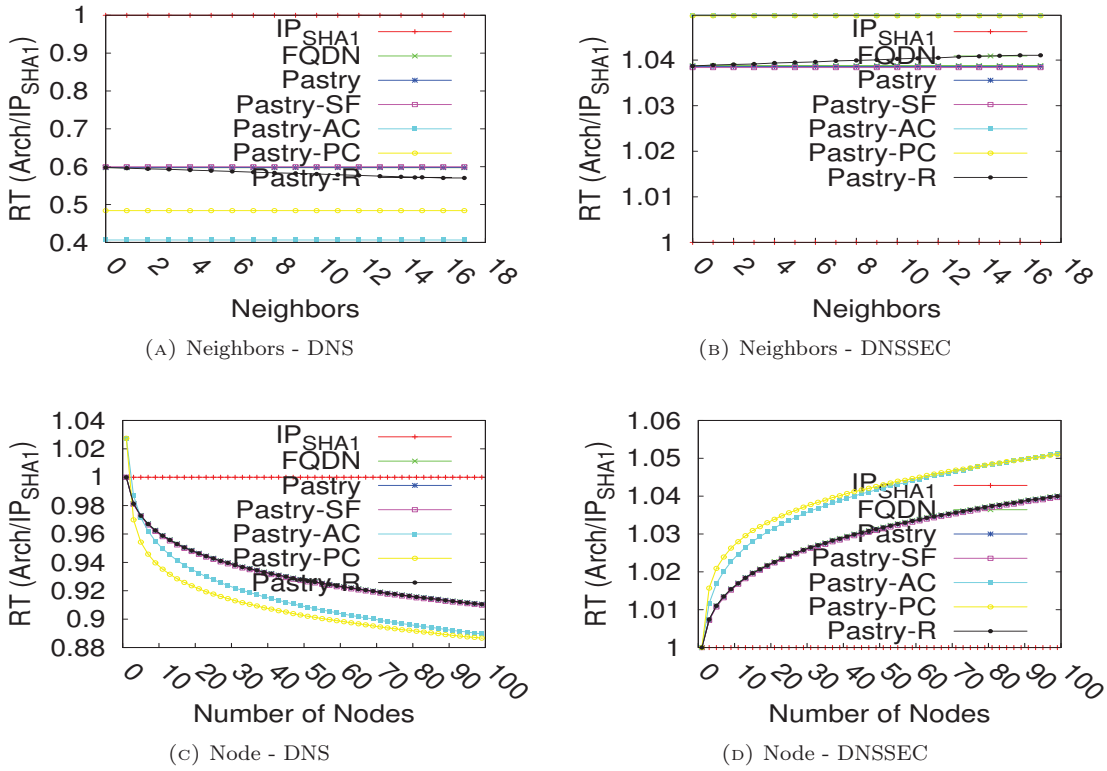


(E) TTL - DNS



(F) TTL - DNSSEC

FIGURE 2.14: Impact of Traffic parameters (γ and TTL) on the Platform Response Time



(A) Neighbors - DNS

(B) Neighbors - DNSSEC

(C) Node - DNS

(D) Node - DNSSEC

FIGURE 2.15: Impact of Platform Parameters (Neighbors and Nodes) on Response Time

2.9 Conclusion

DNSSEC migration of large resolving platforms is expected to increase their size by 5 if one keeps on using the current DNS Resolving platform architecture. This chapter first evaluates different load balancing techniques and shows that load balancing according to the FQDN rather than IP addresses reduces the number of nodes by 30%.

Because FQDN load balancers significantly impact the ISP core network infrastructure and still providing a single point of failure, we considered Pastry based architectures and their associated cache sharing optimizations. With the Zipf distribution of FQDNs, we showed that *Pastry-AC* with Active Cache sharing mechanisms can be at least 3.5 times more efficient than traditional IP architectures.

We also have to consider such results regarding the approximations we performed: constant TTL, we did not consider the impact of the size of the cache on cache lookup or cache insertion. Then we also considered that IP addresses and FQDN were independent.

Although *Pastry-AC* is the most efficient architecture presented in this chapter, we believe this architecture can still be improved. The pro-active mechanism takes advantage of the Zipf distribution of the FQDNs' popularity. More specifically, the 2000 most popular FQDNs correspond to 70% of the traffic. Thus, populating the cache of the nodes with the most popular FQDNs, results in increasing the Cache Hit Rate, and distribute the load of the DNS queries between the nodes.

In fact, when the query is cached, the DNS query is handled by the receiving node and not the responsible node. With the proactive mechanism, the DHT node considers a common cache for the FQDNs that are pro-actively cached and the FQDNs the node is responsible for. The number of FQDNs the DHT node is responsible is quite large, thus increasing the resources needed for a cache lookup.

A first optimization would be to introduce some levels of caches. A first cache would only contain the most popular FQDNs —that is to say the pro-actively cached FQDNs —, a second cache would contain the FQDNs of the DHT. However, this adds complexity to the DHT process.

A second optimization, would consider to keep the DHT process simple, and to take advantage of hardware acceleration. The proactive mechanism can be extracted from the DHT process, and be a front end process to the DHT. Maintaining a 2000 entry cache, and performing lookups to this cache is a relatively light process that can run in Network Acceleration Cards, thus lightening the DHT process from 70% of the traffic. We believe these front-end processes can be more optimal than the DHT process.

Other area includes designing a light Pastry for small and managed Network. we re-used the original version of Pastry, but functions like routing protocols are not involved on our platform. We encountered some performance issues with our Pastry implementation, and believe that a Light Pastry protocol may be designed for the Small Managed Network. From this thesis, we have identified the two following points: (1) removing unnecessary functions —like routing —, and (2) improve failover recovery mechanisms. However, re-designing a Light Pastry protocol would require to also reconsider the Operations and Magement's requirements.

Chapter 3

PREFETCH
Architecture to overcome DNSSEC Performance Issue in large Resolving Platforms

3.1 Introduction

Internet Service Providers (ISPs) have always hosted Domain Name System (DNS) Resolution for both end users and Machine-to-Machine (M2M) communications. End User use Names because IP addresses have no meanings for them, but M2M communications are also using Names to make configuration independent of IP address modifications, like DHCP modifications or IP renumbering. However, little attention was paid to optimize DNS Resolving Platform. Therefore, DNS Resolving Platform currently consists in large farms of servers, and it is now crucial to optimize them to face the future demands on DNS and DNS(SEC) evolutions.

First DNS is now a crucial service for ISP's business. DNS is not only used by end users for web browsing, but ISP's services are heavily relying on DNS. Among these services, PSTN-VoIP convergence with E.164 NUmber Mapping (ENUM) [MD00, Fal00], Mobility Services like MIP6 [GKD07] or HIP [MN06] use DNS to host Rendez-vous Servers and Home Agents. Second, DNS traffic keeps on increasing by 8% per month over the last 10 years, which shows that the resources allocated to DNS are now $\approx 1.08^{12*10} = 10253$ times larger than it was at the beginning. For example our DNS Resolving Platform is composed of 8 18-node clusters. Third, current Internet sees an increase of DNS load balancers or CDNs that associate short Time To Live (TTL) to their DNS responses, thus resulting in increasing the number of resolutions. At last, the DNS Protocol itself is being replaced by DNSSEC [AAL+05a, AAL+05c, AAL+05b], the DNS SECurity extension. With signature checks, DNSSEC resolutions require at least 4.25 times more CPU cycles [MGL10] than DNS's. DNSSEC is already deployed in most Top Level Domains (TLDs) and ISPs need to consider that evolution in the design of their Resolving Platforms.

This chapter proposes the $PREFETCH_X$ architecture, designed to (1) reduce the number of nodes (or resources), and to (2) provide management facilities for Operation, Administration & Management (OAM). Section 3.2 positions the chapter versus existing work. Goals, motiva-

tions and design for this architecture are presented in section 3.3. $PREFETCH_X$ consists in caching $HEAD_X$, the X most popular FQDNs and handling the remaining FQDNs, $TAIL_X$ by a Distributed Hash Table (DHT) architecture. In fact, DHT avoids that different nodes perform resolutions for a given FQDN by assigning, to each FQDN, a node responsible for its resolutions. Furthermore, DHT protocols like *Pastry* come with auto-configuration mechanisms. Section 3.4 analyzes a 1 day DNS live capture traffic and derives that prefetching $X = 2000$ FQDNs results in a uniform distribution of the required resources among the nodes of the platform. Section 2.5 models different DHT architectures. Section 3.5 executes the different DHT models, as defined in section 2.5, so to derive the most efficient DHT architecture that deals swith $TAIL_X$. This evaluates $PREFETCH_X$ from live traffic analysis and from modelization. Finally, section 3.6 provides experimental measurements that confirm our DHT models. We implement a DHT based on FreePastry platform, and compare the measured maximum load to the maximum load provided by our models.

3.2 Related Work

Most popular DHT protocols are Chord [SMK+01], Pastry[RD01a], Tapestry [ZHS+04] and CAN [RFH+01], where nodes are self-organizing, leave and join the ring with no extra burden for the administrators. Pieces of data are randomly spread among nodes, thus resulting in a greater robustness against DDoS attacks. [CMM02] describes a DNS service based on Chord [SMK+01] and DHash [DKK+01]. DNS was run over DHT so to get rid of painful name server administration, and inherit good load balancing and robustness from DHT architecture. The paper reports an experiment with 1000 nodes in the chord ring serving as an authoritative server. Replication was turned off, which means that the information stored in a node is not replicated on the k other nodes. However, this architecture considers passive caching, which means that when a node performs a lookup, it stores the answer. DHash is block driven, which means that files in our case DNS responses are not hosted by one node, but blocks of the file may be distributed over multiple nodes. DHash as well as caching mechanisms provide a well load balanced traffic over the nodes. However, even though DHash and PAST have different design, since the size of (Cooperative File System) CFS blocks of DHash are up tens of kilobytes, in our case, DHash and PAST have more or less the same results. The major drawback of the DNS over Chord architecture is that it results in a too much high latency.

[Mas06, RHM09] analyze how DHT could enhance the robustness of the Naming System. The robustness of both Chord and DNS considers *Data failure rate*, *Path failure rate* and *Path length*. The DNS efficiency was proved to be linked to the popularity of its zone and the number of labels of the domain name, whereas the DHT efficiency is related to the popularity of its RRsets. In fact, the DHT main drawback is its heavy routing algorithms. DHT is also more robust to orchestrated attacks and could achieve the same availability of the current DNS with added mechanisms like proactive caching - Beehive [RS04].

Many works focused on Web caching architectures. [Wan99] describes different caching architectures, providing inputs on where and how placing the cache devices according to the Web requirements. [TDVK98] introduces a cost comparison to different distributed caching methods. [WVS+99] investigates the benefits of cooperative caching, and finds out that it might be useless for a large scale population. The DHT web caching methods are investigated with Squirrel [IRD02], which is based on Pastry [RD01a]. This paper compares two architectures, one using a home node dedicated to a bench of web pages and one with a home node that can also delegate the resolution to other nodes during a flash crowd. Here flash crowd designates an unexpected increase of traffic. It happens that the less flexible but simpler architecture proved to have better performances.

Our chapter differs from above papers where DHT ring is used for hosting authoritative data,

whereas our architecture uses the DHT Pastry as a way to define the node responsible for performing the resolutions. The way the data are stored into the DHT also differs from DHash and we used PAST. In DHash, blocks of the files are spread over the DHT nodes, whereas in PAST the whole file is hosted on the node. Our architecture is also expected to be at maximum one or two hundred nodes, whereas the DHT in [CMM02] experiments over a 1000 node platform which is mentioned as being a restricted number.

3.3 PREFETCH Architecture: Goals and Design

This chapter proposes an architecture that optimizes DNS(SEC) Resolving Platforms. Our current Resolving Platform is composed of $8 \times 18 = 144$ nodes, and a migration to DNSSEC would raise this number to around $144 \times 4.25 = 612$ nodes [MGL10]. Currently, load balancers split the DNS requests by XORing the IP addresses [alt03]. The proposed architecture must:

- 1. Reduce the number of nodes (or resources), and
- 2. Provides OAM facilities.

3.3.1 Better load balancing and smaller cache for reducing CPU

The resource we care about in this chapter is the CPU which is heavily used for DNS resolution with signature checks and cache lookup over large caches. As a result our goals are to (1) Reduce the number of resolutions, (2) Reduce the size of the cache, (3) Reduce the number of cache lookup.

3.3.1.1 Better load balancing

Our current platform IP_{XOR} load balances the DNS requests between the nodes by XORing the 24 lower bits of the IP addresses [alt03]. Hence, the Most Popular FQDNs (MPFQDNs) queries can be sent to any nodes, triggering redundant resolutions and redundant cache entries. This is avoided with $FQDN$ that load balances the DNS queries according to the FQDN. Compared to IP_{XOR} for a n node Resolving Platform, $FQDN$ reduces, for the MPFQDNs, by n the number of resolutions, number of cache lookup and the cache size of the nodes.

Section 3.4.1 measures, for different Load Balancer, the distribution of the queries, resolutions and CPU rate over the nodes. Compared to IP_{XOR}, $FQDN$ reduces by 30% the CPU, but its distribution among the nodes presents too large variations. Variations (or dispersions) are 3.32 larger for DNS and 1.78 times larger for DNSSEC, making $FQDN$ unlikely to be deployed.

One way to overcome the non-uniform CPU distribution is to define a routing table [FN10] for the MPFQDNs that results in distributing the CPU uniformly among the nodes, whereas the remaining FQDNs are load balanced with a hash function. This alternative is not considered in this chapter because it modifies the load balancer which interconnects the platform to the CORE Network, making this operation too much risky.

Instead, we re-use the IP_{XOR} load balancing and modify the nodes of the platform and cache the MPFQDNs on all nodes. Considering the traffic associated to the MPFQDNs, IP_{XOR} uniformly load balances the queries between the nodes, and thus the associated CPU. In addition, it avoids computing the routing table for the $FQDN$ load balancer. Because, DNS responses are cached, this reduces the number of DNS resolutions and partly fulfils goal (1).

3.3.1.2 Reducing cache size

To completely fulfill goals (1) and (2), nodes are organized in a DHT so each FQDN is associated to a single node, responsible for its resolution. For MPFQDNs, the Responsible Node (RN) maintains the cache on every nodes and is expected to prefetch the MPFQDNs, i.e. avoids a cache miss occurs. For the remaining FQDNs, only the Responsible Node (RN) hosts the response, either in its cache or after a resolution over the Internet if a cache miss occurs. This DHT architecture reduces the number of resolutions and reduces the cache size of each node thus fully achieving goals (1) and (2).

3.3.1.3 Reducing the number of cache lookup

It may be surprising to reduce the number of cache lookup by caching the Most Popular FQDNs (MPFQDNs) on all nodes. In fact, it differs from the one of the DNS(SEC) Resolving Server like BIND [BIN] or UNBOUND [UNB]. It is much smaller, can be isolated from the Resolving Server, and in this chapter, we have in mind, that it is hosted in a Network Hardware Acceleration Card (NHAC) [cav, end] which works as a front end on each node. For each incoming DNS query, a first cache lookup is performed by the NHAC, and passed to the Server only if a cache miss occurs. Note that alternative architectures may consider a few front end nodes dedicated to respond to MPFQDNs queries and to forward the remaining FQDNs queries to a back-end DHT platform. In all cases, the DNS Resolving Server performs fewer cache lookup, which achieves goal (3).
Note we are taking advantage of cache levels rather than any current cache strategies like Least Recently Used (LRU) or Most Recently Used (MRU). We believe this is the most efficient strategy because the cache of the DNS Resolving Server is expected to be large, making any operation very costly.

3.3.2 Pastry auto-configuration to reduce Operations, Administration and Maintenance

Operations, Administration and Maintenance (OAM) facilities provided by Pastry [RD01a] are mainly provided by auto-configuration when a node is added or removed from the platform. Note that in this chapter we only use a subset of the Pastry functionalities. More specifically, we do not use the *Pastry Dynamic Node Discovery* used to define which Pastry node hosts a given content. This mechanism is quite complex because DHT protocols have been designed for billions of dynamic nodes, making impossible each node to have a global knowledge of the platform. On the other hand, our platform is not expected to have more than a (few) hundreds nodes, and all nodes are administrated by the same ISP. This makes possible each node to consider the other nodes as its *Neighbors*, and thus does not require *Dynamic Node Discovery*. The Pastry mechanisms we are taking advantage of are: 1) Content distribution among the nodes of the platform —that is to say in this chapter: which hash function is used —and 2) auto-configuration mechanisms that make the platform recover from a *Neighbor* fail over or addition of a new node.

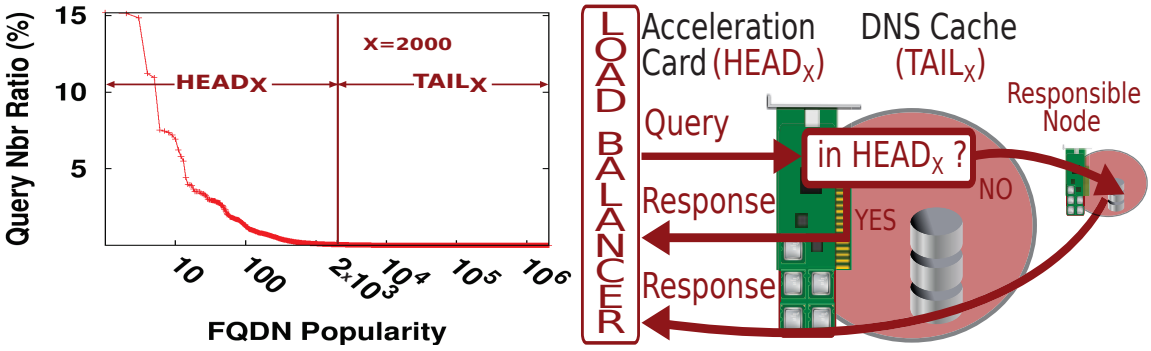
3.3.3 $PREFETCH_X$ Architecture Definition

This chapter proposes the $PREFETCH_X$ architecture illustrated by figure 3.1b. $HEAD_X$ stands for the X Most Popular FQDNs (MPFQDNs) cached in all nodes, and $TAIL_X$ the remaining FQDNS. Each FQDN is associated to a Responsible Node (RN), responsible for its resolutions.

The Load Balancer IP_{XOR} splits the queries between the nodes in an uniform way. Because the load balancer splits the traffic according to the IP addresses rather than the FQDN, a query for a given FQDN can be sent to any node with the same probability. $PREFETCH_X$ handles the queries differently, depending on whether it belongs to $HEAD_X$ or $TAIL_X$. When a node receives a DNS query from the Load Balancer, the query is first handled by the Network Hardware Acceleration Card (NHAC). The NHAC is a card placed on front end of the node, with its own CPUs. Figure 3.1b represents the NHAC connected to the node, facing the Load Balancer. In our case, it makes possible to proceed to treatments on the query —more or less like a firewall —and forwards it to the node's CPU only if necessary. More specifically, NHAC determines whether the queried FQDN is in $HEAD_X$ or not. If it is in $HEAD_X$, then the NHAC sends back to the End User the response, and no additional treatment is required from the node. The exchange is completely handled by the NHAC and is represented in figure 3.1b with the upper response. If the queried FQDN is not in $HEAD_X$, then it is in $TAIL_X$ and is forwarded to the DHT process of the node. From now, the resources involved are provided by the node. The node implements Pastry, and checks if it is responsible for that FQDN. In figure 3.1b, we represent the case where the node is not the Responsible Node. Thus the node sends to the RN associated to the queried FQDN. The Responsible Node checks its cache, eventually performs a resolution over the Internet, and sends the response to the node that in return sends the response to the End User. The response is represented in figure 3.1b with the lower response.

For simplification, we consider that all nodes have a NHAC, and that the Load Balancer balances the traffic to all nodes of the platform. Alternative architectures may also consider the Load Balancer balances the traffic to a subset of the nodes (the front end nodes) and that DHT is deployed on other nodes (back end nodes).

Figure 3.1a provides the FQDN query ratio versus the FQDN Popularity rank. The figure represents $HEAD_X$ and $TAIL_X$ for $X = 2000$. This value is derived from section 3.4 that defines how to derive X from a traffic analysis.



(A) X, $TAIL_X$, $HEAD_X$ (B) PREFETCH Node Achitecture

FIGURE 3.1: PREFETCH Node Parameters

3.4 Deriving X, $HEAD_X$, $TAIL_X$ from live capture traffic

$PREFETCH_X$ needs to set $HEAD_X$ so the dispersion of the CPU resources becomes marginal —that is to say all nodes of the platform use similar CPU rate for $TAIL_X$. $HEAD_X$, $TAIL_X$ and X are derived from the traffic, and should generate a dispersion that is valid whatever hash function is used by the DHT network and for any time of the day. In other words, the dispersion noted δCPU_X must be stable (1) to the DHT hash function and (2) to time. For DHT hash stability, we restrict our analysis to CRC32, MD5 and SHA1. We compute δCPU_X from a 10 minute DNS live capture of more than 35 millions queries. For various values of X, we remove the X most popular FQDNs, and check how the various hash functions distribute the CPU resources among the nodes of the platform. Then, we compute the dispersion. Note that the dispersion is a relative value, and thus dispersion δCPU_X can be compared for various values of X, as well as for various distributions. Since $PREFETCH_X$ has been designed to enhance IP_{XOR} described in section 3.4.1, in this chapter, X is chosen as the minimum value with a better δCPU_X, compared to IP_{XOR}. Then, we check δCPU_X time stability for the whole day.

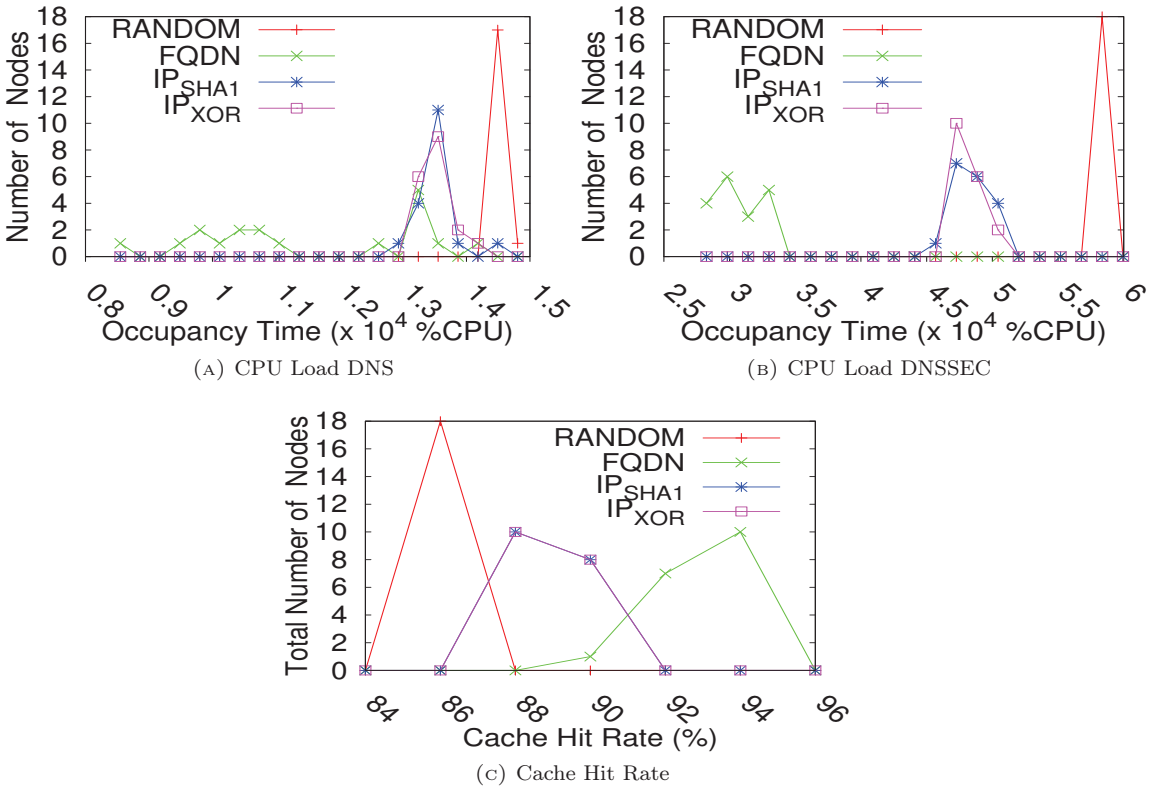
Section 3.4.1 points out the uniform resource distribution issue the $PREFETCH_X$ architecture is solving. More specifically, it shows that Load Balancers that balance the traffic according to the FQDN without removing the caching the X most popular FQDNS results in a very high dispersion of the resources between the nodes of the platform. Thus, it presents different Load Balancing techniques, and shows that load balancing according to the FQDN is not convenient without additional operations. To overcome this issue $PREFETCH_X$ proposes to cache $HEAD_X$, in all nodes of the platform. Then to lower the resource consumption of the nodes, $PREFETCH_X$ proposes to export the cache lookup function to a Network Hardware Acceleration Card. Section 3.4.2 defines δCPU_X the CPU dispersion, we compute $X = 2000$, and section 3.4.3 measures its time stability. Section 3.4.4 shows that $TAIL_X$ does not present a uniform popularity distribution, as figure 3.1a may eventually suggest. As a result, execution of the theoretical DHT models of section 3.5 must be based on our 10 minute live capture. Finally, section 3.4.5 concludes that prefetching divides at least by 2 the required resources for $PREFETCH_{2000}$ compared to IP_{XOR}.

3.4.1 Load Balancer Analysis

This section compares different Load Balancers. On our platform, the Load Balancer IP_{XOR} balances the DNS queries by XORing the 24 lower bits of the IP addresses. We position IP_{XOR} to IP_{SHA1} that performs a $SHA1$ rather than a XOR and then to $FQDN$ and $RANDOM$, that respectively balances the DNS queries according to the SHA1 of the FQDN, or by randomly choosing a node. By comparing IP_{XOR} and IP_{SHA1} we check whether using a hash function with better avalanche effect [Fei73] provides better load balancing. $FQDN$ shows that simple Load Balancer is not sufficient to enhance the Resolving Platform, and that nodes have to work closely with the Load Balancer. For example, they may agree with the load balancer on balancing policies that would result in distributing uniformly the resources between the nodes. $RANDOM$ selects a node independently of incoming packet and provides the best balance we can achieve.

From the live capture traffic, we computed the number of queries, Resolution and Cache Hit Rate (CHR) on each node. Using CPU measurements in [MGL10], we plot in figure 3.2 the CPU load of each node for DNS and DNSSEC. Figure 3.2 shows the CPU distribution over the nodes of the platform. For any value of CPU (on X-axis) the number of nodes (on Y-axis) that associate to the CPU value. Globally, $FQDN$ provides a 30% optimization over IP_*, and better CHR. On the other hand, the CPU load distribution is more uniformly distributed with IP_*. In fact, FQDN's

popularity distribution follows a Zipf distribution [JSBM01, BCF+99] which makes difficult to compensate a very popular FQDN, as shown in figure 3.1a. Thus, $FQDN$ is more efficient and provides a lower mean value for CPU load, but it generates too much dispersion in CPU load. The goal of $PREFETCH_X$ is to take advantage of $FQDN$ efficiency and to lower the CPU dispersion by choosing proper values for X. In the remaining of this chapter we will be focus on the dispersion aspect of distribution.



(A) CPU Load DNS

(B) CPU Load DNSSEC

(C) Cache Hit Rate

FIGURE 3.2: Platform and Traffic Distributions

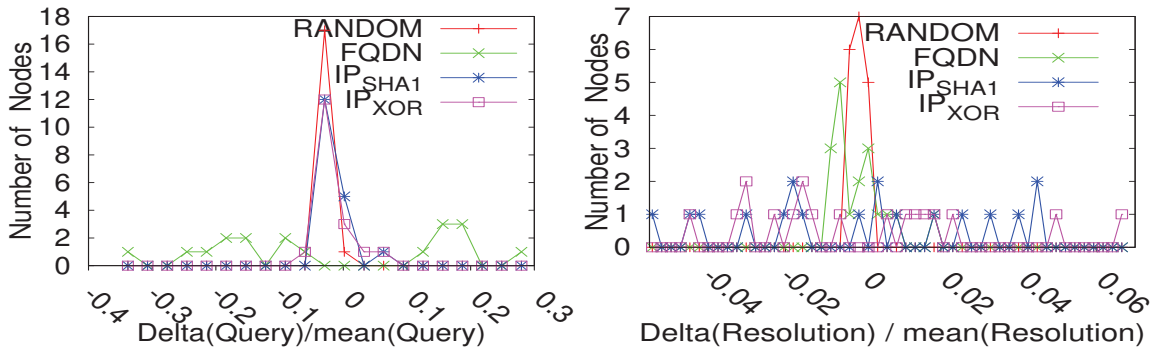
A good Load Balancer provides uniformly distributed queries and responses among a set of nodes, so nodes have exactly the same load. The dispersion is relative to the mean value, so we consider the relative distance to the means value. Figure 3.3 presents the distribution of the relative query (resp. response) number $\hat{Q}_i = \frac{|Q_i - \bar{Q}|}{\bar{Q}}$, with Q_i the number of queries on node i and $\bar{Q}$ the mean query number. Then the dispersion δ_Q defined in equation 3.2 makes dispersion comparison possible between different DHT architectures using different hash, different load balancing strategies, and different traffic over time.

$$\delta_Q = |MAX(\hat{Q}_i) - MIN(\hat{Q}_i)|, i \in [1..n] \tag{3.1}$$

$$\hat{Q}_i = \frac{|\, Q_i - \bar{Q} \,|}{\bar{Q}} \tag{3.2}$$

Compared to IP_{XOR}, figure 3.3 shows that IP_{SHA1} offers the same dispersion for queries δ_Q and for resolutions δ_R. Then $RANDOM$ has similar queries dispersion, but resolutions dispersion is

between 3 time smaller. At last, with $FQDN$ we have $\delta_Q^{FQDN} \approx 5\delta_Q^{XOR}$ and $\delta_R^{FQDN} \approx 0.026\delta_R^{XOR}$. As a result, $SHA1$ does not provide any benefit over XOR in term of load balancing, which confirms the use of XOR for load balancing DNS traffic. Then $FQDN$ is adapted for balancing resolution whereas XOR is more adapted for query load balancing. Considering that DNSSEC resolution requires 10.34 more CPU than with DNS, the difference explains why in figure 3.2 $FQDN$ performs better with DNSSEC than with DNS.



(A) Query (B) Resolution

FIGURE 3.3: Current Platform Distributions

3.4.2 Defining X for a proper δCPU_X

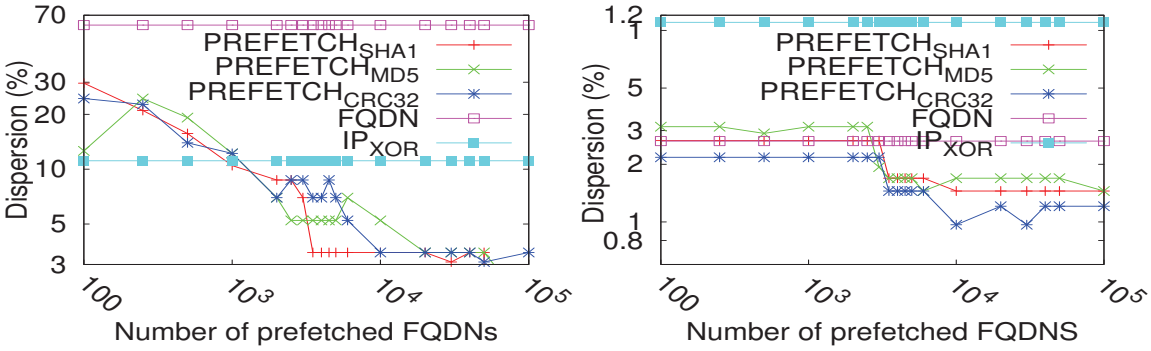
To evaluate $PREFETCH_X$, we are concerned about the CPU dispersion over the nodes of the platform δCPU_X. CPU can be expressed on node i as $CPU_i = Q_i.CPU_Q + R_i.CPU_R$, where Q_i (resp. R_i) designates the number of queries (resp. responses) handled by node i, and CPU_Q (resp. CPU_R) the CPU required for a query (resp. resolution). One issue is that CPU_Q and CPU_R depend on multiple parameters like the implementation, the protocol used (DNS vs DNSSEC), the FQDNs... If one does not want to rely on those parameters, one may consider that $\delta_{CPU_X} \leq max(\delta_{Q_X}, \delta_{R_X})(CPU_Q + CPU_R)$ and look for X_o that makes $\delta_{Q_{X_o}} \leq \delta_{Q_X^{XOR}}$ and $\delta_{R_{X_o}} \leq \delta_{R_X^{XOR}}$, where Q_X^{XOR} (resp. R_X^{XOR}) are the number of queries (resp. responses) resulting from the IP_{XOR} Load Balancer. Figure 3.4 plots δ_{Q_X}, δ_{R_X}, for various X values and for CRC32, MD5 and SHA1 as DHT hash functions. IP_{XOR} and $FQDN$ are also provided to visualize the performances of $PREFETCH_X$ over the IP_{XOR} and $FQDN$ Load Balancing architectures.

Table 3.1 compares ratios between $PREFETCH_X$ and IP_{XOR} for δ_Q, δ_R and δ_{CPU}. CPUs are estimated considering UNBOUND in [MGL10]. Minimum and maximum values depend on the DHT hash function (CRC32, MD5, SHA1).

Considering separately δ_Q and δ_R shows that $X = 2000$ is the first value that provides $PREFETCH_X$ a better CPU dispersion than IP_{XOR}. On the other hand considering the resulting CPU —instead of δ_Q and δ_R —shows that $X = 500$ would be fine. In fact $X = 2000$ is derived by setting 2 conditions whereas $X = 500$ is derived with a single condition.

X	$\frac{\delta_{Q_X}}{\delta Q^{XOR}}$		$\frac{\delta_{R_X}}{\delta R^{XOR}}$		X	$\frac{\delta_{CPU_X}}{\delta CPU^{XOR}}$	
	min	max	min	max		DNS	DNSSEC
1000	1	1	0.19	0.28	100	2.04	0.4
2000	0.63	0.78	0.32	0.47	250	1	0.2
4000	0.31	0.625	0.13	0.15	500	0.75	0.4
					2000	0.25	0.2

TABLE 3.1: Defining X



(A) δ_{Q_X} (B) δ_{R_X}

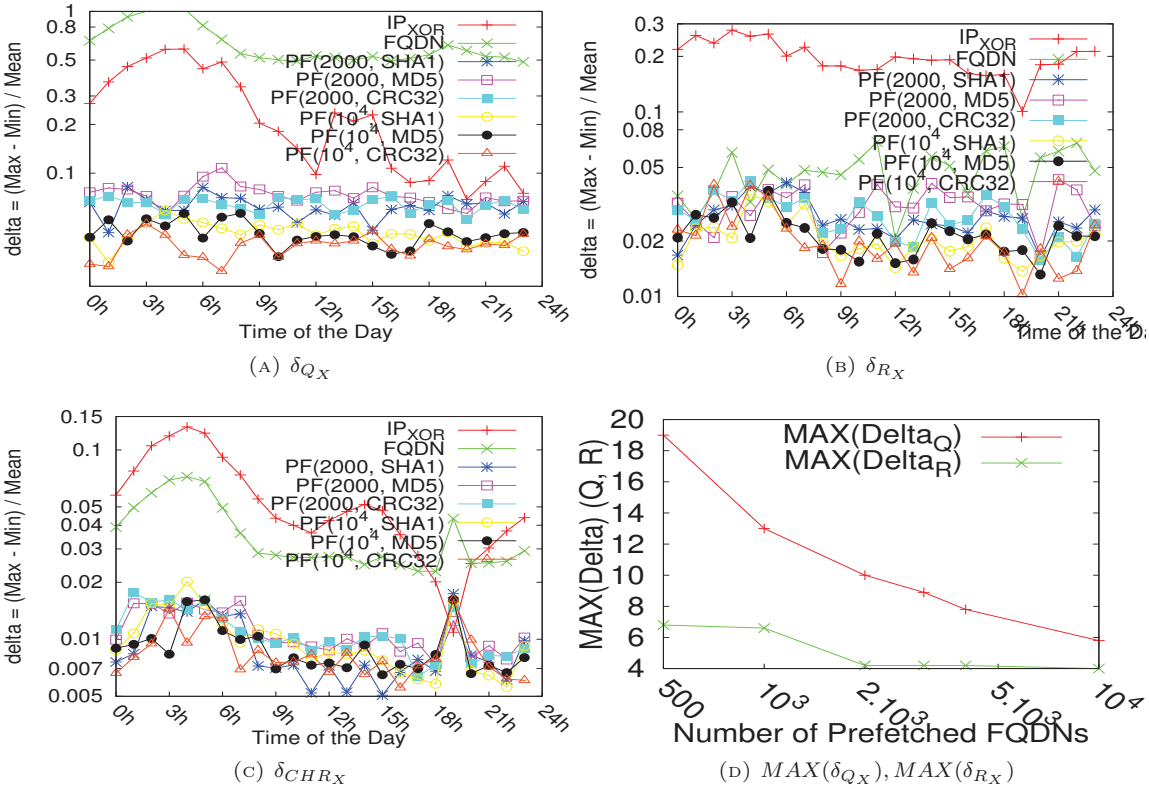
FIGURE 3.4: $PREFETCH_X$ δ_{Q_X}, δ_{R_X} Measurements

3.4.3 δCPU_X time stability

Figure 3.5 plots δ_{Q_X}, δ_{R_X} and δ_{CHR_X} every hour of the day for IP_{XOR}, $FQDN$ and $PREFETCH_X$ with different DHT hash. This is done for various values of X and figure 3.5d plots the maximum variation of δ_{Q_X} and δ_{R_X} observed during the day.

Figure 3.5a shows that, compared to $PREFETCH_X$, $FQDN$ and IP_{XOR} present the largest δ_{Q_X} (35% and 107%), and large variation during the day. In addition, their dispersion is larger at night when the traffic is quite low. $PREFETCH_X$ provides lower δ_{Q_X} and is more stable to traffic variations or time.

Figure 3.5d shows that with $X = 500$ δ_{Q_X} is below 20%, and that δ_{Q_X} exhibits logarithmic growth according to X, the number of prefetched FQDNs. For $X = 2000$ the maximum dispersion over the day does not exceed 10%. For δ_{R_X}, figure 3.5b shows that IP_{XOR} provides large dispersion, but any $FQDN$ or $PREFETCH$ provides a less than 8% dispersion. Figure 3.5d exhibits a step function for δ_{R_X} with low values for X the number of prefetched FQDNs greater than $X = 2000$. In a word, $X = 2000$ reduces $\delta_{Resolution}$ by 2.4% and $X = 10000$ does not significantly improve $X = 2000$. δ_{CHR_X} in figure 3.5c is impacted by δ_{Q_X} and δ_{R_X}, and shows that $PREFETCH_X$ architecture provides a more uniform efficiency over time than other standard load balancing.



(A) δ_{Q_X} (B) δ_{R_X}

(C) δ_{CHR_X} (D) $MAX(\delta_{Q_X}), MAX(\delta_{R_X})$

FIGURE 3.5: Time Stability

3.4.4 $TAIL_X$ Distribution

The various DHT models are executed for $TAIL_X$, and we check if as suggested by figure 3.1a, $TAIL_X$ can be modeled with a uniform popularity distribution. To check whether the $TAIL_X$ we perform a χ^2. A χ^2 test on $TAIL_{2000}$ and the null hypothesis "$TAIL_{2000}$ *is uniformly distributed*" is rejected. The χ^2 test is performed with the subset of FQDNs of our 10 minute capture traffic which provides a $2,061,865$ sample size. The test is run with an acceptable level of 0.001, with 100 degrees of freedom, and none of the 101 classes have less than 5 elements. χ^2 provides 192907396, which compared to 149.44 shows that $TAIL_{2000}$ does not present a uniform FQDN popularity distribution, as shown in figure 3.6. As a result, DHT must be evaluated by replaying the live capture for $TAIL_X$.

3.4.5 Discussion

The nodes of our DNS Resolving Platform are placed behind an IP_{XOR} Load Balancer and traffic is uniformly split between the nodes, but results in multiple redundant resolutions. From our 10 minute live capture we derived that caching $HEAD_{2000}$, the 2000 most popular FQDNs —that is 67.2% of the traffic —on all nodes, results in uniformly distributing the CPU among the nodes where each remaining FQDN of $TAIL_X$ is assigned to a single Responsible Node on the platform.

Because (1) $HEAD_X$ is not processed by the servers but by the NHAC, (2) $TAIL_X$ is uniformly distributed and (3) there is no redundant resolutions, $PREFETCH_{2000}$ is expected to require

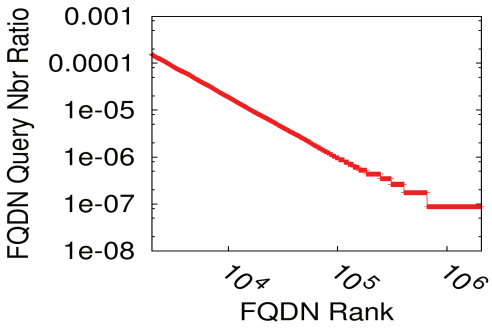


FIGURE 3.6: $TAIL_{2000}$ FQDN Popularity for FQDN with popularity rank below 2000

at least twice fewer nodes than IP_{XOR}. Furthermore, the size of the Server caches has been reduced, and DHT is also expected to enhance the performances. The latest point is the purpose of section 3.5. At last $X = 2000$ can be largely improved, with very few constraints.

3.5 Executing DHT models with $TAIL_X$

This section computes the distribution $TAIL_{2000}$ over various DHT architectures modeled in section 2.5. The efficiency of a DHT architecture depends on the relative cost of caching versus requesting another node. Caching may result in large expensive caches whereas requesting may result in heavy network operations. Because cache length's impact on performance is hard to estimate, in our evaluation, we assume that all cache operations have the same cost over the various DHT architectures. This is true as long as the cache remains small or of the same size.

Figure 3.8 exhibits how modifications of the traffic parameters —like query rate, TTL or the cost of a resolving versus a caching —impact the CPU ratio $\frac{CPU_{DHT}}{CPU_{IP_{XOR}}}$. Architectures based on routing provide better performances, and are more stable to TTL or query rate variations —with constant number of FQDNs. Thus *Pastry-SF* is recommended. *Pastry-PC* and *Pastry-R*, especially for DNSSEC, take advantage of large TTL values and large query Rates because the number of FQDNs remains constant in our computations. Hence, increasing TTL or query rates result in increasing the CHR.

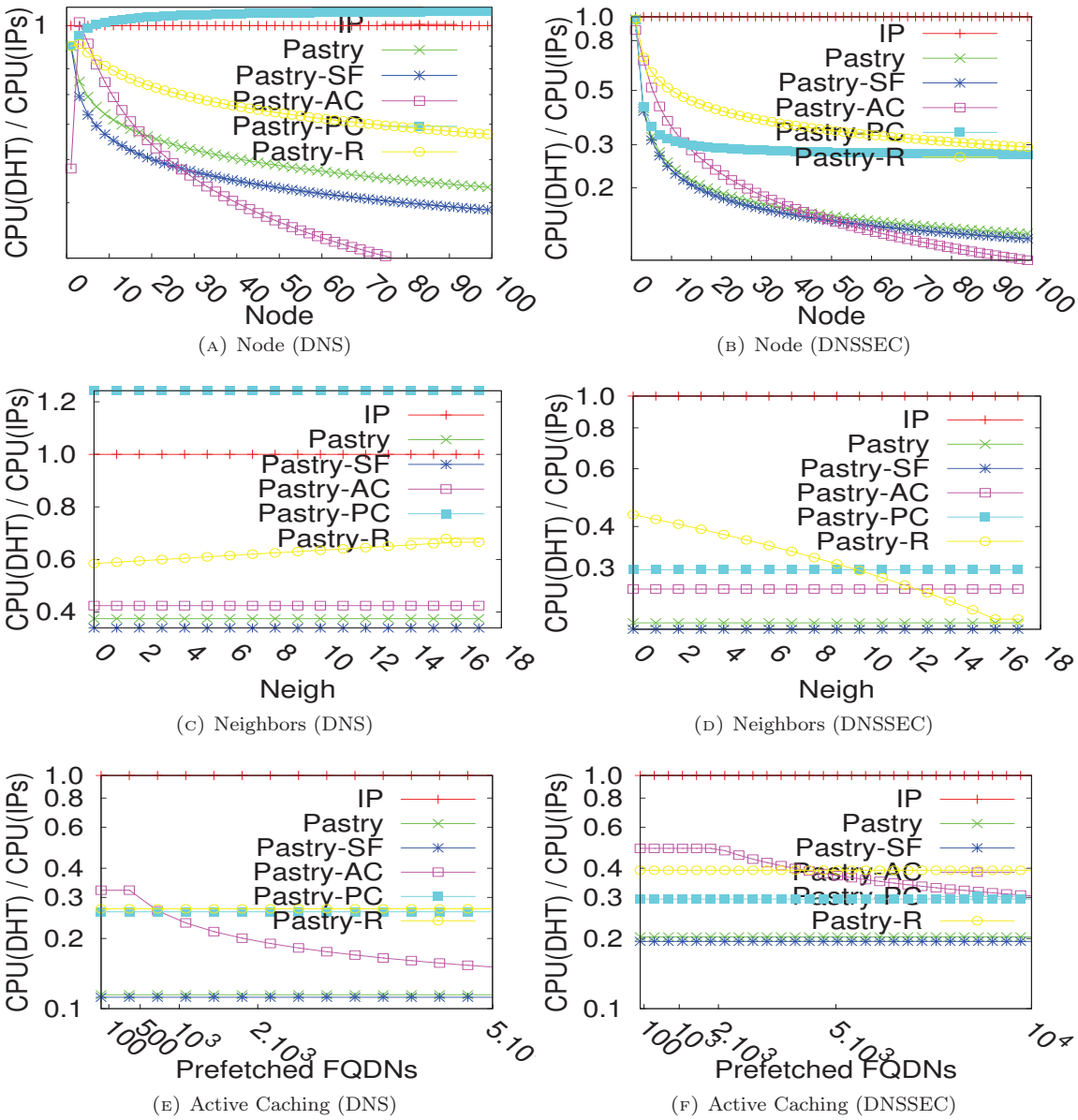
Figure 3.8e shows that for $R_{CPU} = \frac{CPU_{Resolution}}{CPU_{Cache}} \geq 20$, DHT provides a clear advantage over IP_{XOR}, which remains stable for greater values. [MGL10] measured on UNBOUND $R_{CPU}^{DNS} = 3.74$ and $R_{CPU}^{DNSSEC} = 38.69$, which shows *Pastry-SF* only requires 55% of IP_{XOR} resources for DNS and 19.2% for DNSSEC.

Figures 3.8a and 3.8b show that with $TAIL_{2000}$, TTL above 100 have small impact on the platform. Furthermore, routing packet is lighter than caching operations, which makes *Pastry-SF* and *Pastry* the recommended architectures for $TAIL_X$. Since our simulations run with a constant number of FQDNs, it reduces the Cache Hit Rate (CHR), and increasing the query rates in figures 3.8c and 3.8d increases the Cache Hit Rate. This makes *Pastry-PC* and *Pastry-R* take much more advantage to other architecture, especially with DNSSEC. Architectures provide a balance between caching mechanisms and routing and the efficiency of an architecture really depends on the traffic characteristics.

Figures 3.7a and 3.7b confirm that DHT architectures reduce by up to 60% the CPU consumption over IP_{XOR}. With DNS, *Pastry-PC* requires more CPU than IP_{XOR} because DNS resolutions are light and caching adds an overhead. Then, *Pastry-AC* is clearly the most scalable

architecture, and is able to lower the load by expanding the number of prefetched FQDNs. This is confirmed by figures 3.7e and 3.7f. However, *Pastry-AC* is redundant with $PREFETCH_X$ which is expected to take advantage of NHAC, and *Pastry-SF* is the most appropriated architecture in term of scalability.

Figures 3.7b and 3.7c show that response replication decreases performance with DNS, whereas it proves to be beneficial with DNSSEC due to the relative costs of caching versus forwarding. However replication on multiple nodes increases the size of the cache which is not taken into account in our models. *Pastry-R* is useful in case of failover and future work should measure how starting a node with empty cache may impact the platform. If it is of importance, then the marginal cost provided by figure 3.7b and 3.7c may require to activate replication.



(A) Node (DNS)

(B) Node (DNSSEC)

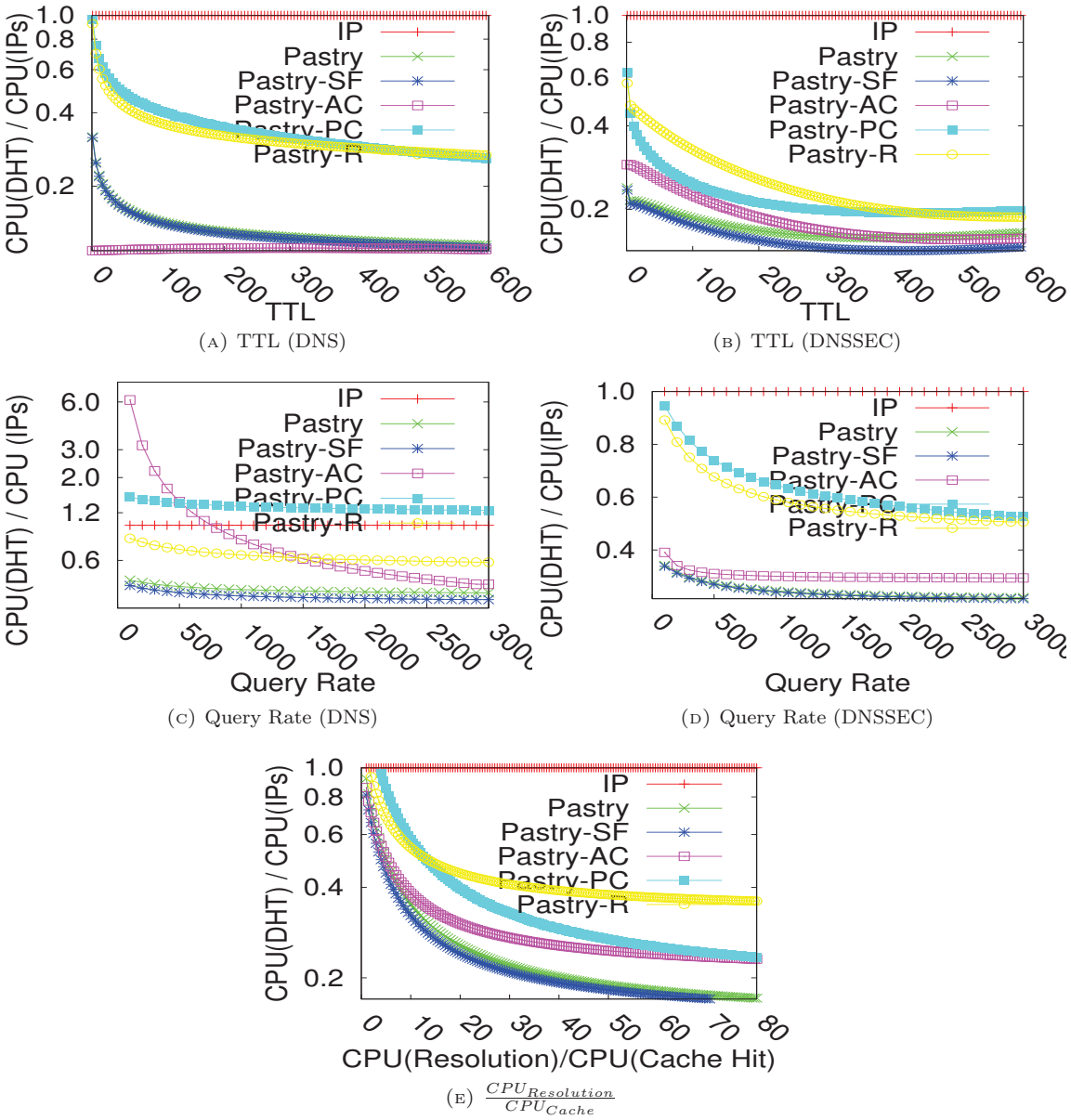
(C) Neighbors (DNS)

(D) Neighbors (DNSSEC)

(E) Active Caching (DNS)

(F) Active Caching (DNSSEC)

FIGURE 3.7: $TAIL_{2000}$ Architecture Evaluation



(A) TTL (DNS)

(B) TTL (DNSSEC)

(C) Query Rate (DNS)

(D) Query Rate (DNSSEC)

(E) $\frac{CPU_{Resolution}}{CPU_{Cache}}$

FIGURE 3.8: $TAIL_{2000}$ Network Impact

In section 3.4.5, we concluded that prefetching $X = 2000$ results in dividing the number of nodes of $PREFETCH$ by 2 compared to IP_{XOR}. In this section we show that, for the remaining $TAIL_X$ traffic, DHT can reasonably decrease the necessary resources by 55% and 80%. Note that $TAIL_X$ has small CHR, thus combined to prefetching, DHT decreases by roughly between 20% and 35%, which confirms the 30% resource reduction provided by the $FQDN$ Load Balancer. As a result, $PREFETCH_X$ requires at least 4 times fewer nodes than IP_{XOR}. The purpose of section 3.6 is to validate our models with experimental measurements.

3.6 Free Pastry Experimentation

This section validates our models by implementing and testing a DNS platform based on FreePastry [Fre], results are compared to those provided by the theoretical model of *Pastry* which models Pastry without any cache and IP_{XOR}. In this section, a uniform FQDN popularity distribution is used for the experimentation and the simulation.

First, we measure the performance on a single node, and then extend the platform to 10 nodes. Our 10 node Pastry experimental platform is based on the Java FreePastry library (version 2.1alpha3) [Fre]. FreePastry implements Pastry [RD01a] as the routing protocol between DHT nodes, PAST [RD01b] implements the DHT, and GCPAST on the top of PAST implements a garbage collector to remove expired contents. DNS responses are stored in the DHT and indexed by the hash of the queried FQDN and type, the assigned GCPAST TTL is the one associated to the DNS response. DNS resolutions and interaction with GCPAST are performed with DNSJava [Wel].

The format of the DHT contents must be chosen carefully, and we choose to store a complete answer involving multiple RRsets, rather than each RRsets individually. The hash key is built using the name, class and type of the query field. The main drawback is that it generates a high redundancy between the RRsets contained in different DHT contents and the size of the DHT caches on the nodes. For the TTL, we assign to GCPAST the TTL provided for the ANSWER field in the DNS response.

Our experimental platform is composed of 10 Pentium III and Pentium II servers with Debian *Lenny*. Although different configurations were used, the CPU frequency varies from $500\,MHz$ to $1\,GHz$, and RAM varies from $128\,M\,bytes$ to $384\,M\,bytes$. The 10 node platform is loaded with traffic with a $T = 3$ minute TTL. We set $CHR_o = 0.7$, so caches are pre-populated with FQDNs from a list, and 7 out of 10 queries are from this list. Tests last $T_{test} = 1$ minute so with a maximum of $Q = 1100\,q.s^{-1}$ with *Pastry*.

Figure 3.9a shows for different numbers of nodes n the percentage of answered queries. We estimated the Maximum Load (ML) reached when there are more than 2% errors. Figure 3.9b plots experimental measured ML, and plots the Stand Alone value (for $n = 1$, $ML(1) = 200$) as well as the linear approximation for the experimental value with $n \geq 2$: $ML^{Pastry}(n) = 114.7n - 96.9$. In a Stand Alone mode, the whole FreePastry network is supported by the same hardware, and no routing operations are required. Although in our theoretical model, we neglect here the routing table lookup, because we have reliable nodes. Our routing discovery protocol is much lighter than the one originally designed in Pastry, and all CPU estimations have been modeled by measuring a C implementation of a performance driven server: UNBOUND. In FreePastry and the DNS module we add results from academic development of a Java software, and routing table is definitely not negligible. As such, the Stand Alone mode does not provide a good estimation for the experimental IP_{XOR} measurement. In order to provide a experimental measurement for IP_{XOR}, we need to estimate the costs of routing table on a FreePastry node. $ML(n)$ estimates that routing table lookup cost to $\approx 96.9 queries.s^{-1}$, which leads to the experimental IP_{XOR} experimental measurement $ML^{IP}(n) = 200n - 96.9\,queries.s^{-1}$.

Figure 3.9c compares our theoretical and experimental results for $\frac{ML^{Pastry}}{ML^{IP}}(n)$. Theoretical and experimental measurements confirm that the ratio between *Pastry* and IP_{XOR} remains stable and independent of n. However, theoretical model expects *Pastry* to be equivalent to IP_{XOR}, whereas Experimental measurement shows that *Pastry* costs around twice as much as IP_{XOR}. The difference between the two results can be explained by the FreePastry implementation and the testing conditions: 1) FreePastry is not optimized for performance. Profiling the code with JRat [JRa] revealed that 64% of the time would be spent on insertion and DHT look up, if we were using FreePastry in synchronous mode. Using asynchronous mode leverages this bottleneck, but does not mean the code is now optimal. In addition, 2) Routing operations are much heavier than those we require in *Pastry*. When we compare the time necessary to perform a resolution in a Stand

Alone configuration or in a FreePastry configuration we measure that when the DNS response is in the cache, it takes $5\,ms$ to $65\,ms$ (resp. $4\,ms$ to $300\,ms$) in a Stand Alone mode (resp. in a DHT mode). When the responses require a resolution it takes $8\,ms$ to $175\,ms$ (resp. $59\,ms$ to $342\,ms$) in a Stand Alone (resp. DHT) mode. In both cases time variation reveals that optimization may be improved in event thread synchronization of the FreePastry implementation. On the other hand, it also shows that the routing function is clearly not negligible in this implementation. At last, the lake of FreePastry and hardware performance only makes experimental measurements under quite small load which makes overhead more visible. In our case, routing overhead consists in a bit less than half of the performance.

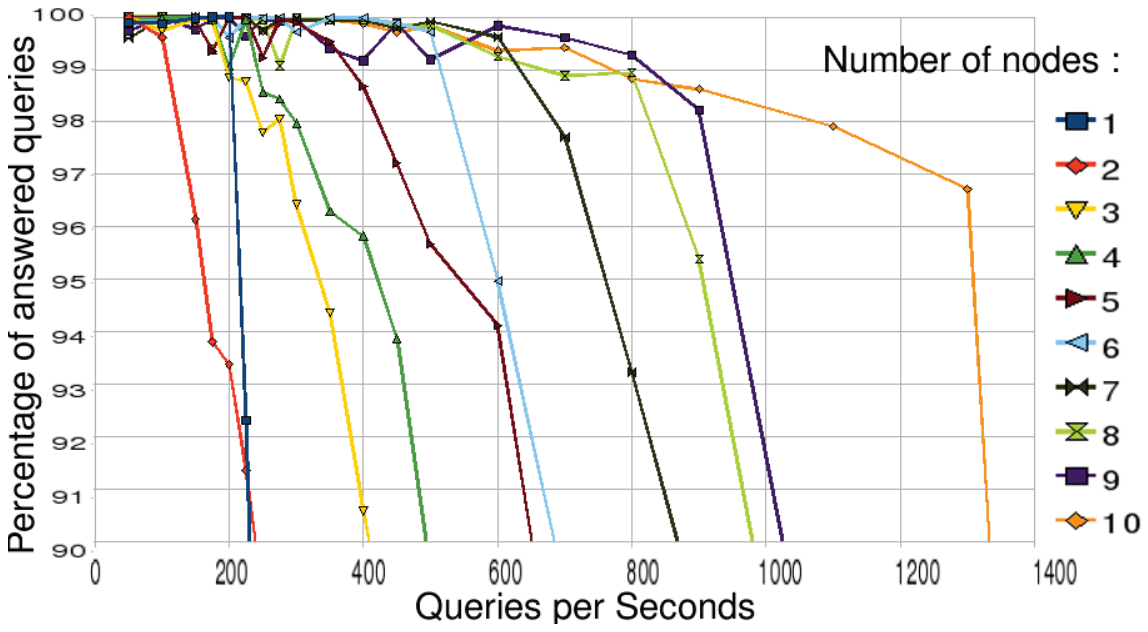
Another way to measure the dependence between the experimental and theoretical values is to derive the sample correlation coefficient as an estimator of the Pearson correlation. For both IP_{XOR} and *Pastry*, we consider the set of experimental measurements for Maximum Load: ML_{exp}, and the set of computed values for Maximum Load: ML_{model}. The various values are those measured and computed with various values of n the number of nodes. We derive $r^{IP}_{ML_{exp}ML_{model}} = 0.9991$ and $r^{Pastry}_{ML_{exp}ML_{model}} = 0.9827$. Correlation coefficients are very close to 1 which shows a perfect positive linear relationship between the measured values and those computed from our models.

As a result, experimental measurements shows our theoretical model does not present major bias, and confirms the stable ratio of $\frac{ML^{Pastry}}{ML^{IP}}(n)$ in the testing conditions. On the other hand it also shows that a FreePastry implementation may not fill the requirements of our models, and that further investigation would need specific developments, especially to simplify the routing code and algorithms.

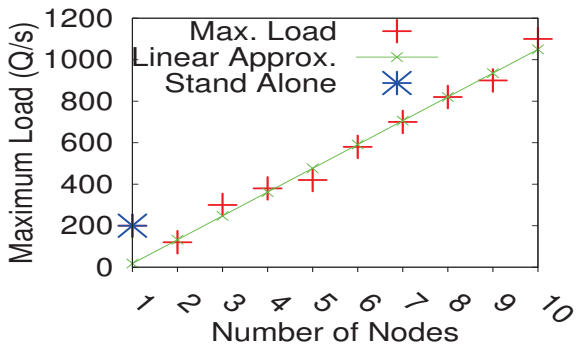
3.7 Conclusion

This chapter proposes the $PREFETCH_X$ architecture which prefetches X FQDNs ($HEAD_X$) and handles the remaining FQDNs ($TAIL_X$) with a Distributed Hash Table (DHT) structure. Traffic analysis sets $X = 2000$ to uniformly distribute the CPU on the nodes, whatever the Server implementation, the protocol DNS or DNSSEC is used. Prefetching $HEAD_X$ reduces the number of nodes by 2, and DHT reduces nodes for $TAIL_X$ between 55% and 80%, making $PREFETCH_X$ 4 times more efficient than the current IP_{XOR} architecture.

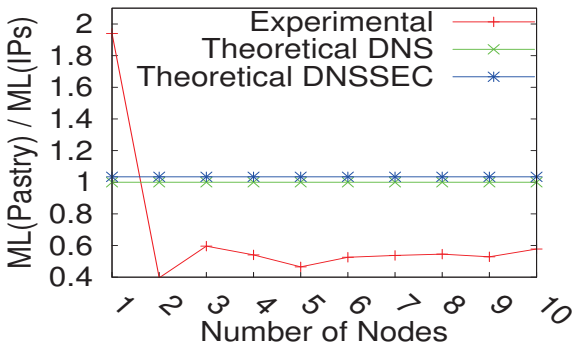
$PREFETCH_X$'s efficiency can be enhanced by increasing X, providing an adapted light *Pastry*-like implementation and reducing the Server's cache. In this chapter, we keep X small to limit the exchanges between the nodes to fill their cache. Large values for X require protocols and architectures optimized for cache updates. Similarly, a light *Pastry* implementation is also expected to enhance the architecture, especially if the redirection can be integrated into the multi-core Network Hardware Acceleration Card (NHAC). At last, DHT reduces the cache sizes on the nodes over IP_{XOR}, but reducing the number of nodes may counter this benefit. One may take advantage of multi-core, and start multiple independent (virtual) servers and take advantage of traffic analysis to define FQDNs that should not be cached. All these aspects are left for future work.



(A) Experimental Error



(B) Experimental Maximum Load



(C) Experimental vs Theoretical

FIGURE 3.9: Maximum Load

Part II

Providing Seamless Security for Multiple Interfaces

Introduction of Part 2

Description of the Work

The second part of the thesis is about Interactions between IPsec [KS05] *Mobility*, *Multihoming* and *Multiple Interfaces*. The first motivation for this part is to improve the End User experience for a secured communication. This means enhancing the current combination of IPsec and Mobility Protocols so to ease Mobility and add other Multiple Interfaces functionalities. Scenarios typically include an End User connected to its company network via a VPN, or protecting communications with another terminal or a specific service. Most likely, this user will be Mobile, Multihomed, and using Multiple Interfaces. Our goal is to optimize Security so the End User keeps on combining Security and ease of use. Then, Offload combines Security, Mobility, Multihoming and Multiple Interfaces and constituted the second motivation for this part. Mobile data traffic is expected to be around 50 times greater than it is today, and upgrading the current 3G / 4G infrastructure for that traffic represents too much investment compared to using Alternate Networks such as WLAN. WLAN requires *Mobility*, *Multihoming* and *Multiple Interfaces* agility, to improve the End Users Experience over unreliable WLAN networks. In fact as opposed to 3G / 4G Networks, WLAN are neither owned nor managed by ISPs and thus are considered as unreliable in term of availability. Furthermore, for the same reasons, WLAN cannot be trusted as 3G / 4G Networks. Securing Radio Access is not enough and higher layer Security is required. The most common way to secure a communication are: to secure the IP layer with IPsec [KS05], to secure the Transport Layer with TLS [DR08] or DTLS [Phe08], or that security is handled by the application itself. In the case of Offload, the Security depends on the Network used and the application may not have been designed with security. In other words, the same application using 3G / 4G RAN does not need to secure its communication, whereas it does on WLAN. Implementing application Security or TLS / DTLS needs to modify the application code, whereas IPsec does not. This is the reason this part of the thesis focuses on IPsec.

Mobility, *Multihoming* and *Multiple Interfaces* are quite intuitive for a Node, and can briefly be defined as follwos: Mobility, means a Node is changing Access Point. Multihoming means that a Node provides some IP addresses that may be used in case the currently running address is not reachable anymore. At last, Multiple Interfaces means that a Node can discover some Network Access Point be attached and get a new IP address and use it while still being connected to the previous IP addresses. Multiple protocols have been designed for that purpose, among these SCTP [OY02], SHIM6 [NB09], MPTCP [FRH+11].

IPsec has been designed to protect IP packets. IPsec defines for a given IP packet if it should

be DISCARDed, BYPASSed or PROTECTed, and in the latest case how the IP packets must be secured. In that sense, IPsec can be seen as a firewall whose configuration needs to be modified when IP addresses of the Mobile Node change. IPsec requires to be properly configured so to avoid blocking connections. This specific configuration is required because (1) IPsec is an independent layer between the IP and the Transport Layer, (2) IPsec can DISCARD IP packets, and (3) IPsec Security Rules are based on IP addresses.

MOBIKE [Ero06] is the IPsec Mobility and Multihoming extension. However, it has only been designed for the VPN with a Node that only has a single interface. In fact, at the time MOBIKE was designed, Mobile Nodes were labtops rather than Smartphones, and Mobility, Multihoming requirements were thus much different. More specifically, Mobility was expected to happen only occasionally, not as a regular base as it is with Smartphones. In addition, Multiple Interfaces was not considered, nor scenarios that require end-to-end security with a Server rather than with a Security Gateway. As a result, the second part of the thesis provides IPsec *Mobility*, *Multihoming* and *Multiple Interfaces*, so that IPsec can be appropriately configured when used with *Mobility*, *Multihoming* and *Multiple Interfaces* protocols like SCTP [OY02], SHIM6 [NB09], MPTCP [FRH$^+$11].

It is true, that IPsec needs its own protocol to keep communications secured in a Mobile, Multihomed and Multiple Interface environment, but does it mean that secure communications could not have been moved until now? Can't I really move my VPN? Is IPsec so independent from transport layer? Why choosing IPsec?

We do not assert ourselves as pioneers in the IPsec Mobility Multihomed and Multiple Interfaces Area. At first one should mention MOBIKE that has been designed for one specific use case: a client VPN connected to a Security Gateway is able to change its IP address, and provide Alternate IP addresses in case the current running IP address is down. Actually the VPN case is the most confusing example because, the use of the tunnel mode is using IPsec as a "Transport" protocol, similar to Mobile IP [JPA04]. In the case of the VPN, the VPN client tunnels a communication to the Security Gateway, and if the End User is changing its IP addresses, only the outer IP addresses of the Tunnel are modified. Changing the outer IP addresses of the IPsec Tunnel results in tunneling the ongoing communication to / from one IP address to another IP address without breaking the communication, thus performing a Mobility. In other words, in that specific case, IPsec is used both to protect a communication and to carry the data. Of course, if the communication would not have been tunneled, this would not work. The reasons we are looking for extending MOBIKE are (1) IPsec is not always a tunnel involving a Security Gateway, (2) we are interested in protecting communications with end-to-end security (IPsec Transport mode) and (3) Even for Tunnel mode, we want to provide IPsec Soft Handover so that Mobility be optimized for Mobile Nodes. These are the goals of our MOBIKE-X protocol that we designed and proposed to IETF standard.

Independence of the IPsec and transport layer is more obvious when IPsec is used without tunneling the data, that is to say using the Transport mode. With Transport mode, modifying the configuration does not result in any Mobility. We are interested in using Transport for End-to-End connectivity to avoid the Tunnel overhead, especially for Real Time Applications that are sensitive to large latencies. As a result, we are considering IPsec as an independent layer because we are not interested in the Tunnel mode only. Then we are also interested in Multiple Interfaces and the current VPN case only considers a single interface which simplifies the Interface to choose for sending the data and the possible configurations. As a result MOBIKE-X extends MOBIKE features to the IPsec Transport mode. In addition it also extends MOBIKE to Multiple Interfaces, thus adding to MOBIKE the possibility of performing Mobility Soft Handover instead of Hard Handover.

The second part is organized as follows: Chapter 4 presents the various protocols related to simultaneous support of security, Mobility, Multihoming and Multiple Interfaces. Its goal is to position and describe the IPsec protocol we designed: MOBIKE-X. More specifically, we start by defining Mobility Multihoming and Multiple Interfaces, and show how those concepts are handled by the SCTP Transport Protocol. Then we provide the IPsec Overview and we focus on Mobility Multihoming and Multiple Interfaces impacts over the IPsec Architecture. A brief description follows with the already existing IPsec protocol that could currently be used to handle Mobility Multihoming and Multiple Interfaces: IKEv2 and MOBIKE. Finally chapter 4 provides the use cases, the problem statement and a brief description of MOBIKE-X.

Chapter 5 is focusing on the performance measurements. This chapters describes the testing platform used, the different tests performed. This chapter aims at measuring the Mobility and Multihoming performances of MOBIKE-X versus MOBIKE, as well as the costs of IPsec security overhead. Costs are measured according to Time, and the major criteria we use are: $T_{STALLED}$ the time the connection is stalled and T_{SYS} the time it takes to the system to notice modification occurs on the Interfaces. To measure the cost of IPsec over a Mobility or Multihoming operation, we performed Mobility and Multihoming using SCTP with non IPsec-protected connections and with IPsec protected connections. Considering the Transport and the Tunnel IPsec modes, Transport reduces both the data to encrypt and the Network Complexity. In fact, encryption has a cost, but one should not underestimate the interruptions that occur in the Kernel as mentioned in section 6.3.4.

In fact when we compare SCTP Mobility over two IPsec connections mobility is 2.5 times more stable and 15% faster. This is mostly explained by the complexity introduced with the Tunnel mode. In fact in the Linux Kernel, the Tunnel mode requires IP packets to go through the IP stack twice. Note that those measurements did not introduce any IPsec configuration. The different involved IPsec connections are established before the SCTP Mobility occurs. The remaining measurements are dedicated to IPsec Mobility and we compare MOBIKE Mobility and Multihoming top MOBIKE-X's. MOBIKE is used for the traditional VPN, Mobility is performed by modifying the outer IP address of the Tunnel header. This is transparent to the communication that is tunneled, and so the communication does not need to handle mobility nor multihoming. In the tests we considered a regular TCP connection. On the other hand, with MOBIKE-X and the Transport mode, the communication that is protected with IPsec needs to handle Mobility and Multihoming in the same way as IPsec does. Others might say IPsec must handle Mobility and Multihoming as the transport protocol. Measuring with those two configurations $T_{STALLED}$, we found out that MOBIKE-X is between 9.3% and 15.6% faster than MOBIKE. This is probably due to the use of the Transport mode which requires fewer interactions with the routing indirections of the Kernel.

Chapter 6 analyses the Offload use cases. Current mobile data are carried through 3G / 4G also called Radio Access Network (RAN). Because the demand for mobile data is likely to be 50 times larger than it is today, and upgrading the RAN Infrastructure represent high costs for a flat rate subscription, ISPs are investigating on how using alternative Networks like WLAN. WLAN Access Points are very cheap, and most European ISPs can also take advantage of a WLAN Access provided by their DSL End Users. In fact, in Europe, ISPs provide their End User a DSL box with a set of services. WLAN Access can thus easily be deployed in major European cities. Furthermore, most of those ISPs have already deployed WLAN communities and DSL subscribers are provided with credentials to benefit from WLAN Access Point of other DSL subscribers. However, in this specific case, ISPs are providing a best effort service. In the case of offload, the ISPs are using WLAN but are expected to provide the same End User experience as on a RAN, with equivalent Services as those on the RAN. More specifically, the ISP is expected to provide the same confidentiality of the

communication for an offloaded communication as it would be the case on the RAN. Then, in term of End User experience, the ISP must provide equivalent connectivity with an unreliable WLAN Network. WLAN Network is claimed as unreliable since it may not be managed by the ISP.

In chapter 6, we provide different architectures for offloading traffic. The offload architecture should be chosen according to the ISPs' needs. One way consists of using a specific architecture dedicated to a service: Offload Service Architecture (OSA). Since this architecture is dedicated to a given service, the communication can use end-to-end security, and can take advantage of the IPsec Transport mode. A service may not have a dedicated offload architecture. One reason may be that the service does not take advantages of the enhancements provided by the use of Transport or end-to-end connectivity. Another reason may be that deploying multiple dedicated offload architectures for all services may cost too much, and ISPs want to mutualize an Offload Architecture for multiple services with the Offload Access Architecture (OAA). Unlike the OSA, OAA tunnels the End User traffic to a Trusted Entry Point in the ISP CORE Network. This Trusted Entry Point is a Security Gateway (SG), and communications are using the IPsec Tunnel mode. OAA secures an Access, and compared to OSA may add latencies because of routing indirections as well as encapsulation decapsulation operations performed by the SG. Then tunneling packets may also add network latencies.

The chapter compares OSA and OAA and provides a functional as well as a performance comparison. The performance comparison is mainly based on experimental measurements provided in chapter 5. Of course, traffic that does not require security must neither be protected by OSA nor by OAA. Comparing OSA and OAA does not mean one Architecture has to be chosen and the other rejected. In fact OSA and OAA address different cases and should be chosen, according to the Network and Service requirements.

This chapter also explains how to deploy OSA and OAA with limited costs deployment. In fact OSA and OAA use MOBIKE(-X) for Mobility, Multihoming and Multiple Interfaces facilities. On the other hand, we cannot rely on MOBIKE(-X) for moving a non IPsec protected communication on the regular 3G / 4G RAN to the OSA or OAA. The two challenges to overcome are: (1) how to move from RAN to OSA or OAA and (2) how to optimize IPsec negotiation so the communication is not blocked during the negotiation. We detail two solutions: one which uses SCTP combined with IPsec MOBIKE(X) and another one that only uses MOBIKE-X. Once the two deployment solutions are compared, we provide guidelines and recommendations for ISPs to offload their End User mobile traffic. First ISPs should split / identify the traffic that does not need protection and the traffic that requires protection. Then OSA should be dedicated to services with Real Time constraints. To ease OSA deployment, ISPs may start by deploying OSA with the Tunnel mode which makes MOBIKE(-X) deal with Mobility on the WLAN. To move from RAN to WLAN, MOBIKE may also be used at first. This would correspond to a first deployment version of the *ISP Offload Infrastructure*. For version 2, we recommend ISPs decide to port application to SCTP or to configure properly the session resumption mechanisms so that OSA can migrate from Tunnel to Transport mode and does not rely on MOBIKE for Mobility between RAN and WLAN. For version 3, we recommend to optimize MOBIKE(-X) so to perform Soft Handover. For version 4, we recommend ISP to focus on the RAN to WLAN optimization with MOBIKE-X for applications that are not ported to SCTP.

Notations & Abbreviations

In this part, we are using the following notations and abbreviations:

- AH: Authenticated Header
- CN: designates the Correspondent Node the MN is talking to.
- Communication: designates data exchange between the MN and the CN. Communication does not specify how the data are exchanged, that is to say how many IP addresses are

involved, if there are different flows...
- Connection: designates data exchanged through a TCP, SCTP MPTCP socket.
- DF: Don't Fragment
- DSCP: Differenciated Services Code Point
- E2E: End-to-End Security
- ESP: Encapsulating Security Payload
- EU: End User
- FWDA: Forwarding Architecture
- ISP: Internet Service Provider
- Interface: designates a hardware network Interface. In this thesis we consider that a MN can have multiple Interfaces, and each Interface is assigned a single IP address. We do not consider the subtleties of virtual Interfaces, mainly for clarification reason. As a result IP addresses and Interfaces can be exchanged
- MM: Mobility Multihoming & Multiple Interfaces
- MN: designates a Mobile Multihomed and Multiple Interface Node.
- MNO: Mobile Network Operator
- MTU: Maximum Transmission Unit
- OAA: Offload Access Architecture
- OSA: Offload Service Architecture
- PAD: Peer Authorization Database
- PFP: Populated From Packet
- RAN: Radio Access Network
- RTA: Real Time Applications
- SA: Security Association
- SAD: Security Association Database
- SG: designates a Security Gateway. The MN can be connected to the CN directly, or via an SG.
- SP: Security Policies
- SPD: Security Policy Database
- SPI: Security Policy Index
- VPN: Virtual Private Network
- WLAN: Wireless LAN

Chapter 4

Issues and Protocols Relative to Simultaneous Support of Security, Mobility, Multihoming and Multiple Interfaces

4.1 Introduction

This chapter presents an overview of IPsec and how Mobility Multihoming and Multiple Interfaces impacts IPsec configuration. Then, we present MOBIKE the currently defined protocol that deals with IPsec Mobility and Multihoming. MOBIKE presents restrictions leveraged by MOBIKE-X the protocol we designed, implemented, tested and proposed to IETF for standardization.

This chapter is organized as follows: Section 4.1.1 starts by defining the concepts this chapter is dealing with: Mobility, Multihoming and Multiple Interfaces. This section identifies their meaning regarding a transport layer like SCTP and then regarding IPsec, and briefly describes SCTP which is the transport protocol used to evaluate MOBIKE-X. The remaining sections are then exclusively focused on IPsec protocols.

Section 4.2 describes the IPsec Architecture, that is IPsec principles, the different IPsec components and protocols. Section 4.3 focuses on the IPsec databases and section 4.4 shows how Mobility, Multihoming and Multiple Interfaces impact these IPsec Databases. Once the impacts on IPsec databases have been clarified, this chapter presents the various IPsec protocols that have been designed for setting up IPsec configuration between two Nodes. Section 4.5 presents IKEv2 [KHNE10] the protocol designed for negotiating IPsec parameters. This section especially describes how IKEv2 is able to create a new Security Association and how IKEv2 can delete a Security Association. This could be useful for example if a Node wants to add a new Interface or remove an Interface that is not longer reachable. Then Section 4.6 presents MOBIKE [Ero06], the protocol designed for Mobility and Multihoming. This protocol has been designed only for the VPN scenario and thus partly addresses our requirements. Finally, section 4.7 presents MOBIKE-X. From the use cases, it shows that neither IKEv2 nor MOBIKE addresses them. Then it provides guide lines on how MOBIKE-X is designed, based on the already existing MOBIKE and it gives a brief description of the MOBIKE-X exchanges. The goal of this chapter is to design MOBIKE-X, the MOBIKE Extension for Mobility Multihoming and Multiple Interfaces. The content of this chapter provided input on book chapters for [JMDJ+07, JMDJ+09], Contributions to a Research

Project supported by the French National Agency for Research [DMM+09], paper submitted to a conference [MPL] and Internet drafts [Dan09b, Dan09a, DC12].

4.1.1 Mobility, Multihoming and Multiple Interfaces Definition

This thesis focuses on interactions between IPsec [KS05] and Mobility, Multihoming and Multiple Interfaces. This chapter defines Mobility, Multihoming and Multiple Interfaces. Each definition especially focuses on their meaning toward IPsec.

Mobility consists in changing an IP address for a given communication. It can be done through Hard Handover or Soft Handover. (1) Hard Handover consists in updating the current IP address of the Mobile Node with the newly acquired IP address. Hard Handover Mobility is performed by Mobile Nodes with a single interface and always results in interrupting the connection —at least the time it takes to modify the IP address. More specifically, if the Mobile Node has a single interface, it will be disconnected during the change of IP address. Note that most of the protocols —like TCP or SCTP —are not resilient to an IP address update, and this will break the transport layer. (2) Soft Handover consists in changing the IP address by avoiding this interruption. The Mobile Node needs two Interfaces, and while the old IP address remains active on one interface, the other becomes active. When the new IP address gets all the traffic, then the Mobile Node removes the old IP address. As a result, Soft Handover keeps the Mobile Node connected, and at least reduces the impact of Mobility compared to Hard Handover.

Multihoming prevents failover of an ongoing communication. We call *Primary Interface*, the used interface of the Mobile Node, and *Alternate Interfaces*, the interfaces that should be used in case the *Primary Interface* is not reachable any more. *Multihoming* differs from Mobility Hard Handover because Mobile Node provides the Correspondent Node a list of *Alternate IP addresses* which makes the Correspondent Node (CN) able to handle the Mobility. When the Mobile Node is no longer reachable, the Correspondent Node may attempt to reach it through the *Alternate Interfaces*. Note that Multihoming includes a detection delay, and connection managers are expected to perform Mobility before the Primary Interface is down.

A Node has *Multiple Interfaces* if it can be attached to Multiple Access Points. It does not necessarily refer to a Mobile Node, but in this thesis, we consider it as a Mobile Node, which makes dynamic configuration properties more obvious. Suppose the Mobile Node has a given communication with a Correspondent Node, the Mobile Node is expected to be able to add the Interface for the given communication. For a communication adding an Interface can be understood as load balancing the traffic between the two interfaces. Load balancing can be done by splitting connections between the two Interfaces, or by splitting a single connection between the two Interfaces. Note that in the first case each connection can be Multihomed and use the other IP address as an alternate IP. Similarly, a Mobile Node with Multiples Interface must also be able to remove an Interface.

4.1.2 SCTP for Mobility Support

SCTP [Ste07] is a transport protocol designed at the IETF. This protocol has been primarily designed to provide reliable transfer and multi-stream communications to avoid the head-of-line blocking. In our case, we use SCTP [Ste07] because 1) it provides Mobility and Multihoming features for the Mobile Node without requiring the ISP to deploy any infrastructure. Furthermore 2) SCTP is the more mature Mobility and Multihoming protocol with Kernel and User Land im-

plementations, which ease the deployment of SCTP either within the terminal's Operating System or integrated in the applications.

With Multihoming, SCTP can initiate a communication with multiple IP addresses - one Primary and one or multiple Alternate. In case the Primary IP address is not reachable, the communication switches to the Alternate. SCTP Mobility features are based on Multihoming and Dynamic Address Reconfiguration [SXT+07] and makes Soft Handover possible, as shown in figure 4.1. [SXT+07] makes possible the ADDition of a new IP address as an Alternate IP address —step 2 in figure 4.1 —, as well as the change of the IP address status —step 3 in figure 4.1. Hence, Soft Handover [RT07] consists in changing an Alternate IP address that is expected to receive the traffic in a Primary as illustrated in figure 4.1. The MN has established a SCTP association with the CN. The MN is using IP^1_{MN} and IP^2_{MN} and the CN is using IP_{CN}. In step 1, the MN acquires a new IP address IP^*_{MN} and adds this IP address to the SCTP association. IP^*_{MN} is considered as an Alternate IP address. In step 2, the MN agrees with the CN to use IP^*_{MN} as the Primary IP address, and that IP^1_{MN} is used as an Alternate IP address. In step 3, the MN is no longer reachable on IP^1_{MN} and removes this IP address from the SCTP association. Note that SCTP cannot perform Hard Handover, and Mobility with SCTP requires at least two Interfaces.

With SCTP Kernel based *LKSCTP* [LKS] and user land (*sctplib* [sct]) implementations, ISPs can provide terminal with SCTP enable kernels, and for terminals that are non SCTP enable —either old terminal, or terminal provided by other ISPs —ISPs can provide applications that are SCTP enable. Note that porting TCP applications to SCTP is quite easy with tools such as *withsctp* from *lksctp-tools*.

Although SCTP does not present major deployment issues, there are still few complications to overcome. First ISP must maintain different TCP and SCTP application versions, then some applications like FTP interact closely with IP address, and so cannot easily rely on the SCTP Mobility and Multihoming facilities. At last NAT, firewalls proxies have not yet been configured for SCTP.

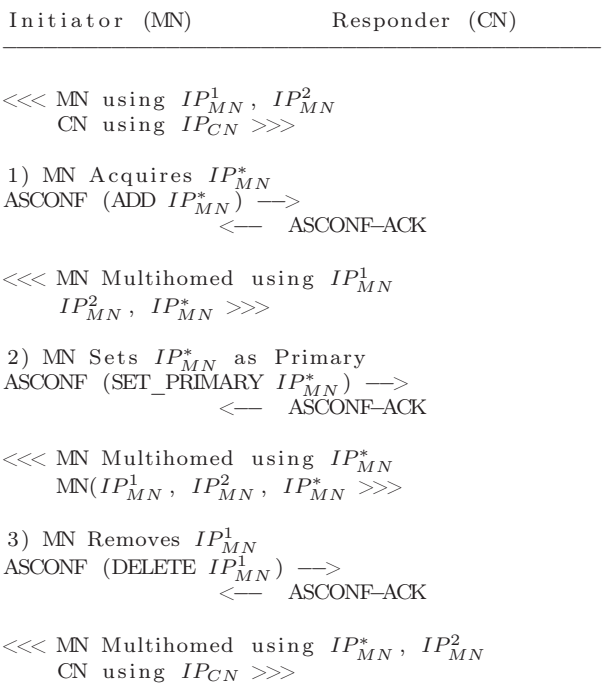


FIGURE 4.1: Messages Exchanges for SCTP Soft Handover

4.2 IPsec Overview

As presented in figure 4.2 IPsec makes possible communication between two Trusted Environment through an Untrusted Network. IPsec [KS05] secures communications at the IP layer, by encrypting or authenticating each IP packet. IPsec is standardized by the IETF and includes an architecture description [KS05], as well as a suite of protocols IPsec is a suite of protocols that includes Encapsulating Security Payload (ESP) protocol [Ken05b] mostly used for encryption, Authenticated Header (AH) protocol [Ken05a] mostly used for authentication, and Internet Key Exchange (IKEv2) protocol [KHNE10] used by both peers to agree on the cryptographic material.

Security Policies define how IPsec handles the traffic and are stored in the Security Policy Database (SPD). SPD defines if a packet must be DISCARDed, BYPASSed or PROTECTed. As presented in figure 4.2, for each outgoing packet, IPsec checks the SP in the SPD and determines whether the IP packet must be dropped, sent, without protection or protected. When the packet needs to be protected, cryptographic material is stored in Security Association (SA) in the Security Association Database (SAD). If the SA has not been established yet, then an IKEv2 negotiation is initiated to set the SA —IKEv2 is triggered with the first packet. For inbound IPsec protected packet, the packet is decrypted using cryptographic material of the SAD, before checking the decrypted packet has been protected according to the proper SP. In case no SA is found or there is no SP matching, the packet is discarded. Other non IPsec inbound packets are BYPASSed or DISCARDed according to the SPD. The PAD is essentially used by IKEv2 during the authentication.

IPsec databases lookup is complex because (1) lookup depends on the traffic and (2) SPD and SAD are not central databases, but rather optimized for handling the traffic. Firstly, IPsec databases lookup is processed differently with inbound and outbound traffic, then lookup is performed differently for protected and non protected inbound traffic. Secondly, SAD and SPD are not central databases. First SAD and SPD are two distinct databases, but there are highly dependent from each other. As detailed in section 4.3.1, SPD can be cached and decorrelated. When we specify cache SPD, we mean a SPD that is stored in the kernel and that can be expressed with packet level information. This is opposed to a SPD where SPs are expressed at a higher level. More specifically, a SPD can have SP such as "*packet exchanged with users of company.com*", whereas the SPD cache only has SP for (IP_{source}, $IP_{destination}$, $Port_{source}$, $Port_{destination}$). A decorrelated SPD is a SPD split for IPsec Secured traffic (SPD-S), Inbound non-protected traffic (SPD-I) and Outbound non-protected traffic (SPD-O). On the other hand, SAD is only cached in the Kernel, and indexed with SPI or eventually IP addresses that can be read from the inbound secured packets.

Note that this thesis only considers the case where IP packets are protected with ESP or AH, but not a combination of those protocols. Combination of the two protocols makes the IPsec packet go twice through the IPsec process. This is what [KS05] refers as "nested SA" in section 5.1. This only reason is that combination would make explanations more complex.

Section 4.3 provides a more in depth description of the various database, to point out *Mobility*, *Multihoming* and *Multiple Interface* related issues.

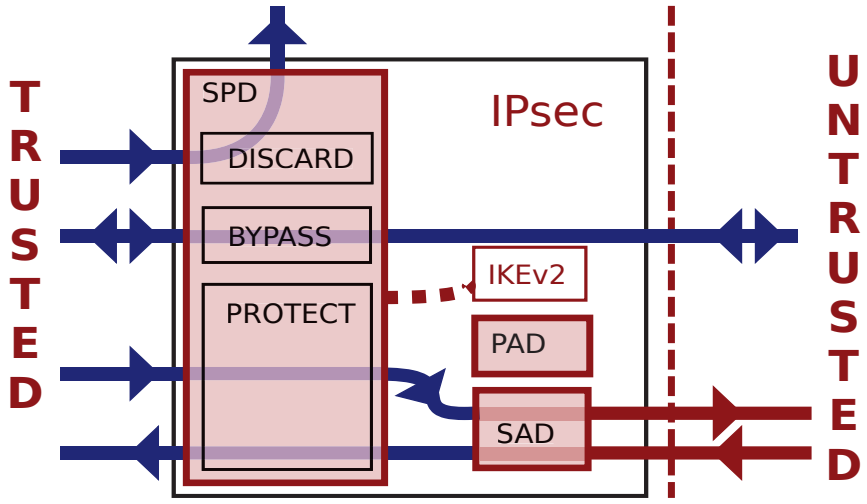


FIGURE 4.2: Diagram of IPsec Principles

4.3 IPsec Databases

As presented in figure 4.2, IPsec deals with traffic thanks to three different Databases: *Security Policy Database* (SPD), *Security Association Database* (SAD) and the *Peer Authorization Database* (PAD). This section provides a more detailed description of those Databases, so that we can expose clearly in section 4.4, how *Mobility*, *Multihoming* and *Multiple Interfaces* impact the IPsec configuration.

The three IPsec Databases are:

- Security Policy Database (SPD): contains a high level description of the Security Policies (SP).
- Security Association Database (SAD): contains the Security Association (SA) that hosts the cryptographic material, and the necessary information to encrypt and decrypt IPsec protected IP packets.
- Peer Authorization Database (PAD): contains information on who can communicate with the IPsec layer and how IKEv2 must proceed to the authentication. The PAD should be seen as a meta database that provides inputs for IKEv2 to configure the Security Policies.

4.3.1 Security Policy Database

The SPD contains a high level description of the Security Policies to apply. We call this high level, because the description is intended to be generic and more or less human readable. We oppose, in this section, the high level SPD and the SPD cache where SP are indexed only with information read from the packet.

The SPD Cache is derived from the SPD, stored in the Kernel whereas the SPD may be stored at the application layer. More specifically, a given IP packet with IP addresses and Ports IP_{src}, IP_{dst}, P_{src}, P_{dst} will have an exact match in the SPD Cache with eventually some wildcards (ANY/OPAQUE) to specify that some values must not be looked at or that any value matches. This is not the case with SP expressed in the SPD (not the SPD cache). A Security Policy in

the SPD may be "*Initiator authenticated from company.com*" or "*Packet with a this protocol*" or "*Packet issued from a subnetwork*". In the first case, establishing the SPD Cache entry requires interactions with IKEv2. IKEv2 proceeds to the authentication and configures the SPD cache before deriving the SP in the SPD Cache. Deriving such SP could not have been done by reading information from the packet. In addition, the two other cases exhibit that the SPD must also specify how the SP of the SPD Cache must be generated. If SPD generates a single SP associated to a unique SA, then, all IP addresses would share the same cryptographic keys, meaning every one shares the same "secured" channel. This is not what we are expecting! What we are expecting is that each IP address is being assigned its own secured channel, that is to say its own cryptographic keys. This section details how to generate such SPD Cache entries. The SPD Cache indexes its SP thanks to Traffic Selectors that can be the Source / Destination IP Address, the Next Layer Protocol, Source / Destination Port or, —when no port is available —the ICMP type / code or Mobility Header.

For outbound traffic, SPD cache lookup is much faster than SPD lookup, so SPD lookup is generally performed when a SPD Cache miss occurs. Unless specified, and as represented in figure 4.2, SPD lookup means SPD Cache lookup followed by high level SPD lookup. This is also a reason the systems may create an SPD Cache entry with a DISCARD action. A packet that is not explicitly BYPASSed or PROTECTed should be DISCARDed as a default rule. However, to define the default rule, we need to perform a SPD lookup. Note that for inbound traffic the SAD is used instead of the SPD Cache (cf. section 4.3.2). The remaining of this section specifies how SPD Cache is generated.

A SPD entry associates Traffic Selectors to a processing action. This traffic is not protected, and the SPD defines:

- 1. if it must be protected or not,
- 2. if protection is required, then how it must be protected and
- 3. how to derive the SPD Cache Entry and the Security Association (SA).

Note that to understand in an easier way how the SPD works, it is recommended to consider in the first time, the outbound traffic.

- 1. **Needs to be secured?** To define if the IP packet must be protected or not, the SP uses PROTECT, BYPASS or DISCARD to designate the processing action.
- 2. **How to secure?** To define how the packet must be protected, the SP associates to a PROTECT processing action, information like IPsec mode (Transport or Tunnel), security protocol(s) (AH and/or ESP), encryption algorithms. If the Tunnel mode is used the outer IP addresses of the Tunnel are also specified. [KS05] section 4.4.1.3. provides an explicit list of the SP associated information.
- 3. **How to derive the SA?** The SPD Cache and associated SAs are derived from the high level SPD thanks to the Populated From Packet (PFP) Flag. This flag indicates how the Security Policy must be indexed in the SPD Cache, that is to say whether the SP must be indexed considering the Traffic Selector value provided by the packet or not. Table 4.1 illustrates the case of a security Gateway that applies the same SP to all the users of a given company, but still wants that each member has its own Secure Tunnel, that is to say their own SPD Cache Entry and their own SA. For that purpose, the SPD indicates that any packet must be protected by setting the ANY value to all Traffic Selectors of the SPD entry. To mention that the SP in the SPD Cache is only associated to the source IP address of the packet, and that any different source IP address must have its own SP, the SPD sets the PFP of the source IP traffic Selector. Any new inner source IP address should lead to a new SPD Cache

/ SAD entry. The SP is indexed with IP_{src}, the source IP address of the packet and ANY for the remaining index components (destination IP address, source port and destination port). As result, any packet sent by the member of the company has IP_{src} as the source IP address, IP_{dst} for the destination IP address, P_{src} and P_{dst} for the source and destination port. (IP_{src}, IP_{dst}, P_{src}, P_{dst}) matches the index (IP_{src}, ANY, ANY, ANY) of the SP, and the packet can be encrypted. Similarly, for inbound traffic the Selectors are (IP_{src}, ANY, ANY, ANY) provided in the SA, which makes possible the SPD check. Remember that SAs are unidirectional and negotiated by pair.

Note that as mentioned earlier, high level SPD can be expressed according to various Traffic Selectors, with different granularity, different level of abstraction. This may result in multiple matches for a given packet. To avoid controversial interpretation, the SPD is an ordered database, and the first is considered. As detailed later in section 4.4, this may provide a restriction for Multiple Interfaces Node, that may not perform load balancing between two Interface for a given connection.

Note that applying [KS05] on Linux, the SPD is mostly managed at the application layer by IKEv2 for example. IKEv2 defines what are the Traffic Selectors and Security Associations and then pushes them in the Kernel. In that sense, the Linux Kernel only hosts an SPD Cache.

Packet Traffic Selectors	SPD Entry	PFP	Value in Triggering Packet	SPD Cache index / SAD Selectors
Name	FQDN	0	-	-
IP_{dst}	ANY	0	Destination IP address	ANY
IP_{src}	ANY	1	Source IP address IP_{src}	IP_{src}
P_{dst}	ANY	0	Destination Port	ANY
P_{src}	ANY	0	Source Port	ANY

TABLE 4.1: Deriving SPD Cache Traffic Selectors from Packet Traffic Selector

4.3.2 Security Association Database

Security Associations (SA) contain information relative to:

- 1. **Cryptographic material and processing information:** that is to say the different keys and protocols for authentication and encryption, and when the Tunnel mode is used, the outer IP addresses. First the SA specifies which IPsec mode is used (Transport/ Tunnel). In case the Tunnel mode is used, the outer IP addresses of the Tunnel are provided. The SA also specifies if AH is used, and if so, which authentication protocol, as well as its associated keys. Similarly the SA specifies if ESP is used as well as protocols and associated keys for authentication and encryption. If AH and ESP are used simultaneously in a combined mode, the SA specifies authentication and encryption keys and protocol. Finally, the SA also contains the life Time of the SA.

- 2. **IPsec signaling:** includes SPI, Sequence Number Counter used in the AH or ESP header, Sequence Counter overflow indicates if roll over is permitted, the Anti Replay Window which prevents from replay attacks.

- 3. **Additional checking information:** includes Stateful fragment checking flag, bypass Don't Fragment (DF) bit, Differentiated Services Code Point (DSCP) values and path Maximum

Transmission Unit (MTU).

- 4. **Traffic Selectors used in the SPD Cache:** The SAD also contains the Traffic Selectors used by the SPD Cache. It serves for inbound packets, as represented in figure 4.2. SAD lookup is performed according to the SPI and optionally the IP addresses. The IPsec protected packet is decrypted, according to the information provided by the SA. Then, the Traffic Selectors are used for checking that the decrypted packet has been encrypted according to the appropriate Security Policies.

SAs are indexed with IP Addresses and the Security Parameter Index (SPI). The SAD lookup is performed first by looking for a match of IP source Address, IP destination Address and SPI, if no match occurs, then a lookup is performed with IP destination Address and SPI, and then at last with the SPI only.

4.3.3 Peer Authorization Database

The Peer Authorization Data base (PAD) is a link between IKEv2 and the SPD. IKEv2 authenticates the peer according to an Identifier —a raw Key, a FQDN, email address —and an authentication method. Then once authenticated, the identifier may be used in the SPD.

4.4 Mobility Multihoming and Multiple Interfaces impacts on IPsec

Section 4.2 and section 4.3 show that IPsec processing is highly dependent on the IP addresses, first to index the SP, then to check the SAD. *Mobility*, *Multihoming* and *Multiple Interfaces* result in modifications of IP addresses associated to a communication. If a Mobile Node wants to take advantage of *Mobility*, *Multihoming* and *Multiple Interfaces* facilities with an IPsec secured communication, then IPsec databases must be appropriately configured. The goal of this section is to show how SAD and SPD are impacted by such operations.

In this section, we consider, as represented in figure 4.3, that a Mobile Node (MN) has established an IPsec protected communication with a Correspondent Node (CN). The communication can be protected by tunneling the communication to a Security Gateway (SG) with an IPsec protected tunnel. It can also be protected with End-to-End Security between the MN and the CN. In that case, IPsec Transport or Tunnel can be used but we will consider Transport mode for End-to-End Security and Tunnel mode with the SG. The MN and CN are using respectively (IP_{MN}, P_{MN}) (resp. (IP_{CN}, P_{CN})) as IP addresses and Ports. The new Interface used by the MN is noted IP^{*}_{MN}. When the Security Gateway is involved, the IP address of the Security Gateway is noted IP_{SG}, IP_{MN} is used as the inner IP address and IP^{o}_{MN} the outer IP address. We note $ENCR = ENCR\_3DES$ (resp. $AUTH = AUTH\_HMAC\_SHA1\_96$) the encryption (resp. authentication) algorithms used to secure the IPsec packet.

4.4.1 Initial Configuration at the MN

Figure 4.3 presents the initial network configuration and table 4.2 the corresponding SAD and SPD configurations. The SPI is highlighted because it is used by the SP as a pointer to the proper SA.

(Note that the implementation uses other types of links like pointers).

Figure 4.3a, illustrates the case where the communication between the MN and the CN is protected with End-to-End (E2E) security, that is from the MN to the CN. In that case the communication is protected with the IPsec Transport mode. The communication to be protected is the one between IP_{MN} and IP_{CN} which defines the SPD Selectors. The encrypted packet is sent on the network using IP_{MN} and IP_{CN} as well so, they are also used to index the SA. Table 4.2a provides the corresponding SA and SP.

Figure 4.3b illustrates the case where the MN tunnel its traffic with the CN to a SG. For packets coming from the MN, the SG decapsulates and forwards them to the CN. Packet coming to the MN are encapsulated by the SG before being forwarded to the MN. In that case, the communication is protected from the MN to the SG. Similarly to figure 4.3a, the communication to protect is the one between IP_{MN} and IP_{CN} which defines the SPD Selectors. Contrary to figure 4.3a, the encrypted packet is using IP^{o}_{MN} and IP_{SG}, that is to say the outer IP addresses of the tunnel. Hence, the outer IP address of the Tunnel IP^{o}_{MN} and IP_{SG} are used to index the SA. Table 4.2b provides the corresponding SA and SP, and makes possible configuration comparison between the Transport and Tunnel mode.

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN} IP_{src} : IP_{MN} P_{src} : ANY P_{dst} : ANY	PROTECT Mode : Transport Proto. : ESP Algo. : ENCR, AUTH SPI

SAD (Outbound Traffic)	
SPI : SPI IP_{src} : IP_{MN} IP_{dst} : IP_{CN}	Crypto. : K_e, K_a Counters: C_{esp}, C_w SPD Selectors IP_{src} : IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY

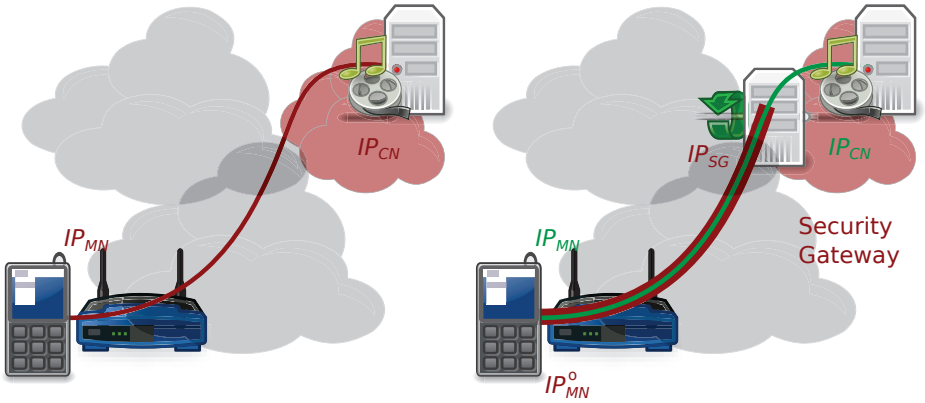
(A) ESP Transport

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN} IP_{src} : IP_{MN} P_{dst} : ANY P_{src} : ANY	PROTECT Mode : Tunnel IP^{Tunnel}_{src} : IP_{MN} IP^{Tunnel}_{dst} : IP_{SG} Proto. : ESP Algo. : ENCR, AUTH SPI

SAD (Outbound Traffic)	
SPI : SPI IP_{src} : IP^{o}_{MN} IP_{dst} : IP_{SG}	Crypto. : K_e, K_a Counters: C_{esp}, C_w IP^{Tunnel}_{src} : IP^{o}_{MN} IP^{Tunnel}_{dst} : IP_{SG} SPD Selectors IP_{src} : IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY

(B) ESP Tunnel

TABLE 4.2: SPD & SAD MN IPsec Configuration: Initial Configuration

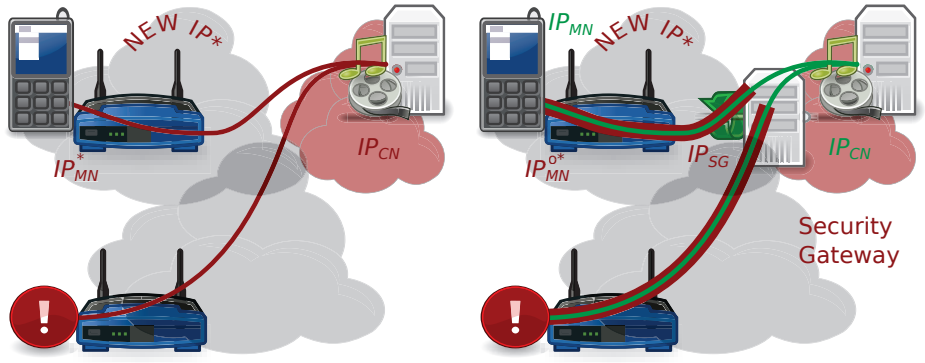


(A) ESP Transport (B) ESP Tunnel

FIGURE 4.3: Network Initial Configuration for IPsec

4.4.2 Mobility

With Mobility, as represented in figure 4.4 a new IP address IP^* is available, and the MN uses this new IP address instead of the previous IP_{MN}. This section specifies how IPsec must be configured so the same IPsec protected communication can use IP^*_{MN} instead of IP_{MN}. With the Transport mode, both SA Selectors and SP Traffic Selectors must be updated (cf table 4.3a). With the Tunnel mode, only the outer IP address is modified. Thus SP Traffic Selectors are not modified but only the SA Selectors as well as the SA and SP Tunnel IP addresses information (cf. table 4.3b) are. SA / SP modifications during the Hard Handover Mobility are indicated with a light grey background. This is what occurs with MOBIKE [Ero06] as described in section 4.6.



(A) ESP Transport (B) ESP Tunnel

FIGURE 4.4: Description of IPsec Mobility with Transport and Tunnel mode

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN}	PROTECT
IP_{src} IP^{*}_{MN}	Mode : Transport
P_{dst} : ANY	Proto. : ESP
:	Algo. : ENCR, AUTH
	SPI

SAD (Outbound Traffic)	
SPI : SPI	Crypto. : K
IP_{dst} : IP_{CN}	Counters: C_{esp}, C_w
IP_{src} IP^{*}_{MN}	SPD Selectors
	IP_{src} IP^{*}_{MN}
	IP_{dst} : IP_{CN}
	P_{src} : ANY
	P_{dst} : ANY

(A) ESP Transport

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN}	PROTECT
IP_{src} : IP_{MN}	Mode : Tunnel
P_{dst} : ANY	IP^{Tunnel}_{src} IP^{*}_{MN}
P_{src} : ANY	IP^{Tunnel}_{dst} : IP_{SG}
	Proto. : ESP
	Algo. : ENCR, AUTH
	SPI

SAD (Outbound Traffic)	
SPI : SPI	Crypto. : K_e, K_a
IP_{dst} : IP_{SG}	Counters: C_{esp}, C_w
IP_{src} IP^{*}_{MN}	IP^{Tunnel}_{src} : IP_{MN}
	IP^{Tunnel}_{dst} : IP_{SG}
	SPD Selectors
	IP_{src} : IP_{MN}
	IP_{dst} : IP_{CN}
	P_{src} : ANY
	P_{dst} : ANY

(B) ESP Tunnel

TABLE 4.3: SPD & SAD MN IPsec Configuration: Impact of Mobility

4.4.3 Multihoming

Multihoming works like Mobility for the IPsec Databases. The difference with Mobility, is that MN sends CN (resp. SG) some Alternate IP addresses in the case of Transport (resp. Tunnel), where the MN may be reached if the MN is no longer reachable on the current IP address. Thus before the Mobility occurs, the CN (resp. the SG) may check the MN is reachable on its Alternate IP addresses. For example, MOBIKE [Ero06] defines a Return Routability Check exchange with COOKIE2 Notify Payload for that purpose. If the MN is reachable on the Alternate IP address, then the CN (resp. SG) updates the IPsec Databases. If the MN performs the update, Alternate IP addresses are not used and a Mobility Hard Handover is performed as described in section 4.4.2. Note that the Return Routability Check is optional, and some protocols like SCTP triggers the update by directly using the Alternate IP address.

4.4.4 Multiple Interfaces

Contrary to Mobility and Multihoming, adding an interface cannot be handled by updating existing SP / SA, but, most of the time requires creating a new SP / SA. Suppose MN is connected with the Transport mode to CN, using IP_{MN}. When the MN acquires IP^{*}_{MN}, it expects to use both IP_{MN} and IP^{*}_{MN}. Similarly, if the MN tunnels its traffic to a SG using IP^{o}_{MN}, when it acquires $IP^{o}_{MN}*$, it expects to tunnel the communication to CN to the SG using both IP^{o}_{MN} and $IP^{o}_{MN}*$. This section details for Transport and Tunnel mode, why a new SA and a new SP need to be created when the MN adds a new interface.

Suppose the MN uses the Transport mode to secure its communication to the CN. The MN is expected to send packet using IP_{MN} and IP^{*}_{MN}. One way consists of updating the SP index so a match occurs. SP indexes do not accept lists of IP addresses, so there is no possibilities to add IP_{MN} and IP^{*}_{MN}. However, SP has wildcards like ANY or OPAQUE, but ANY is not equivalent to IP_{MN} and IP^{*}_{MN}. As a result adding an Interface with the Transport mode requires the MN to add a new SP. For inbound traffic, SA are indexed with the IP addresses and the SPI, SAD

lookup can be done with the SPI only. However, once the packet has been decrypted, the packet is checked with the Selectors of the SP. SP Selectors in the SA cannot be provided with lists, and so adding an interface with the Transport mode requires the MN to add a new SA.

Suppose now the MN uses the Tunnel mode. Communication to the CN is tunneled to the SG, and the MN wants to reach the SG by tunneling both with IP^o_{MN} and $IP^o_{MN}*$. With the Tunnel mode, the SP index is not modified. However, the SP contains information on which IP address to use for the Tunnel header. The MN has two different possibilities, and the SP cannot contain a list of IP addresses. Note that if a list of IP addresses could be used, then we assume the SP would be associated to the same SA for both IP^o_{MN} and $IP^o_{MN}*$, which also means the SA would be able to handle a list of IP address for the Tunnel header. Although not completely impossible, it is hardly possible adding a new interface would not require the creation of a new SP. For inbound traffic, the SA is indexed with the outbound IP addresses, and the SPI. The SPI is sufficient to perform the SAD lookup, but the SA cannot have a list of IP addresses for the Tunnel header. SP Selectors in the SA cannot be provided with lists, and so adding an interface with the Tunnel mode requires the MN to add a new SA.

Note that we do not mean that it is completely infeasible to introduce a list of IP addresses rather than a single IP address for SPD Selectors and SA parameters. However, we found out that creating a new SA is the easier way to deal with Multiple Interfaces and all IPsec implementations. In addition it makes possible to configure properly the SAD indexes and provide matches with IP addresses and SPI. The important thing to note is that the new SP / SA can be derived from the existing SP / SA, which avoids creating a SP / SA from scratch, an dmakes this operation even easier.

Figure 4.5 and table 4.4 illustrate the SPD and SAD for both Transport and Tunnel mode. With the Transport mode, MN uses either IP_{MN} or IP^*_{MN} to communicate with the CN. The newly created SP (resp. SA) has different Traffic Selectors (resp. Selectors).

Figure 4.5 and table 4.4b show with the Tunnel mode how using the same Traffic Selectors impacts the SPD / SAD. This may generate conflicts. As mentioned in section 4.3.1, the SPD is an ordered Database, and only one SP will be selected for the given Traffic Selectors —the first SPD match. As a result outbound traffic only uses the same SP. On the other hand, there are two SAs with different Selectors, so the MN is able to receive inbound traffic on both Interfaces. This is very convenient for Soft Handover —especially when transport protocols like TCP or UDP are not able to deal with multiple IP addresses. The MN appends the SP corresponding to the new Interface IP^*_{MN} to the SPD, then re-order the SPD. The new SPD tunnels the traffic through IP^*_{MN}, but still accepts the remaining traffic on IP_{MN}. However, if the MN wants to simultaneously use for a given SP Selectors both IP_{MN} and IP^*_{MN}, then the SPD must be indexed per Interfaces. Protocols that require such a decorrelated SPD are protocols that have been designed for Multiple Interfaces, —SCTP or MPTCP —for example to perform bandwidth aggregation. Such protocols are likely to handle Soft Handover, and doe not require IPsec to perform it on their behalf. In fact, decorrelating the SPD per Interface prevents IPsec to perform Soft Handover.

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN} IP_{src} : IP_{MN} P_{dst} : ANY P_{src} : ANY	PROTECT Mode : Transport Proto. : ESP Algo. : ENCR, AUTH SPI
IP_{dst} : IP_{CN} IP_{src} IP^{*}_{MN} P_{dst} : ANY P_{src} : ANY	PROTECT Mode : Transport Proto. : ESP Algo. : ENCR, AUTH SPI^{*}

SAD (Outbound Traffic)	
SPI : SPI IP_{dst} : IP_{CN} IP_{src} : IP_{MN}	Crypto. : K_e, K_a Counters: C_{esp}, C_w SPD Selectors IP_{src} : IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY
SPI SPI^{*} IP_{dst} : IP_{CN} IP_{src} IP^{*}_{MN}	Crypto. : K_e, K_a Counters: C_{esp}, C_w SPD Selectors IP_{src} IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY

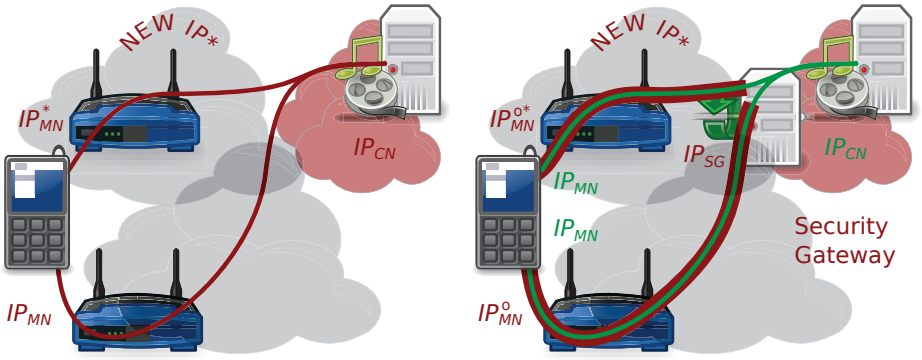
(A) ESP Transport

SPD (Outbound Traffic)	
Selectors	Processing info
IP_{dst} : IP_{CN} IP_{src} : IP_{MN} P_{dst} : ANY P_{src} : ANY	PROTECT Mode : Tunnel IP^{Tunnel}_{src} : IP_{MN} IP^{Tunnel}_{dst} : IP_{SG} Proto. : ESP Algo. : ENCR, AUTH SPI
IP_{dst} : IP_{CN} IP_{src} : IP_{MN} P_{dst} : ANY P_{src} : ANY	PROTECT Mode : Tunnel IP^{Tunnel}_{src} IP^{*}_{MN} IP^{Tunnel}_{dst} : IP_{SG} Proto. : ESP Algo. : ENCR, AUTH SPI^{*}

SAD (Outbound Traffic)	
SPI : SPI IP_{dst} : IP_{SG} IP_{src} : IP_{MN}	Crypto. : K_e, K_a Counters: C_{esp}, C_w IP^{Tunnel}_{src} : IP_{MN} IP^{Tunnel}_{dst} : IP_{SG} SPD Selectors IP_{src} : IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY
SPI SPI^{*} IP_{dst} : IP_{SG} IP_{src} IP^{*}_{MN}	Crypto. : K_e, K_a Counters: C_{esp}, C_w IP^{Tunnel}_{src} IP^{*}_{MN} IP^{Tunnel}_{dst} : IP_{SG} SPD Selectors IP_{src} : IP_{MN} IP_{dst} : IP_{CN} P_{src} : ANY P_{dst} : ANY

(B) ESP Tunnel

TABLE 4.4: SPD & SAD MN IPsec Configuration: Impact of Multiple Interfaces



(A) ESP Transport (B) ESP Tunnel

FIGURE 4.5: Description of IPsec Multiple Interfaces with Transport and Tunnel mode

4.5 IKEv2

IKEv2 [KHNE10] is used to negotiate SA between the MN and the CN (resp. between the MN and the SG). It has been standardized by the IETF. When the MN and the CN (resp. SG) are negotiating an SA, they first establish an IKEv2 authenticated and secure channel. This channel is protected by an SA called IKE_SA and is used to exchange IPsec signaling. Presentation will be limited to the signaling of the creation in section 4.5.1 and deletion in section 4.5.2 of a SP / SA.

4.5.1 CREATE_CHILD_SA Operation

The CREATE_CHILD_SA exchange is used to: (1) rekey a SA, (2) Rekey an IKE_SA, and (3) Create a new SA. When Traffic Selectors are provided, the Responder knows it is for a CHILD_SA, and absence of the Notify Payload REKEY_SA indicates the exchange is for a creation. Figure 4.6, illustrates how Initiators and Responders can create a child SA. More details can be found in [KHNE10] Section 1.3 and Annex C.4.

Figure 4.6 presents the exchanges for creating a new SA. Many payloads may be involved in order to deal with all possible configurations, but 4 payloads are mandatory whereas the remaining payloads mentioned in "[ ]" are optional. As a result, there are three categories of Payloads involved in the CREATE_CHILD_SA:

- 1. **SA Configuration Payloads:** When a SA is created, Initiators and Responders agree on: (1) *Which Traffic to protect?*. This is done with Traffic Selectors Payloads (TSi, TSr). Then, (2) *How to protect the selected traffic?*, that is to say agreeing on which protocol to use. This is done with the Security Association Payloads (SA). Finally, (3) *Which cryptographic material are we using?*. The cryptographic is generally generated from the existing established IKE_SA in combination with Nonces. This is done with the Nonce Payload (Ni, Nr). Eventually Initiators and Responder can agree on another shared secret and proceed to the Diffie-Hellman exchange with the Key Exchange Payloads (KE).
    - TSi, TSr are the Traffic Selectors Payload described in [KHNE10] section 2.9. Traffic

Selector (TS) Payload consists in a list of Traffic Selectors which are an address range (IPv4 or IPv6), a port range, and an IP protocol ID. TSi consists in the Traffic Selector associated to the Source (or the Initiator), TSr those associated to the Destination (or Responder). SA Payload contains a list of SA proposals. Each proposal has a Number, a SPI, a protocol —IKE, ESP, AH —. Then follows a list of transforms: Encryption Algorithm (ENCR), Pseudorandom Function (PRF), Integrity Algorithm (INTEG), Diffie-Hellman group (D-H), Extended Sequence Numbers (ESN). A proposal can have multiple values for the same type of transform —typically; a proposal can include multiple ENCR algorithms. Each Transform can be associated attributes —like the Key Length. Note there is a single Key Length per transform.

- Ni, Nr are the Nonce Payload and are used to generate the cryptographic material. New key material is generated with SK_d established during the IKE_SA exchange, the Nonces (Ni and Nr) and the Diffie-Hellman value if negotiated.

- KEi and KEr are the Key Exchange Payload and are used for the Diffie Hellman exchange.

- 2. **SA fine-grained Configuration Payloads:**
  - N(USE_TRANSPORT_MODE) indicates the negotiated SA uses the Transport mode. If not specified, the Tunnel mode is considered. It must also be included in the Response.

  - N(IPCOMP_SUPPORTED) indicates the compression algorithms supported by the Initiator. The Responder must select at most one.

  - N(ESP_TFC_PADDING_NOT_SUPPORTED) Notify Payload asserts that packets containing Traffic Flow Confidentiality (TFC) padding over the Child SA being negotiated will not be accepted.

  - N(NON_FIRST_FRAGMENTS_ALSO) Notify Payload indicates how fragmentation is managed.
  - V Notify Payload is a Vendor ID Notify Payload.
- 3. **Road Warrior Configuration Payloads:** The Configuration Payload (CP(CFG_REQUEST)) is only used with a SG when the MN wants the SG to provide the inner IP address in a CP(CFG_REPLY) Configuration Payload.
  - CP(CFG_REQUEST)

  - CP(CFG_REPLY)

The SA Payload sent by the Initiator contains various proposals, with different transforms and attributes. The proposals are ordered, and the Responder is expected to select different attributes within a proposal.

Note that SA are unidirectional, thus the SA exchange message results in creating two distinct SAs on the Initiator and on the Responder. The SPI field corresponds to the one of the sending entity for its inbound traffic. Once the SA exchange is finished, the Initiator (resp. the Responder) has two SAs: one for inbound traffic whose SPI has been assigned by the node, and one for outbound which has been assigned by the other node. Each node can chose SPI associated to inbound traffic so to avoid SPI collision. On the other hand outbound traffic SPI is assigned by the other peer. This may result in a collision —although it rarely happens in practice. In fact

SPI collision for outbound traffic does not really matter, since the SA is being provided by the SP, which most likely addresses the proper SA not using SPI but addresses for an SA structure. On the other hand, for inbound traffic, the proper SA is identified by the SPI and so must be unique in the system.

In figure 4.6 HDR is set for the IKEv2 header, and SK indicates that the message is encrypted and authenticated.

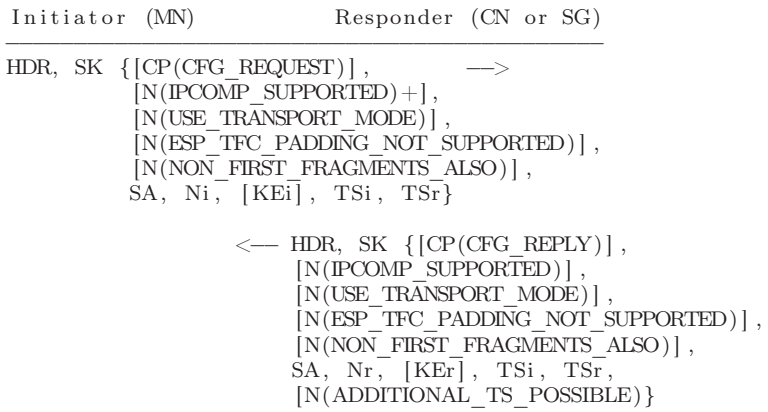


FIGURE 4.6: Messages Exchange for IKEv2 CREATE_CHILD_SA

4.5.2 DELETE Operation

The DELETE Payload (D) contains a protocol field (ESP, AH, IKE), and a number of SPIs associated to the sender's inbound traffic. The Responder sends back a DELETE Payload (D) with the other SPI associated to its inbound traffic, meaning that both SAs have been deleted. The Delete exchange is represented in figure 4.7. Similarly to figure 4.6 HDR is set for the IKEv2 header, and SK indicates that the message is encrypted and authenticated.

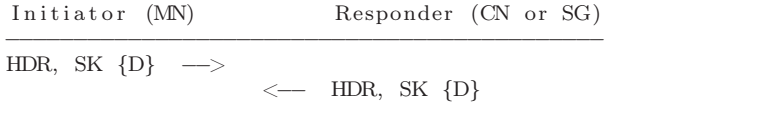


FIGURE 4.7: Messages Exchange for IKEv2 DELETE

4.6 MOBIKE

This section presents MOBIKE [Ero06], the MOBility and Multihoming IKEv2 extension, defined at the IETF. The use case considered by MOBIKE is a MN connected to a SG with the Tunnel and through a single Interface. The communication between IP_{MN} and IP_{CN} is tunneled to the SG, and the packet on the network has IP^{o}_{MN} and IP_{SG} as outer IP addresses.

Figures 4.8 and 4.9 illustrate the exchanges between the MN and the SG during Mobility and Multihoming operations. In both figures, steps 1) and 2) are the IKE_INIT exchanges.

In step 1, the MN creates the IKE_SA. The IKE_SA is a protected channel used to exchange IPsec signaling between the MN and the SG. What makes this SA different to the other SA is that during the IKE_SA negotiation a shared secret is generated between the MN and the SG, which is used to derive all keys for the next SAs, also called Child SA. The payloads used to create the IKE_SA are SAi1, KEi, Ni Payloads on the Initiator's side and SAr1, KEr, Nr on the Responder's side. SA(i,r)1, KE(i,r) and N(i,r) Payload works as for the CREATE_CHILD_SA of section 4.5.1. In step 2, the MN and the SG proceeds to the authentication, network configuration, Child SA configuration —the SA intended to protect the communication between the MN and the CN —and MOBIKE configuration. Usually the Child SA (SA2) is the reason why we initiated the IKE_INIT exchange. Authentication is proceeded by asserting its Identity (ID(i,r)), providing proof-of-ownership (AUTH(i,r)) and Certificates (CERT(i,r)). Network configuration consists, for the MN in requesting an IP address to the SG (CP(CFG_REQUEST)), that is provided in the Configuration Payload (CP(CFG_REPLY)). The creation of the SA requires the MN and the SG to agree on the traffic to secure with the Traffic Selectors (TSi, TSr) as well as the different protocols required to secure the communication. These protocols are provided in the Security Association Payloads (SAi2, SAr2), as described in section 4.5.1. Finally the N(MOBIKE_SUPPORTED) indicates that the MN and the SG support MOBIKE, and that MN or SG can proceed to further Mobility or Multihoming operations. A complete security analysis of the Secure Diffie Hellman and the IKEv2 Initial Exchange is provided in [Roy09, RDM08].

4.6.1 Hard Handover Mobility

This section details, when the MN acquires a new outer IP address $IP^o_{MN}*$, and updates its IPsec Tunnel. The Hard Handover Mobility results in tunneling the communication between the MN and the CN (IP_{MN} - IP_{CN}) using $IP^o_{MN}*$ and IP_{SG} as outer IP addresses. The scenario is illustrated in figure 4.4b of section 4.4.2.

When the MN has changed its IP address ($IP^o_{MN}*$), it sends the SG a N(UPDATE_SA_ADDRESSES) Notify Payload as represented in step 3 of figure 4.8. This Payload does not contains any data, like for example the new IP address $IP^o_{MN}*$. Since the MN only has a single interface, and MN sends the Payload from its new interface, the new IP addresses for the outer header are necessarily those of the IP header. When the SG receives the N(UPDATE_SA_ADDRESSES) Notify Payload, it updates the SA indexes and the information associated to the Tunnel in the SA and eventually in the SP —actually, SP may not be updated, because outer IP addresses are only used to create the SA. Section 4.4.2 and table 4.3b details how Mobility impacts the IPsec databases.

Since the IP header is not protected, IP addresses are not protected by the IKEv2 channel, and the Responder may check their validity, before updating the SPs / SAs. The Return Routability Check exchange using the COOKIE2 Notify Payload has been designed for that purpose and is represented in step 4 of figure 4.8. Note that with the Tunnel mode, the Traffic Selectors of the SP are not changed, only the Selectors of the SA, and the Tunnel outer IP addresses are changed, as mentioned in section 4.4.2.

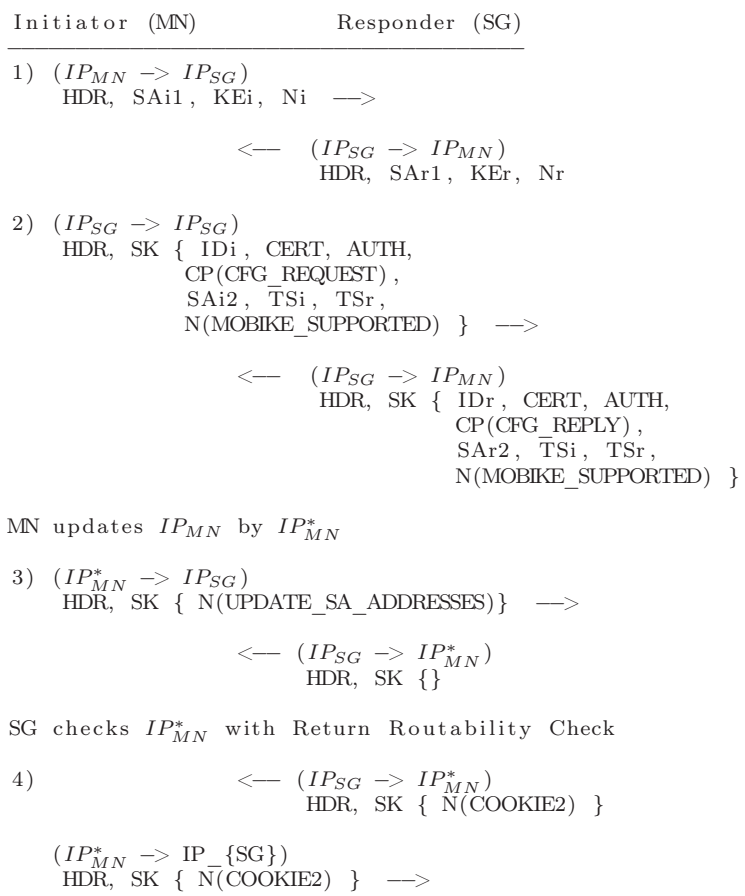


FIGURE 4.8: Messages Exchanges for IKEv2 MOBIKE Mobility Hard Handover

4.6.2 Multihoming

To indicate Alternate IP addresses may be used for Multihoming, the MN sends N(ADDITIONAL_IP4_ADDRESS) or N(ADDITIONAL_IP6_ADDRESS) Notify Payload to the SG (resp. CN), as represented in step 3 of figure 4.9. When the Primary Address is not reachable anymore, the CN checks the MN is reachable on the Alternate IP addresses via the Return Reachability Check exchange and N(COOKIE2) Notify Payload. The Routability Check Exchange is illustrated in step 3 of figure 4.9. Then, the SG (resp. the CN) sends the MN a N(UPDATE_SA_ADDRESSES) Notify Payload so the MN updates the IP outer addresses of the Tunnel, in step 3 of figure 4.9, as for Mobility. Unlike with Mobility, the SG (resp. CN) makes the MN updates its own IP address, rather than the sender's one.

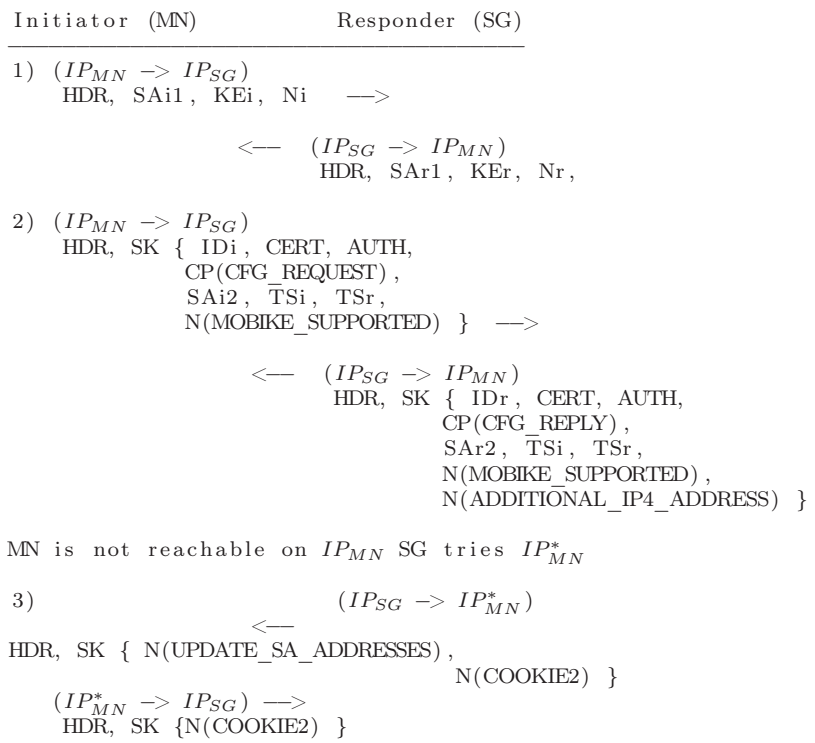


FIGURE 4.9: Messages Exchanges for IKEv2 MOBIKE Multihoming

4.7 MOBIKE-X

This section describes MOBIKE-X, the MOBIKE extension we designed to provide, for both Tunnel and Transport mode, IPsec Mobility Hard Handover, Soft Handover, Multihoming and Multiple Interfaces operations. Section 4.7.1 provides the use cases we consider for IPsec Mobility, that is: (1) providing IPsec Mobility with Soft Handover, and (2) using IPsec to offload the mobile data traffic of End users from RAN to WLAN. From the use cases description, section 4.7.2 provides the problem statement and show why neither MOBIKE nor IKEv2 really address the problem. Then, section 4.7.3 provides the main guidelines we used to design MOBIKE-X. More specifically it shows how MOBIKE-X is reusing the design of MOBIKE. Finally section 4.7.4 defines the exchanges defined in MOBIKE-X, with the main involved Notify Payload. Currently, Motivations for designing MOBIKE-X have been published in IETF drafts [Dan09a, DC12]. MOBIKE-X has been described in [Dan09b].

4.7.1 Use Cases

MOBIKE [Ero06] described in section 4.6 addresses one use case which is a MN with a single Interface connected to a SG. Since the design of MOBIKE other use cases have emerged, due to terminal evolutions, new Mobile Usages and Network economy. This section lists the different use cases we have considered to figure out MOBIKE's next extensions.

MOBIKE [Ero06] has been designed and published in 2006. At that time, MNs were not Smartphones, but rather labtops. The Mobility use case considered in MOBIKE is a labtop VPN connected through a wired connection in the office, then the labtop may use the company's WLAN, eventually the GPRS connectivity or the home WLAN network. The main goal of MOBIKE is to avoid reconnection that would require the authentication to be replayed, with eventually user interaction. Thus MOBIKE provides Mobility and multihoming mechanisms that deal with the change of IP addresses. Note that while using GPRS, mobility is handled by the radio layer, not MOBIKE, which means that the scenario considered in MOBIKE had very few MOBILITY operations. In June 2007, the iPhone started modifying the use of Smartphones. They were not used only for calling / texting, but large screens make possible web browsing, documents reading, checking and writing emails... Executives started heavily using such devices, which required a protected access to the office Network. IPsec with SG happens to be the natural solution. Today's, IPsec VPN are supported by most Mobile Platforms, iPhones [App] , Android [And], Windows Mobile [Mic].

Contrary to the use case foreseen in MOBIKE, Smartphones are much more likely to perform Mobility than labtops. In fact labtop using GPRS for mobility is not likely to happen since labtops are not easy to handle in buses, underground. On the other hand, Smartphones are much more convenient for Moving End Users. Then WLAN Access Points are much more common than they used to be in 2006. Many DSL Boxes provide WiFi Access, and many European ISPs are developing "WiFi Communities". Furthermore pricing policy makes WiFi Access via WiFi Community free, whereas 3G data is taxed. As a result, Smartphones are designed for Mobility, and are encouraged to use WiFi Access for their VPN. This results in multiple Mobility Operations during a given communication.

MOBIKE provides Hard Handover Mobility which happens not to fit current usage. In fact MOBIKE provides Hard Handover Mobility which results in a lost of IP packets. When too many packets are lost, new arriving packets out of the Anti Replay Attack Windows, are rejected, and this may result in breaking the connection. To avoid those inconvenient, MOBIKE should be extended to Soft Mobility.

Mobile data in 2015 are expected to be around 50 times larger than it is today. RAN infrastructure will not be able to handle such traffic, and ISPs are looking for offloading this traffic from RAN to WLAN. Contrary to a pricing policy that encourages the End User to use WLAN rather than RAN, in this case, the End User requests a RAN connectivity, and the ISP must be able to provide the End User a connectivity with similar Quality of Service and with the same Security. Unlike RAN, WLAN may not be managed by the ISP. WLAN Access can be provided by third party, by individual DSL subscribers... In term of User Experience, the ISP can hardly rely on the WLAN Access availability. A DSL subscriber is likely to reboot its box for example.

Some protocols like Multi Path Protocol [FRH$^+$11] (MPTCP) and to some extent SCTP have been designed to establish a communication over unreliable accesses with Multihoming, or with Multiple Interfaces. ISPs, manufacturers and academics are implementing MPTCP in Smartphones, and the offload is one of the two scenarios that motivated MPTCP. On a security point of view, when the communication is going through a WLAN not owned by the ISP, the ISP has no guarantee about the privacy, confidentiality of its End User. RAN used to rely on Radio Security. This was fine since the Access Point belonged to the ISP, thus providing a security entry point of the ISP trusted network. Offloaded communications cannot rely on Radio or WLAN security, because the WLAN Access Point may not belong to the ISP. Security must be done at higher layers and securing the IP layer with IPsec is the only way to secure a communication without modifying third party applications code. In order to benefit from the MPTCP functionalities with IPsec protection, we need to provide IPsec the basic Multiple Interface, Multihoming and Mobility operations. This means that a node must be able to add, remove, update an Interface, and provide alternate IP addresses in case of failover. This must be done for a single connection or for

all connections. A MN can benefit from Multiple Interfaces and access (even without MPTCP) a single SG. On the other hand, for Real Time Traffic, ISPs are interested in avoiding the extra tunnel overhead, and use Transport mode with dedicated application Servers. MOBIKE should be extended to Multiple Interfaces, Mobility, and Multihoming for both IPsec Transport and Tunnel modes.

Some companies use IPsec as a distributed firewall, to protect equipments / resources against intruders. This is complementary to the access policy provided by a single entry point firewall. In that case, IPsec Multiple Interfaces, Mobility and Multihoming operations would make possible a node to preconfigure its IPsec layer while discovering new Interfaces. While the MN discovers the IP addresses, IPsec won't need re-authentication. An alternative is to distribute all the company policies to all MN, which could be avoided with dynamic configuration.

4.7.2 Problem Statement

Today, MNs with Multiple Interfaces are common, and MNs want to take advantage of Multiple Interfaces to enhance the End User experience or to better use the Network. The scenario addressed in this thesis is a MN connected to services via Wireless LAN (WLAN) and Radio Access Network (RAN). Attachment to RAN and WLAN already requires at least two interfaces, but we assume that the MN has multiple WLAN Interfaces. WLAN may not be managed by the ISP, and thus may be unreliable. On the other hand, WLAN may provide higher bandwidth and availability than RAN. So, the MN must take advantage of its Multiple WLAN Interface to balance WLAN unreliability. More specifically, MN wants to:

- Add an Interface. While the MN is moving and discovering a new Interface, the MN wants to be able to ADD this new connectivity. If connections are able to use multiple IP addresses, this interface should be added either to all communication or to a selection of those communications.
- Remove an Interface. While the MN is moving, some Interfaces may not be reachable. In that case, the MN should remove those IP addresses from the connections.
- Move traffic from one Interface to the other. Mobility can be associated to some traffic, or to the whole traffic. Mobility can be done through Hard Handover or Soft Handover.
- Benefit from Multihoming on a connection basis. Multihoming is intended to prevent WLAN Access Point Failover. Depending on the nature of the traffic, the MN may want to use different Alternate IP addresses for different traffic. For example, the RAN Interface should be used only for Real Time Applications, whereas data downloads or emails may only use other WLAN Access Points.
- Manage one or multiple connections over Multiple Interfaces. More specifically, the MN may perform bandwidth aggregation by adding an Interface for a single traffic. Similarly, it may remove traffic on an Interface. In addition, the MN may also be able to move with Soft Handover or Hard Handover a given connection. At last, the MN may configure the various connections with different and adapted alternate IP addresses.

Protocols like MPTCP or SCTP provide most of these facilities. The purpose of these protocols is to carry data. In this thesis, we want that such protocols can benefit from IPsec protection, and that IPsec does not results in blocking communications. In fact, IPsec can be seen as a firewall, which if not properly configure can block the traffic and break the communication. Thus, all above mentioned requirements must be provided for IPsec.

Section 4.5 shows that IKEv2 provides the CREATE_CHILD_SA and DELETE exchange to create an SP/SA and to remove it, which makes Multiple Interfaces traffic management, Mobility, as well as addition, removal and Interface update possible. However, (1) the CREATE_CHILD_SA exchange is not mandatory especially for light IKEv2 implementation. [KHNE10], section 1.3 *The responder sends a NO_ADDITIONAL_SAS notification to indicate that a CREATE_CHILD_SA request is unacceptable because the responder is unwilling to accept any more Child SAs on this IKE SA. This notification can also be used to reject IKE SA rekey. Some minimal implementations may only accept a single Child SA setup in the context of an initial IKE exchange and reject any subsequent attempts to add more.* (2) CREATE_CHILD_SA and DELETE exchanges are per SA exchanges. When the MN has multiple SAs with its CN, for example, the MN must initiate multiple CREATE_CHILD_SA exchanges. The exchanges can be carried in the same INFORMATIONAL exchange. (3) CREATE_CHILD_SA has much larger Payload. The most basic use case is a MN connected to a SG or a CN via Multiple Interface and wants to announce that it may be reachable on an additional Interface. Traffic Selectors (TSi, TSr), Security Association (SA) Nonces (Ni, Nr) do not necessary have to be specified. (4) CREATE_CHILD_SA exchange requires unnecessary CPU computation, Keys are derived involving a Pseudo-Random Function (PRF). This may be insignificant with few SA creations, but with WLAN connectivity, a MN is likely to encounter multiple WLAN, and so likely to create multiple SAs, which makes us consider this additional cost.

MOBIKE described in section 4.6 considers mobility and multihoming operations. However (5) MOBIKE only considers updating the Tunnel mode SA : SP parameters and ignores the Transport mode. Then, (6) MOBIKE considers with a single Interface, thus Mobility operations are provided for the entire MN and there is no possibilities to Multihoming or Mobility on a traffic base. More specifically, traffic based operation are an intermediate between the per SA and the entire MN. (7) MOBIKE only makes possible Hard Handover because Soft Handover requires two distinct Interfaces.

4.7.3 Protocol Design

From the problem statement section (section 4.7.2), current IKEv2 [KHNE10] or current MOBIKE [Ero06] does not address Mobility, Multihoming and Multiple Interfaces requirements of current communications. In order to address those requirements, MOBIKE must be extended on the following points:

- Traffic Selector: the MN MUST be able to explicitly specify which traffic the operation applies. Expression of the traffic can be done with different granularities to enhance the IKEv2 per SA or the MOBIKE ALL SAs.
- Multiple Interfaces Management: MOBIKE must consider Multiple Interfaces Management for operations it has been designed for like Mobility and Multihoming. It must also provide generic extension to make Multiple Interface Management, such as ADDing, REMOVing or UPDATing an Interface. Those operations must be defined for both Transport and Tunnel mode.
- Multihoming for Multiple Interfaces: Multihoming should be provided with different Alternate IP addresses depending on the network the connection is currently working. Note that it is also related to Multiple Interface Management.
- Mobility: MOBIKE Hard Handover must be extended to the Transport mode. to make possible Soft Handover for both Transport and Tunnel mode. Note that Soft Handover is related to Multiple Interfaces Management.
- Mobility for Transport: to support all offload architecture, especially those with End-to-End Security.

Note that while we are focused on SP / SA creation / removal / update, we do not have yet really defined how the IKE_SA is impacted by those operations. The IKE_SA is not used to carry traffic, but the signaling IPsec messages. The questions are thus: what should we do with the IKE_SA when the MN adds an Interface? Should we have Soft Handover for the IKE_SA? How Multihoming and Mobility are performed with the IKE_SA channel?

First, we did not see any benefits for the IKEv2 channel to use Multiple Interfaces. IKEv2 signaling does not require bandwidth aggregation for example, and using Multiple Interface in case one fails is related to Multihoming. As a result, for simplicity, we consider that IKEv2 channel uses a single path at a time. These restrictions on the IKEv2 channel simplify the implementation of MOBIKE-X, by avoiding changes with the IKE_SA. Similarly, we do not see either clear advantage for Soft Handover. More specifically, IKEv2 has established a channel based on the IKE_SA. This makes, for example, a MN perform a Mobility, change its IP address and send a N(UPDATE_SA_ADDRESSES) to the SG, as illustrated in figure 4.8. The Payload is received by the SG from an unknown IP address, but is not rejected. This means the IKEv2 channel is not broken when the IP addresses are changed. In addition, IKEv2 exchanges are small, and usually involve a Question followed by a Response. Only very few exchanges involves more than one exchange. This means that the probability mobility would occur during a exchange is quite low. Soft Handover for the IKEv2 channel would consider this corner case, and for simplicity, we did not consider it in MOBIKE-X and leave it for future MOBIKEv3. As a result, Hard Handover as defined in MOBIKE seems fine for the IKEv2 channel. Finally, the IKEv2 channel must be Multihomed, in the same way it is Multihomed with MOBIKE. By Multihomed, we understand, the application is able to detect an IP address has changed, and do not crash when IP addresses are changed.

The reason we do not insist on modifying the IKEv2 channel behavior is that the IKEv2 application has been designed to handle all those properties, and so modifications on IKEv2 behaviors would introduce major changes on IKEv2. This is completely different from adding functionalities to IKEv2. Note that some work has investigated how the IKEv2 application may benefit from Mobility and Multihoming support provided by the Transport Layer rather than the application layer. [GTS+06] for example considering using IKEv2 over SCTP. In that sense SCTP would be able to provide IKEv2 Mobility and Multihoming features as described in section 4.1.2

4.7.4 Protocol Description

This section provides a brief description of the MOBIKE extension MOBIKE-X described in [Dan09b]. There is an INFORMATIONAL exchange that makes the MN and the CN or the SG agree on the MOBIKE version. We considered MOBIKE-X as the second version of MOBIKE.

4.7.4.1 MOBIKE-X SELECTORs Notify Payloads

First of all, MOBIKE-X Selectors must not be assimilated to the Traffic Selectors used in the SPD. MOBIKE-X Selectors are used to Select which SA or SP an UPDATE_SA_ADDRESSES, an ADD_SA_ADDRESS or a REMOVE_SA_ADDRESS is performed. Selectors are designed to provide different granularities. In addition to the various granularities, the MN may be able to select traffic using SPD Selectors Index or SAD Selectors Index. In fact SPD Indexes are more convenient for applications with no view on the SAD. SAD Indexes are more convenient for lower layers. However, whatever Selector is used this will generate modifications on at least the SAD and eventually on SPD.

```
N(SELECTOR
    SELECTOR_SAi+
    SELECTOR_SAr+
    SELECTOR_SPI+)
N(SELECTOR
    SELECTOR_META
...
ACTION_1
ACTION_...
ACTION_n

N(END_OF_SELECTOR)
```

FIGURE 4.10: Description of the SELECTOR Notify Payload with MOBIKE-X

Figure 4.10 illustrates the structure we used for selecting the traffic. When designing SELECTORs Notify Payload, [Dan09b] considers Traffic Selector Payload designed in [KHNE10] section 3.13. The Traffic Selector Payload contains different Traffic Selectors of type TSi or TSr. There can be numerous Traffic Selectors, the matching traffic is the one that matches one TSi and one TSr. We adopt a similar strategy. The SELECTOR Notify Payload contains various types of Selectors: SELECTOR_SAi, SELECTOR_SAr —that have the same structure as the Traffic Selectors defined in [KHNE10] section 3.13.1. —and SELECTOR_SPI. SELECTOR_SA* carries indication of the Protocol —TCP, UDP, SCTP —, the port range and the IP range. SELECTOR_SPI carry the SPI value. There are other specific Selectors. SELECTOR_IPSEC_PROTO that selects the IPsec protocol —ESP, AH —, SELECTOR_IPSEC_MODE Selects the IPsec Mode —Transport, Tunnel —, SELECTOR_META that makes possible to select IKE_SA, ALL_NON_IKE_SA... Within a SELECTOR Notify Payload, the traffic considered is the traffic that matches ALL Selector parameters. If more than one SELECTOR Notify Payload is used, the selected traffic is the traffic selected by one SELECTOR or another SELECTOR. All actions that follow the SELECTOR Notify Payload are only applied to the traffic indicated by SELECTOR, until the END_OF_SELECTOR Notify Payload is encountered.

If the SELECTOR is invalid, or an Action is unexpected an error is reported.

To be compatible with MOBIKE, if no SELECTOR is specified, we assume that Actions apply for all SAs.

Note that SELECTOR Notify Payload are used to defined the SA the action must be performed. For each action, —remove, add, update....—, the Responder must check that the action matches the Security Policies. Checks must be performed for each action, and SELECTORs do not provide any guarantee, whether an action can or cannot be performed.

4.7.4.2 ADDITIONAL_IP_ADDRESS

The ADDITIONAL_IP_ADDRESS Notify Payload works like the ADDITIONAL_IP4_ADDRESS and ADDITIONAL_IP6_ADDRESS Notify Payload. The reason we use a single Notify Payload for both IPv4 and IPv6 addresses is that we use a generic IP parameter in the whole protocol. In this IP parameter, the version is specified, as well as other information relative to the IP address.

Another difference is that the Alternate IP address is assigned to a single or a set of SAs rather than to the entire MN. The Selection is performed via the SELECTOR Notify Payload. If not specified, the Alternate Address applies for the entire MN, that is to say the SAs and the IKE_SA, as specified in [Ero06].

4.7.4.3 UPDATE_SA_ADDRESSES Notify Payload

The UPDATE_SA_ADDRESSES Notify Payload has a similar function as the one defined in [Ero06]. If the Responder receives an UPDATE_SA_ADDRESSES Notify Payload without any associated parameters, then the Responder extracts the IP addresses of the IP header and updates all SAs specified by the SELECTORS. For SAs using the Tunnel mode, it works exactly as specified in [Ero06], and only the outer IP addresses of the Tunnel are updated. Contrary to [Ero06] SAs using the Transport mode are also updated.

However, it is also possible to specify which IP address is to be replaced (OLD_IP) and the new value for the IP address (NEW_IP). First for the Tunnel mode, it makes possible to specify whether the Inner or the Outer IP address needs to be replaced. Then it also makes possible a simple UPDATE without involving SELECTORs. Suppose a MN is connected via IP_A, IP_B, IP_C and acquires another IP address (NEW_IP). There is not much signal on IP_B, and the MN wants to replace IP_B by IP_NEW. This scenario is quite common and does not require SELECTOR Notify Payloads. For All SAs, the MN wants to replace IP_B by IP_NEW. On the contrary, if part of the traffic on IP_B is to be moved on IP_NEW, then SELECTOR Notify Payloads are required.

4.7.4.4 REMOVE_SA_ADDRESS Notify Payload

The REMOVE_SA_ADDRESS Notify Payload is used by the MN to remove an Interface when the MN is not reachable on this Interface. Similarly to the UPDATE_SA_ADDRESSES Notify Payload, the REMOVE_SA_ADDRESS Notify Payload specifies the IP address to be removed. Note that SAs with the matching IP address are removed. When the MN wants to remove a single SA, it is recommended to use the DELETE exchange as specified in [KHNE10].

4.7.4.5 ADD_SA_ADDRESS Notify Payload

The ADD_SA_ADDRESS Notify Payload is used when the MN get a new Interface. Since IP addresses are used as Selectors, and Selectors do not allow list of parameters, it is hardly possible to simply ADD a new IP address to an existing SA. Adding an IP address requires to create a new SA. This new SA must have its own SPI and we must make both MN and CN (resp. SG) aware of the SPIs. On the other hand, cryptographic material, counters, selectors are not negotiated and are derived from an existing SA. Thus the ADD_SA_ADDRESS Notify Payload must carry the following information: the IP address to be added, the derived SPI, the SA used to derive the new SA. The new SP/SA can be seen as a copy of the previous SP/SA where SPI, IP address are modified.

The IP address to modify in the copied SP/SA is always the IP address of the Initiator (or the MN in our case). With Tunnel mode, the IP parameter specifies whether the inner or outer IP address is considered.

SPI agreement between the MN and the CN or the SG can be done in various manners. One way to do so, is that the ADD_SA_ADDRESS carries an SPI specified by the Initiator, and the SPI of the SA used to derive the new SA. When the responder acknowledges, it must send back the SPI used for its incoming traffic. This method provides the following advantages: (1) it provides randomness in the SPI, and (2) it identifies the SA used to derive the new SA. The inconvenient of this method is that it provides a per SA negotiation, and this is why we designed an alternative

method that defines a method used to generate the SPI. One way to generate is to provide a Nonce. When the Responder receives this Nonce, for all SAs indicated by the SELECTOR, or ALL SAs that are non IKE_SA, it generates a NEW_SA whose content is derived from the selected SA and whose $SPI(NEW\_SA) = SPI(SA) + Nonce$. The advantage of this method is that is provides a generic way to derive the new SAs. The inconvenience of the method is that SPIs are not randomly generated, SPI may encounter SPI collision and, SAs may be unnecessarily created.

SPI are not randomly generated because they are derived from Nonce and a previous SPI value. SPI are public values, and it does not seem that a lack of randomness may cause any troubles. On the other hand, generating SPI this way may generate SPI collision. First the MN —or the Initiator in our case —must choose a Nonce that avoids SPI collision on its side. This means that for all SAs selected by SELECTOR, there is no SPI of value $SPI(SA) + Nonce$. Once the Nonce has been chosen, all inbound and outbound new SAs are created. A collision on outbound SAs SPI does not really matter, however, when possible, we prefer SPI to be unique. When the Responder receives the Nonce, it proceeds in a similar way on its side. All new SAs are derived from the selected SAs. Collision for outbound SAs does not really matter. As mentioned in section 4.4.1 SAs for outbound traffic do not use the SPI as an identifier of the SA. Those SAs are pointed by the SP, which most of the time uses memory addresses. However, for inbound SA, collisions are not acceptable. The Responder sends an Error Notify Payload with inbound SAD Selectors where collision occurs as well as the SPI to be used for inbound SA. Because SPI may not be unique, we provide all SAD Selectors (IP source, IP destination, SPI). When a large number of SAs are added, we believe collisions would happen only to a small number of SAs only. When a collision occurs, we recommend to proceed on a per SA negotiation. If unnecessary SAs have been created and are not used, then we also recommend to DELETE those SAs. This can happen when the MN has already three Interfaces connected to the SG. Adding a new Interface will trigger the creation of three new SAs based on the three existing SAs, but a single SA would have been sufficient. In that case, we recommend to use SELECTOR to avoid the creation of unnecessary SAs.

MOBIKE recommend that the SG or the CN performs a Return Routability Check when the MN performs an UPDATE. The Return Routability Check is performed with the COOKIE2 exchange, and is intended to check (1) the MN is reachable to the updated IP address and (2) that the MN is really behind the new IP address. Because MOBIKE only uses a single interface, the interface the MN is reachable on is also the Interface used for the IKEv2 channel. Thus the COOKIE2 exchange is using the IKEv2 channel, while testing the reachability. When the node has Multiple Interfaces, and wants to check the reachability of the different Interfaces, it cannot use the IKEv2 channel. In section 4.7.2, we specified that IKEv2 channel is using a single interface. Hence, the COOKIE2 exchange must be sent to the Interface we want to check the reachability, as opposed to the interface used for the IKEv2 channel. When the MN receives the COOKIE2 exchange, it SHOULD send it on tested Interface. If that is not possible, the IKEv2 channel can be used. This would mean at least that the MN is reachable on the new Interface.

Similarly, if the MN sends the ADD_SA_ADDRESS Notify Payload from the newly acquired IP address, the SG may respond on that new IP address. The COOKIE2 exchange is not necessary only if the new IP address is mentioned in the Notify Payload, that is, if the new IP address to add is not taken from the IP header. Otherwise, the SG should proceed to a COKKIE2 exchange.

4.7.4.6 Soft Handover and SOFT_HANDOVER Notify Payload

Soft Handover is clearly a combination of adding an Interface followed by removing the previous Interface after some delay. This involves two exchanges that are replaced by the SOFT_HAND_OVER

Notify Payload. This payload includes all necessary arguments for ADD and REMOVE, and optionally a time parameter between ADD and REMOVE. If not specified, the default Time is 2 *sec.*

Figure 4.10 compares the exchanges for a Soft Handover and a Hard Handover. In figure 4.11a, step 1 consists in establishing a IKEv2 channel. In this step, MN and CN or SG agree they are both MOBIKE or MOBIKE-X enabled. The initial IKEv2 exchange is described in section 4.6.In step 2, the MN sends the SG or the CN an UPDATE_SA_ADDRESSES Notify Payload and indicates it has already changed its IP address. The message has been sent using the IKEv2 channel, however, the IP addresses may not have been securely carried. For example, as described in MOBIKE, the IP addresses are carried in the IP header. The IP addresses may thus have been forged. Furthermore, the CN or the SG may also want to check whether the MN is reachable at this IP address. The CN or the SG performs a Return Routability Check in step 3, to check both reachability and IP ownership.

Figure 4.11b illustrates the case of a Soft Handover where all actions are performed one after the other. The reason we do not perform all those actions in a single Payload is that the MN may not know in advance if it has to perform a Soft Handover or not. It can, for example and as represented in step 1 of figure 4.11b, get a new IP address $IP^*_{MN}o$, but does not really know which IP address is going to be used later IP_{MN} or $IP^*_{MN}(o)$.

With the Tunnel mode, the main advantage of the Soft Handover, is that it can be transparent to the upper layers. In other words, the Soft Handover can be done with a TCP connection without breaking it. On the other hand, this advantage makes Soft Handover a bit more complex than with the Transport mode, and requires to introduce Primary and Secondary Interfaces. If the SPD is not decorrelated on a per interface base, then a given Traffic Selector, can match multiple SPD. The SPD is an ordered database, and only the first match will be considered. The first match corresponds to the IP outer IP address that is said to be Primary. Other SPs that are not selected correspond to the IP address that is said to be Secondary. Hence, when the MN adds an new Interface, as represented in step 1 of figure 4.11b, a new SP and a new SA with $IP^*_{MN}(o)$ as the outer Tunnel IP address are created. $IP^*_{MN}(o)$ is first considered as a Secondary IP address. This is performed by appending this SP to the SPD. By being Secondary, the SG or the MN can receive traffic on that Interface, but will NOT tunnel any traffic on that Interface. This makes possible to perform the Return Routability Check exchange without disturbing the ongoing communication. After some time, represented by T1 seconds in figure 4.11b, the MN decides that $IP^*_{MN}(o)$ becomes a Primary Interface. This is represented in step 2 of figure 4.11b with the SET_PRIMARY Notify Payload. When receiving this Payload, the SG must "simply" re-order the SPD, so that the Traffic matches the SPD with outer Tunnel IP address $IP^*_{MN}(o)$. From our experience, the easier way is to remove the SP associated to IP_{MN} and then appends it to the ordered list of Security Policies. After some time, if IP_{MN} will not be used anymore, the MN may decide to remove the SP associated to IP_{MN}. The Time is represented with T2 and SP / SA removal is performed in step 3.

With the Transport mode, Soft Handover must be handled by other protocols than the IPsec protocol. For example, it can be MPTCP, SCTP or directly the application that decides to use IP^*_{MN} instead of IP_{MN}. This would correspond to step 2) in figure 4.11b, but no signaling are needed. Then, the application, or the MPTCP or SCTP decides that old IP address IP_{MN} will not be used anymore, and decide to remove it. This is represented in step 3, and even with Transport mode would require an exchange.

Figure 4.11b provides a Soft Handover that involves multiple messages exchanges. It is important that the MN can perform a Soft Handover with a step by step approach, especially when the MN discovering a new IP address, but has not decided yet if a Mobility should be performed.

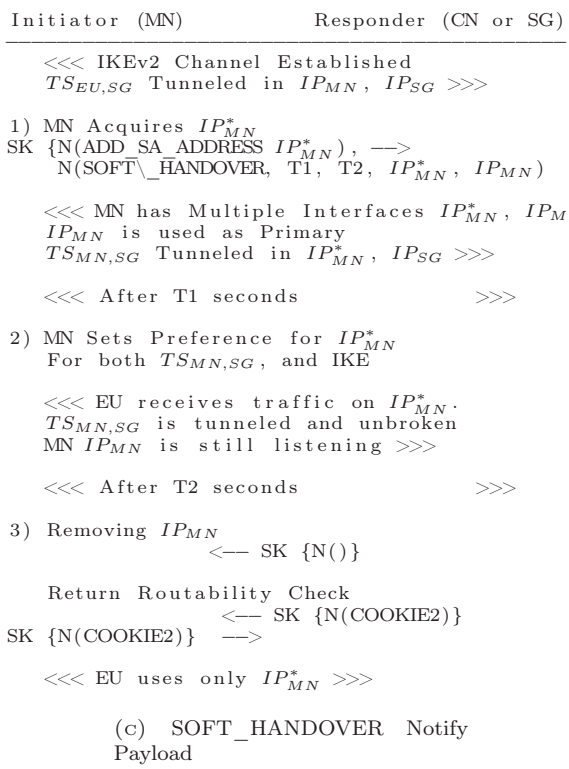
On the other hand, a MN may also perform a Soft Handover while ADDing the new Interface, or request a Soft Handover, on an IP address that has previously been added. The goal of the SOFT_HANDOVER Notify Payload is to limit the number of exchanges between the MN and the SG while doing a Soft Handover. The SOFT_HANDOVER Notify Payload provides parameters to indicate the number of seconds T1, the replacing IP address IP^*_{MN}, the IP address to be replaced IP_{MN}, T2. The SOFT_HANDOVER Notify Payload may also be used in conjunction with the ADD. In order to avoid the COOKIE2 exchange, it is recommended to send the ADD and SOFT_HANDOVER Notify Payloads from the new Interface. The SOFT_HANDOVER exchanges are represented in figure 4.10c.

Initiator (MN)	Responder (CN or SG)
<<< No IPsec Protection >>>	
1) Setting IKEv2 Channel SA^1_{EU}, KE_{MN}, N_{MN} -->	
	<-- SA^1_{SG}, KE_{SG}, N_{SG}
SK {ID_{MN}, AUTH, SA^2_{MN}, TS_{MN}, TS_{SG}, $CP(CFG_REQUEST)$, N(MOBIKE_SUPPORTED)} -->	
	<-- SK {ID_{SG}, AUTH, SA^2_{SG}, TS_{MN}, TS_{SG}, CP(CFG_REPLY), N(MOBIKE_SUPPORTED)}
<<< IKEv2 Channel Established $TS_{MN,SRV}$ Tunneled in IP_{MN}, IP_{SG} >>>	
2) Mobility Hard Handover IP^*_{MN} SK {N(UPDATE_SA_ADDRESSES)} -->	
	<-- SK {N()}
<<< EU uses IP^*_{MN}, $TS_{MN,SG}$ unbroken $TS_{EU,SG}$ Tunneled in IP^*_{MN}, IP_{SG} >>>	
3) Return Routability Check	
	<-- SK {N(COOKIE2)}
SK {N(COOKIE2)} -->	

(A) Hard Handover

Initiator (MN)	Responder (CN or SG)
<<< IKEv2 Channel Established $TS_{EU,SG}$ Tunneled in IP_{MN}, IP_{SG} >>>	
1) MN Acquires IP^*_{MN} SK {N(ADD_SA_ADDRESS IP^*_{MN})} -->	
	<-- SK {N()}
Return Routability Check	
	<-- SK {N(COOKIE2)}
SK {N(COOKIE2)} -->	
<<< MN has Multiple Interfaces IP^*_{MN}, IP_{MN} IP_{MN} is used as Primary $TS_{MN,SG}$ Tunneled in IP^*_{MN}, IP_{SG} >>>	
<<< After T1 seconds >>>	
2) MN Sets Preference for IP^*_{MN} For both $TS_{MN,SG}$, and IKE SK {SET_PRIMARY} -->	
	<-- SK {N()}
<<< EU receives traffic on IP^*_{MN}. $TS_{MN,SG}$ is tunneled and unbroken MN IP_{MN} is still listening >>>	
<<< After T2 seconds >>>	
3) Removing IP_{MN} SK {N(REMOVE_SA_ADDRESSES IP_{MN})} -->	
	<-- SK {N()}
<<< EU uses only IP^*_{MN} >>>	

(B) Soft Handover



(c) SOFT_HANDOVER Notify Payload

FIGURE 4.10: Messages Exchanges for MOBIKE(-X) Mobility

Chapter 5

IPsec Cost Measurement in Mobile, Multihomed and Multiple Interfaces Environment

5.1 Introduction

This section measures the cost of IPsec in a Mobility and Multihoming environment. The testing platform is described in section 5.2. Measurements are provided from an experimental platform. The IKEv2 implementation we used is strongSwan [str], and our MOBIKE-X implementation is based on strongSwan.

To measure the IPsec impact on Mobility, we considered:

- 1. How IPsec configured links impact the Mobility? In other words we compare a Mobility performed over non IPsec protected links to a Mobility performed on IPsec-protected links. In those tests, links are pre-configured, and no IPsec configuration is involved. These tests are necessary to define the true IPsec impact on Mobility when further IPsec configuration will be required. In addition, by comparing the different IPsec modes (Transport and Tunnel), these tests measure the advantages or disadvantages of the Transport mode over the Tunnel mode, when Mobility is required. Currently, IPsec deployment mostly considers the Tunnel mode. Tunneling adds bandwidth overhead with the extra IP header, computation overhead by going twice into the Network stack. Furthermore, sharing a Security Gateway with other Services and End Users adds routing indirections, and latencies that degrade the End User experience. In addition, End Users in a Mobile environment are even more sensitive to such latencies. Comparing Mobility between IPsec protected links with the Transport mode and the Tunnel mode help deciding whether providing a Service specific architecture using the Transport mode is better than a Security Gateway architecture using Tunnel mode. Unlike MOBIKE, that is in charge of the Mobility, in our case the FTP communication is moved using SCTP. The reason we choose SCTP rather than Multi-Path TCP (MPTCP) is that, at the time we started deploying the platform, we thought that SCTP would be more stable and with more advanced developments.

- 2. How IPsec impacts Mobility when IPsec configuration is required? Compared to the previous

case, this case IPsec requires dynamic configuration of the IPsec stack. When the Tunnel mode is used with MOBIKE, changing the outer IP address performs a Mobility Hard Hand over. With the Tunnel mode and MOBIKE-X, Mobility can be done through a Soft Handover. Measuring how MOBIKE impacts the End User makes it possible to check how services using VPN access are impacted by Mobility, and what Soft Mobility may enhance. With the Transport mode, IPsec configuration cannot be used to perform Mobility. In that case, IPsec Mobility configures the IPsec layer so that communication can securely change its interface, but the communication Mobility must be performed by another protocol. For our testing platform, we use SCTP to perform Mobility at the transport layer, and *ping* or *wget* to perform the Mobility at the application layer.

With Mobility and Multihoming operations, performances of the various configurations are evaluated with time durations. One point is to measure the time on our platform, the other is to evaluate how it impacts the End User experience. In fact, Mobility and Multihoming involves multiple operations like Interface change detection, communication to the peer of the Interface change and modifications of the IPsec stack. Thus, to check how updating the IPsec stack impacts the End User experience, we need to add System time to network Time. System time may be optimized by the system, and network time depends on the network environment the MN is located. Our experimental platform uses Ethernet, and to evaluate the time on other network environment we performed some downloads over different environments. The main characteristic we measured on each network is the Round Trip Time (RTT), that provides an indication on the network latency. The various network environments we considered are:

- Public ISP WLAN (*Public HotSpot*), which corresponds to a MN connected to a Public HotSpot provided by the ISP.

- Home WLAN with 1Mbits ($HWLAN_{1Mb}$) and Home WLAN with 10 Mbits ($HWLAN_{10Mb}$), which corresponds to MN connected to their DSL boxes. The scenario addresses the End User switching a communication from 3G / 4G to its own box, or a End User connected to DSL boxes that does not belong to him. The latest case addresses the "WiFi Communities" use case.

- $LabPlatform_{Ethernet}$, which corresponds to our experimental platform.

RTT are evaluated using a TCP connection, but we also compared those values to various exchanges, and the Network Time we measured are:

- T_{SCTP} the necessary time to establish SCTP session.
- T_{IKE} the necessary time to establish an IKEv2 session.
- T_{RC} the necessary time to perform a Return Routability Check exchange. This exchange is used, for example when the MN is changing its IP address. In MOBIKE, the CN or the SG thus wants to check that MN is still reachable on the new IP address and that the update notification comes from the MN.

The System Time we measured are:

- T_{SYS} the time it takes to the system to detect a modification on an Interface and send an announcement to the peer.
- T_{UPDATE} the time it takes to perform the IPsec update and send a notification back to the initiator.

Finally, the Time we measured to evaluate how IPsec impacts the End User experience is:

- $T_{STALLED}$, the time the connection is stalled.

Note that measuring how IPsec impacts the End User experience must be evaluated according to the application. Our measurements only consider the End User experience on a network point of view. In fact some applications like P2P downloads do not really care about interruptions of even few minutes. Some applications sensitive to interruptions have their own recovery or buffering mechanisms that minimize the network interruptions. Other applications that deal with Real Time Applications like Voice over IP, live video streaming, or interactive gaming applications are very sensitive to network interruption, added latencies. In this chapter, we are interested in these types of applications.

Section 5.4 provides measurements when MOBIKE is used. Unlike section 5.2, Mobility results from IPsec modifications changes. Since SCTP Mobility or Multihoming is not required with MOBIKE, we used a regular TCP connection. One reason is that SCTP and the Tunnel mode interacted in unexpected ways, the other is that MOBIKE has been optimized with TCP, and so the measured stalled time values could be considered as the minimum stalled values for Hard Handover. MOBIKE stalls the communication for around $300\,ms$ when a Mobility is performed. With Multihoming, one should add a $45\,ms$ for the OS to detect the Interface is down. In order to avoid the $45\,ms$ Interface detection, Connection Managers are encouraged to perform Mobility before the link is down, rather than relying on Multihoming. Reducing stalled time below $300\,ms$ requires additional mechanisms than Hard Handover Mobility, and we propose Soft Handover with MOBIKE-X.

Finally we measured the performances of our MOBIKE-X implementation. We limited the measurement to the Transport mode and Mobility. The reason is that our implementation has to face the issue of selecting the proper interface when Multiple Interfaces operations are expected. Measuring $T_{STALLED} \approx 264\,ms$ is between 9.3% and 15.6% faster than MOBIKE. This is probably due to the use of the Transport mode which requires fewer interactions with the routing indirections of the Kernel.

In the remaining of this chapter, we note MM for Mobility and Multihoming operations. The content of this chapter provided inputs to [MPH+12b].

5.2 Testing Platform

Our MOBIKE-X implementation is based on *strongSwan 4.3* [str] and we measured MN performances in various configurations for transport protocol (traditional TCP and SCTP) and IPsec: ESP (with aes128-sha1) and ESP_NULL (ESP with sha1 and null encryption).

Our experimental platform is shown in figure 5.1, and we used SCTP to perform MM operations, when we were not using MOBIKE. We used SCTP [Ste07] because that is the most advanced IPv4 protocol that provides End-to-End MM mechanisms. Furthermore, SCTP can be implemented in the Kernel with stacks like *LKSCTP* [LKS] or with user land libraries like *sctplib* [sct]. With kernel implementation, the ISP provides Smartphones with Multiple Interfaces facilities, whereas with user land implementation, the ISP has the opportunity to develop specific SCTP applications even on terminals that are not SCTP enabled. Another advantage is that *sctplib* is provided both for UNIX and Windows OS.

To measure MM performance over SCTP (*LKSCTP-2.6.28-1.0.10* [LKS]), we developed a SCTP client and server that runs on *Fedora 17 Linux OS 2.6.38-rc7* patched for enabling ASCONF [SXT+07] with *fastmsctp-2.6.34-rc5.patch*. The ASCONF patch makes SCTP to dynamically configure its interface. More specifically, SCTP has been designed for Multihoming, but all

Network Type	Data BR (kb/s)	RTT (ms)
Public HotSpot	1021.30 [32.37]	15
$HWLAN_{10Mb}$	10199.42 [1526.35]	2.14
$HWLAN_{1Mb}$	2131.87 [13.07]	9.466
$LabPlatform_{Ethernet}$	34715.04	0.355 [0.289]

TABLE 5.1: Experimental Measurements for Data Bit Rate on Operational RAN and WLAN Networks (mean [standard deviation])

interfaces are provided in the SCTP connection establishment. If an SCTP peer wants to dynamically ADD or REMOVE and Interface, then both SCTP peers need to be patched with ASCONF. The SCTP client can have up to three different Ethernet interfaces that are connected to the server via a router.

The router runs *dummynet* on *Ubuntu Linux OS 2.6.28-11-generic*. We used *dummynet* so to be able to model different types of network. However, changing the bandwidth and delays strongly affected how LKSCTP detects modifications on the interfaces. Default configuration is provided for Ethernet links.

Our testing platform uses Ethernet connection. However offloaded Mobile and Multihomed Node are connected to WLAN Networks, with a higher latency. This means that when exchanges are required, they may take longer in WLAN situation than the measured duration in an Ethernet configuration. In order to evaluate the performances over a realistic situation, and estimate how the EU experience is impacted, we need to consider network latency of WLAN. We measured different WLAN features (*Bit Rate* and *RTT*) with a FTP download over a day: Public ISP WLAN (*Public HotSpot*, $RTT = 15\,ms$, $Data\ BR = 1021.30\,kbs^{-1}$) Home WLAN with 1Mbits ($HWLAN_{1Mb}$, $RTT = 9.466\,ms$, $Data\ BR = 10199.42\,kbs^{-1}$) and Home WLAN with 10 Mbits ($HWLAN_{10Mb}$, $RTT = 2.14\,ms$, $Data\ BR = 2131.87\,kbs^{-1}$). On our $LabPlatform_{Ethernet}$ $RTT = 0.355\,ms$, $Data\ BR = 34715.04\,kbs^{-1}$.

Performance measurements use the candle stick representation to represent the quartiles of

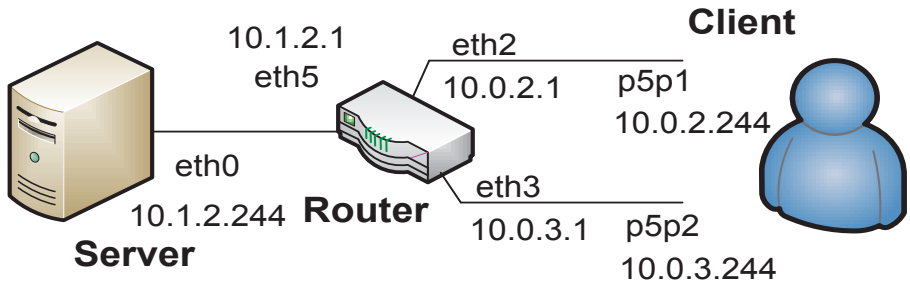


FIGURE 5.1: MOBIKE(-X) Experimental Platform for Mobility and Multihoming Performance

the measured values [Wik].

5.3 SCTP Mobility Multihoming with IPsec

This section analyses how SCTP MM operations are impacted by IPsec. In other words, it measures how protection with Transport or Tunnel affects SCTP Mobility. SCTP can be used both with

Transport and Tunnel mode. The MN is attached to multiple WLAN Access Points to prevent Access Point failure. With Transport mode, the MN has multiple connections with the Service protected with the Transport mode. With the Tunnel mode, the MN has multiple connections to the Security Gateway protected with the Tunnel mode. All links have been secured with IPsec before the Mobility occurs. In that sense, the IKEv2 negotiation and the MN authentication does not impact the Mobility. Mobility is triggered by the Multihoming SCTP mechanism, that is when the Primary interface is down, it switches to the Alternate Interface with a Hard Handover. To compare the various configurations, we measure and compare various time (T_{SCTP}, T_{IKE}, T_{SYS} and $T_{STALLED}$). As a result, we show that MN is more reactive with the Transport mode than with the Tunnel mode: with Transport mode, the MN detects network changes 2.9 times faster —T_{SYS}—, and the Mobility is 2.5 times more stable, and 15% faster. Transport mode provides a clear advanatge over the Tunnel mode.

For all tests, both interfaces $p5p1$ (10.0.2.244) and $p5p2$ (10.0.3.244) are configured with various IPsec configurations. Then, the SCTP communication is initiated with both interfaces. Interface $p5p1$ is used a primary interface by SCTP. After a while $p5p1$ is down, and the traffic switches to $p5p2$. Because our SCTP client and server are ASCONF enabled, after around $T_{SCTP\ ASCONF} = 0.46s$, the client sends an ASCONF that requests $p5p1$ to be removed from the SCTP association.

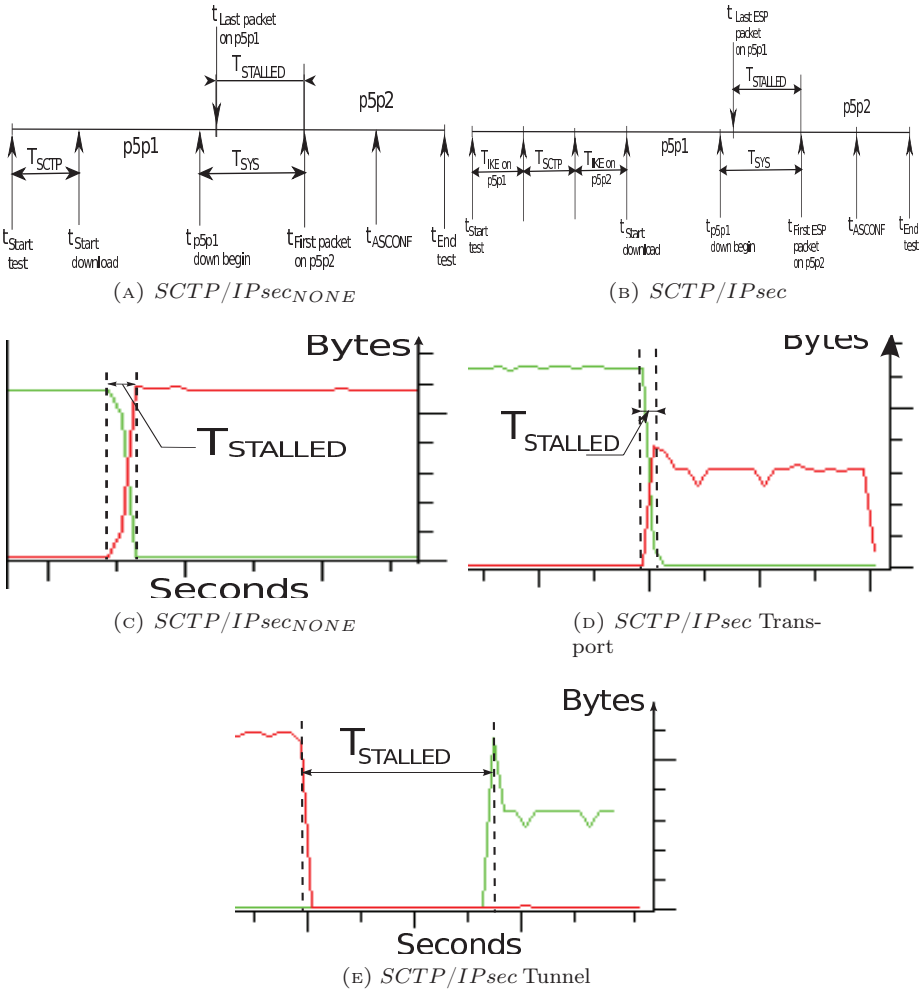
5.3.1 General Input / Output Graphs

Figure 5.2a, (resp. 5.2c) represents the flowchart of MM operations without IPsec protection (NONE), (resp. measured output). Figure 5.2b (resp. 5.2d 5.2e, 5.2d) represents flowshart (resp. measured output) where connections are IPsec protected.

Figures 5.2c, 5.2e and 5.2d show that IPsec is not transparent to the transport layer throughput and behavior, and SCTP may be configured differently for IPsec protected connections than for NONE IPsec connections. With a NONE IPsec configuration —figure 5.2c—SCTP instantaneously uses the whole bandwidth. On the other hand, the Transport mode generates a bandwidth gap when a mobility occurs —figure 5.2d, that is recovered after roughly $10\,s$. With Tunnel mode, —figure 5.2e—SCTP and IPsec encapsulation requires modification of the routing policies, which results in concurrent updates between SCTP and IKEv2. More specifically *p5p1 down* triggers a kernel event for both *LKSCTP* and *strongSwan*. Since *LKSCTP* is kernel based, it updates the routing policies first, followed by IKEv2. This may lock, and delay routing policy stabilization. This makes Tunnel mode more intrusive than Transport which may result in stalling the communication whereas Transport modification may be compensated by transport layer mechanisms.

5.3.2 Measured Time Definition: T_{SCTP}, T_{IKE}, T_{SYS} and $T_{STALLED}$

Figure 5.3 represents the various negotiations involved in the secured SCTP communication: T_{IKE}, T_{SCTP}, T_{SYS} and $T_{STALLED}$ for various IPsec configurations (NONE, ESP_TRANSPORT, ESP_TUNNEL, ESP_NULL_TRANSPORT, ESP_NULL_TUNNEL). T_{IKE} (resp. T_{SCTP}) is the negotiation time for an IKEv2 (resp. SCTP) communication. T_{IKE} is subject to multiple variations especially because it includes an authentication. In our case, we used a preshared key for authentication, but common ISP SIM/AKA authentication requires EAP [ETS10] framework (EAP-SIM [HS06], EAP-AKA [AH06]), which adds the number of exchanges as well as authentication operations. As a result, the authentication part of the exchange may take longer than the one we measured. However, in that case, it only delays the initialization of the communication, and does not impact MM operations. T_{SYS} is the time it takes to the system to detect $p5p1$ is down and starts sending on $p5p2$ which informs the server multihoming occurred. By receiving a message from $p5p2$, the server is informed that a multihoming operation has occurred. $T_{STALLED}$



(A) $SCTP/IPsec_{NONE}$ (B) $SCTP/IPsec$

(C) $SCTP/IPsec_{NONE}$ (D) $SCTP/IPsec$ Transport

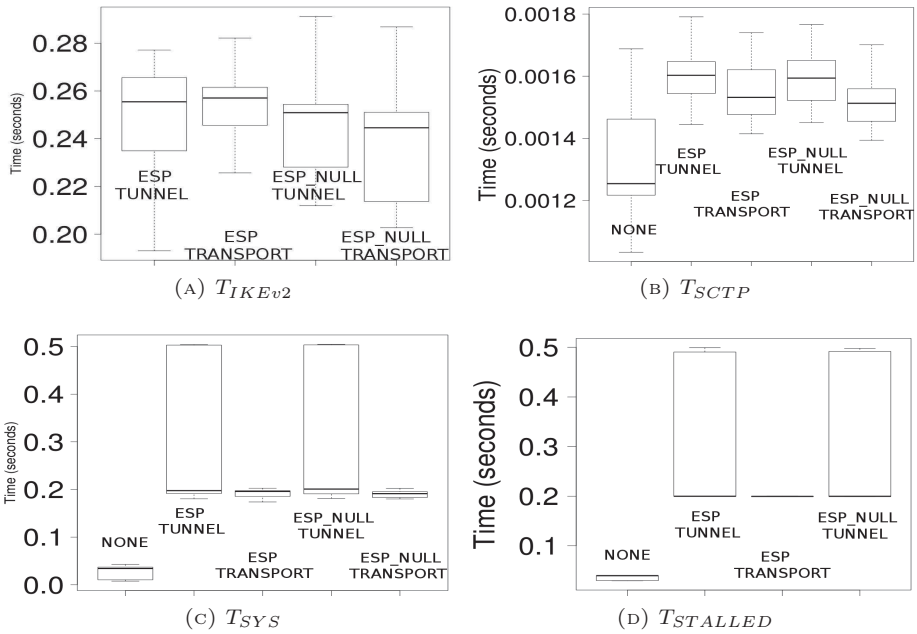
(E) $SCTP/IPsec$ Tunnel

FIGURE 5.2: Multihoming SCTP - Network Flow for Mobility performance with SCTP over IPsec protected links vs non IPsec protected links

is the time duration the communication is interrupted.

5.3.3 T_{IKE} Analysis

Figure 5.3a shows that although Transport mode includes an added Notify Payload, the IKEv2 negotiation for all IPsec configuration (ESP/ESP_NULL, Transport / Tunnel) are between 0.25 s and 0.26 s. From section 5.2, measured network latencies are negligible on our $Lab\,Platform_{Ethernet}$ ($\approx 2 \times 0.355\,ms$), as well as our pre-shared key authentication. Thus, the measured time reflects the systems configurations (SAD, routing policies...).



(A) T_{IKEv2} (B) T_{SCTP}

(C) T_{SYS} (D) $T_{STALLED}$

FIGURE 5.3: Experimental Measurements of IPsec Mobility performance with SCTP over IPsec protected links vs non IPsec protected links

5.3.4 T_{SCTP} Analysis

SCTP negotiation without IPsec takes $1.2545\,ms$ —figure 5.3b. Compared to $2.RTT = 0.71\,ms$, it takes roughly $0.54\,ms$ to set the network stacks. IPsec adds a $0.26\,ms$ overhead for Transport mode vs $0.35\,ms$ for Tunnel mode. Compared to RTT delays introduced by IPsec at the connection initialization should not impact the EU experience. However, the delay introduced by IPsec between 21 and 27% may impact the servers. ESP versus ESP_NULL has no impact on T_{SCTP}, but Tunnel adds a delay 6% higher than the Transport for a mixed transaction of packets between 62 bytes (COOKIE_ACK) and 348 bytes (INIT_ACK).

Finally, initialization times measures how long the connection is delayed. IPsec negotiation delays the communication by at least $0.25\,s$ which is not negligible even for MN connected to *Public HotSpot* with $2.RTT = 30\,ms$. However, the EU experience may not be affected, since it occurs only at the initialization phase, then this delay may be avoided with pre-authentication. Similarly, the delay introduced by IPsec for the SCTP negotiation, is not significant for the MN. If the MN is connected to a *Public HotSpot* the delay is between 1.73% and 2.23% of the RTT, which makes it negligible, mostly because it happens only once. However, SCTP initialization exchange provides an example of small packet exchanges and shows that Transport mode reduces the security overhead by 6% over the Tunnel which makes Transport mode more efficient for offloading RTA.

5.3.5 T_{SYS} Analysis

In figure 5.3c IPsec overhead for T_{SYS} is between $161.85\,ms$ for Transport and $163.48\,ms$ for Tunnel. Tunnel presents large variations, and the added delay is up to $469.991\,ms$. Transport mode makes the System more reactive and stable. The IPsec overhead is the time to activate the dor-

mant SA and modify the routing tables. Note that in Transport, *strongSwan* is configured with the option `install_routes=no` so it does not interfere with the routing tables. With Tunnel, this option cannot be used, which makes Transport between 1.01 and 2.90 times faster, and so preferred for offloaded RTA. IPsec clearly makes the system less reactive, and delays introduced by IPsec impacts the EU experience. This can be avoided either by tuning the system with IPsec, and most probably makes various IP/IPsec/transport layer communicating between each other, or by anticipating a connection is down. In fact connection Managers are expected to decide to switch on one interface before the running interface is down.

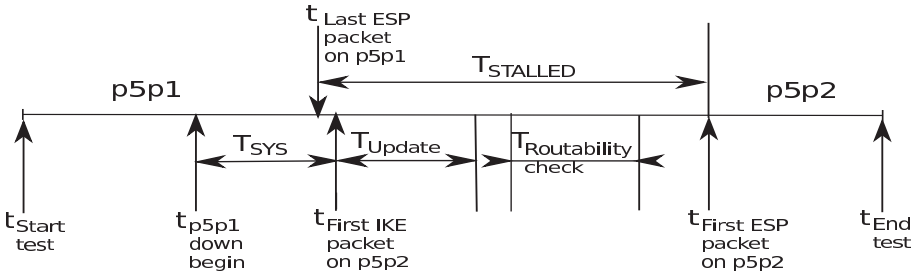
5.3.6 $T_{STALLED}$ Analysis

Figure 5.3d shows that without IPsec, the communication is stalled for $30.0325\,ms$ and $199.587\,ms$ with IPsec, which represents the necessary time for the system to detect events, as well as to make SAD and SPD operational. Similarly to figure 5.3c tunnelling requires system interactions with routing tables which result in large variations, stalling the communication up to $499.20\,ms$. IPsec security overhead results in a longer stalled communication, and clearly impacts the EU experience. Note that IKEv2 Mobility exchanges are not considered in this section. If so, an exchange would add another $RTT = 15\,ms$ on an *Public HotSpot*. The stalled time may be improved and reduced by scheduling transport and IPsec stack modifications, as well as by anticipating and allowing Multihoming Simultaneous Interfaces for a given communication.

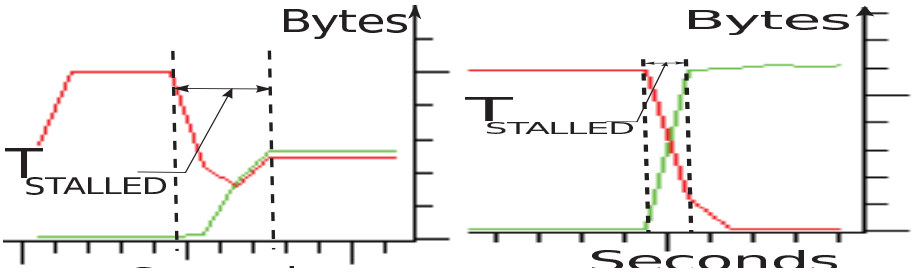
5.4 MOBIKE Mobility Multihoming

MOBIKE only considers the Tunnel mode, and a single interface. Thus switching interfaces is performed through a Hard Handover and an UPDATE_SA_ADDRESSES Notify Payload indicates the new IP addresses to use. In this section we use two different mechanisms to switch from one interface to the other. We designate by *Mobility* the operation that consists, for a MN with a single interface, in changing manually the IP address of the running interface *ifconfig p5p1* IP_{NEW}. By changing the IP address, the MN sends an UPDATE_SA_ADDRESSES Notify Payload. We designate by *Multihoming* the operation that consists, for a MN with Multiple Interfaces, to manually bring the Primary interface down *ifconfig p5p1 down*. By putting down the Primary Interface, the MN checks the Alternate Interface is still reachable by performing a Return Routability Check (RRC), followed by an UPDATE_SA_ADDRESSES as in the *Mobility* scenario. We consider those two distinct mechanisms because *Mobility* may be trigger by a Network Manager, whereas *Multihoming* is a mechanism that recovers from WLAN Access Points Failover.

Figures 5.4a and 5.4b (resp. 5.4c) give MM with MOBIKE (resp. with MOBIKE-X). In figure 5.4b *Wireshark* represents a packet anytime it passes through the IP stack, that is to say for both the inner and outer IP header.

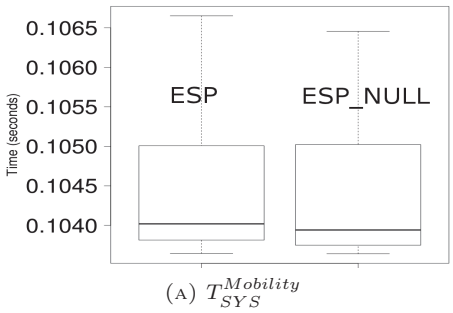


(A) $MOBIKE(-X)$ Mob. / Multih.

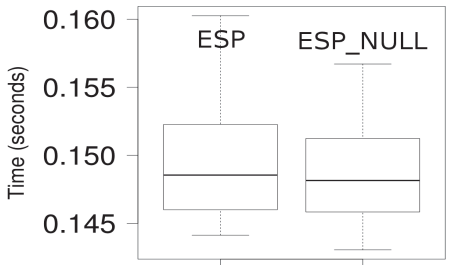


(B) $MOBIKE$ (C) $MOBIKE-X$

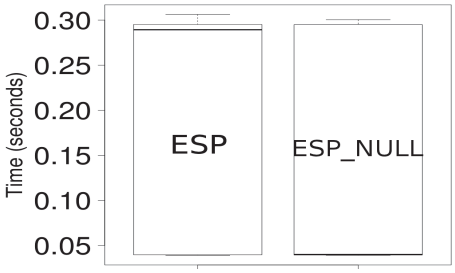
FIGURE 5.4: MOBIKE / MOBIKE-X - Network Flow for Mobility performance with MOBIKE and MOBIKE-X over IPsec protected links



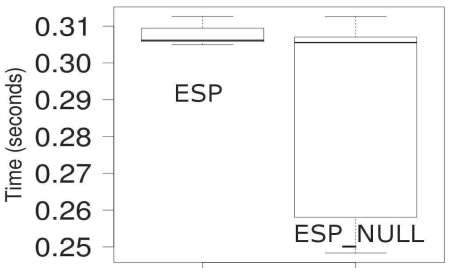
(A) $T_{SYS}^{Mobility}$



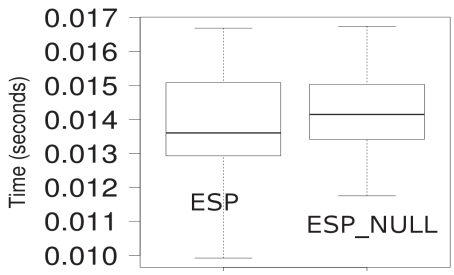
(B) $T_{SYS}^{Multihoming}$



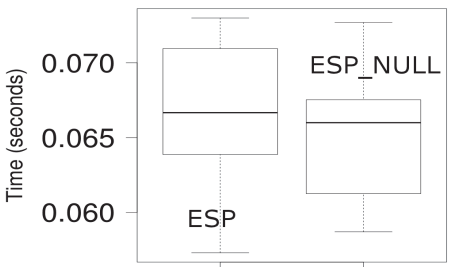
(C) $T_{STALLED}^{Mobility}$



(D) $T_{STALLED}^{Multihoming}$



(E) $T_{UPDATE}^{Mobility/Multihoming}$



(F) $T_{Routability\ Check}^{Multihoming}$

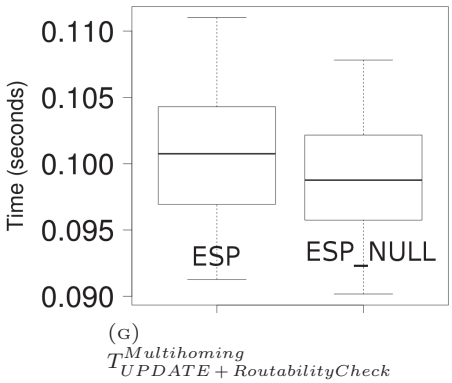


FIGURE 5.4: Experimental Measurements of MOBIKE Mobility performance with MOBIKE

Figures 5.5a and 5.5b compare T_{SYS}, the time required by the system to trigger *Mobility* or *Multihoming*. With *Mobility*, the MN triggers the change of IP addresses, and the configuration of both network and IPsec stack takes 103.71 ms. With *Multihoming* it takes an additional 44.48 ms for the OS to detect the interface is down.

Furthermore, *Multihoming* requires the Return Routability Check (RRC) exchange which adds a 16.61 ms delay to the $T^{Multihoming}_{STALLED} = 305.9345\,ms$ versus 289.318 ms for *Mobility*. We measure $RTT = 15\,ms$ with FTP download on *Public HotSpot*, which makes $T^{Mobility}_{STALLED} \approx 303.96\,ms$ and $T^{Multihoming}_{STALLED} = 335.234\,ms$.

Comparing RTT (0.35 ms), T_{UPDATE} (13.60 ms) and T_{RC} (66.011 ms), T_{RC} is 4.85 times larger than T_{UPDATE} because kernel operations are performed with higher priority than application (polling mode). From $T_{UPDATE} \approx 13.60\,ms$, the IPsec SADs are expected to be updated on the MN and the server in roughly 25 ms. $T^{Mobility}_{STALLED} \approx 289.318\,ms$ because not only SADs must be updated, but also routing policies.

This confirms IPsec configuration time derived from figure 5.3a to create a SA. Furthermore, comparison between *Mobility* and *Multihoming* shows how Network Managers may improve the EU experience by performing a Mobility and thus avoiding the RRC exchange. Furthermore, Network Manager may improve further the EU experience by preparing the Mobility and performing a Soft Handover rather than a Hard Handover. Hard Handover results in a 300 ms stalled communication whereas Soft Handover is expected to no interruption at all. On the other hand, Soft Handover requires to handle Multiple Interfaces which requires MOBIKE-X extension.

5.5 MOBIKE-X Mobility Multihoming

MOBIKE-X extends MOBIKE for the Transport mode and Multiple Interfaces, which enables Soft Handover. Soft Handover provides the ability to change interface without losing any packets. In moving to a new Interface with Hard Handover discards that are on the Network between the time Hard Handover has been started and the time the Server starts sending on the new interface. From measurements in figure 5.5h, we estimate that discovering the new interface and starting the IPsec update takes around $T_{SYS} = 110.9095\,ms$, but it may take more time to configure it for example if authentication to the new Network is required and the IP addresses is obtained via DHCP. Such delays may not impact the communication if the MN has Multiple Interfaces. If the MN has a single interface, those delays must be added to $T_{STALLED}$.

Our MOBIKE-X implementation always performs Routability Checks, which, for Mobility op-

eration, may be avoided. Thus $T_{STALLED} \approx 264\,ms$, which is between 9.3% and 15.6% faster than MOBIKE.

With Transport mode $T_{UPDATE} = 36.6035\,ms$ is 2.69 times larger than with the Tunnel mode because SAD and SPD whereas Tunnel mode only updates SAD. With Soft Handover time and delay is less critical, it delays slightly the Handover, but does not results in loss of packets. On the other hand, T_{UPDATE} is around $2.RTT$ when the MN is offloaded in a *Public HotSpot* which may not affect greatly the EU Experience.

$T_{Routability\ Check} = 39.943\,ms$ is smaller than with MOBIKE. RRC is not different from MOBIKE, and one way to explain the difference is to consider the testing conditions. With MOBIKE-X, we measured duration with the ping application whereas MOBIKE has been tested with a TCP connection. *pings* probably do not fill the NIC buffer as SCTP/TCP packets do. We used pings because SCTP does not support mobility operation, that is changing its interface IP address.

Measurements confirm previous results. Using Transport mode reduces interactions with the Network stacks and communication overheads. In fact Transport avoids tunneling, and do not need the tunnel IP header. As a result, Mobility is performed faster, changes are detected faster by the system —for example when Multihoming is performed. This provides competitive advantages for the Transport based architecture compared to Tunnel. This chapter also shows that Hard Handover always results in degrading the EU experience. However optimized, changing the IP address of a given communication always requires inter-process communications with their own latencies. One way to reduce the Mobility impact on the EU experience is to perform Soft Handover. MOBIKE-X provides this facility, which can be used both with Tunnel and Transport mode.

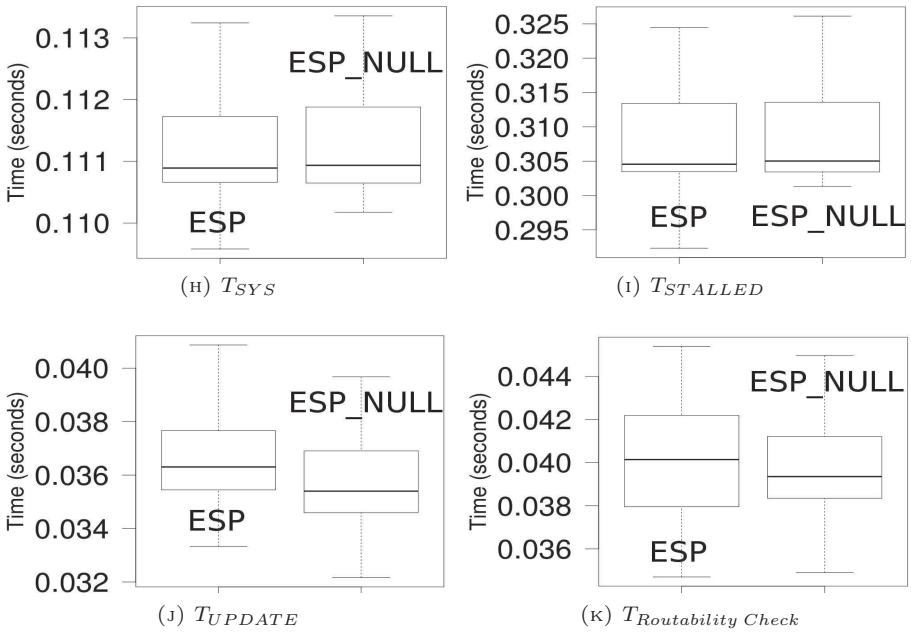


FIGURE 5.5: Experimental Measurements of MOBIKE-X Mobility performance with MOBIKE-X

5.6 Conclusion

In the chapter, we measured how IPsec impacts communications when Mobility or Multihoming is performed. From network measurements, we compare Mobility and Multihoming with different IPsec transport mode and different ways to perform Mobility or Multihoming. In this chapter, Mobility and Multihoming is performed by the IPsec layer with MOBIKE(-X) and the Tunnel mode, by the transport layer with SCTP and by the application with a ping based application.

By measuring network time we derive how the End User experience is affected by a Mobility or a Multihoming over an IPsec protected communication. In this chapter, we did not consider mechanisms provided by the application (like buffering or recovery mechanisms) that may counter the communication interruption. Indeed, we are considering applications that do not have such mechanisms and whose End User experience directly relies on the network layers variations. Such applications are Real Time Applications, including gaming and VoIP applications.

First, our measurements show that securing a communication with IPsec clearly impacts the Mobility and Multihoming mechanisms, at least with SCTP. Using IPsec results in a 200 ms stalled time whereas non IPsec Mobility only stales the connection around 30 ms, which results in a 600% overhead! As a result, the security costs must be balanced with the End User experience, and it is recommended to secure communications when the Network environment cannot be trusted. In other words, when the End User is connected to trusted WLAN Access Points, for example trusted ISP DSL Boxes, relying on the Radio Layer Security may be recommended if the provided security is sufficient.

When IPsec security is required, the advantages of using IPsec Transport over Tunnel mode are:

- Transport mode reduces network processing complexity and cryptographic operations. This significantly reduces the number of CPU cycles ($\approx$ 25% for cryptographic computation).

- Transport mode makes the system more reactive. It detects interface changes around 2.9 times faster
- Transport mode makes the system more stable and presents 2.5 times less variations for both down interface detection (T_{SYS}) and stalled communications ($T_{STALLED}$) (cf. section 5.3).
- Transport mode results in a Mobility performed around 15% faster (section 5.3). On the other hand, Tunnel mode with more complex routing configurations may result in stalling the communication for few seconds (figure 5.2e). MOBIKE shows that specific configurations can partly overcome this issue, but this is done at the expense of layer / process independence. Finally Transport mode optimizes MM for secure offloaded communications.

On the other hand, the main advantage provided by the use of the Tunnel mode is that it can provide Mobility and Multihoming to applications or transport layers that do not provide such facilities. As a result, the drawbacks of using the Transport mode are:

- Transport mode does not provide Mobility or Multihoming facilities for the communication. IPsec Mobility only provides a proper configuration for the IPsec layer.

- Transport mode must be combined with upper layers that are Mobility or Multihoming aware. This can be done by the transport layer with protocols like SCTP or MPTCP. We have not seen much deployment of SCTP for the End Users applications, and more commonly, Mobility or Multihoming is handled by the application itself. This is at least the case for HTTP and FTP applications.

As a result, Securing Real Time applications with IPsec would favor the Transport mode over the Tunnel mode. Most probably, strong Real Time requirements will result in a dedicated IPsec Transport architecture, mainly to avoid traffic congestion and avoid traffic indirection. Even though Transport mode optimizes MM compared to the Tunnel mode, to reduce drastically MM, the MN may anticipate MM operations and prefer Soft Handover to Hard Handover as performed by the current MOBIKE. In fact, Multihoming relies on system interface detection and requires further network verifications such as Return Routability Check which stalls the communication around 5.7% longer than Mobility. As a result, the Network Manager is encouraged to perform Mobility operations rather than relying on failover mechanisms like Multihoming.

During this experimentation, we found out that multiple layers (IPsec, SCTP...) interact with MM. Although they have been designed to work independently, implementations do have strong interactions. We found a significant interest in specifying interactions between the different layers, and our current research includes the design of an IPsec API that would make possible applications and SCTP to take advantage of IPsec features (mobility, multihoming, authentication...).

Chapter 6

MOBIKE-X & Offload

6.1 Introduction

Mobile data traffic for Smartphone is expected to be 47 times higher by 2015, which makes upgrading the current 3G / 4G Radio Access Network (RAN) much too expensive. To overcome this issue, Mobile Network Operators (MNO) are seeking to transfer this traffic on alternate networks: WLAN. This is problematic is currently known as Offload and is addressed in this chapter.

RAN and WLAN are two different technologies that have been designed with different goals, for different use cases and with different economic perspectives. To make it simple, people are subscribing to RAN access for a set of service provided by the Operator —the main service used to be voice —, where WLAN access was mainly deployed to provide Internet access for End User in a given place like a campus, a hotel, a station. Hence, convergence between RAN and WLAN comes with a few challenges to overcome.

Section 6.2 provides an economic view on Offload. It describes the economic advantages of offloading RAN traffic to WLAN as well as the new business relations, and new actors the WLAN access introduces. In fact, it is (almost) impossible for an MNO to cover a whole national area with WLAN Access Point as they used to do with RAN Access Points. WLAN Access is much more localized to a place like an airport, or a hotel, which makes MNOs deal with new actors like WLAN Operators, or WLAN aggregators.

The remaining sections of this chapter are focused on technical aspects, and propose Security Architectures for Offload. As, we have just seen before, with multiple new actors and WLAN Operators, the End User of an ISP or Operator can be attached to a WLAN that does not necessarily belong to the ISP the End User has subscribed. In other words, the IP network may not be trusted and the ISP must provide its end user a secure way to access the service. Because the End User cannot rely on Layer 2 security, in this chapter we adopted layer 3 security with IPsec. The architectures we proposed in section 6.3 are Offload Service Architecture (OSA) and Offload Access Architecture (OAA). OSA consists in securing with IPsec the communication to a given service. In that sense, it is equivalent to the use of TLS. The main advantage over TLS is that the communication can be secured according to the level of trust of the network, and that IPsec with MOBIKE-X can deal with Mobility, Multihoming and Multiple Interfaces. On the other hand, OAA proposes the traditional Security Gateway architecture that tunnels the traffic to a Trusted Entry Point of the ISP network. The main advantage of OAA, is that the architecture can be global to multiple services, whereas OSA is dedicated to a single service. The advantage is provided at the expense of routing indirection, and a Tunnel overhead. Section 6.3 provides

an in-depth comparison of these two architectures and defines what is addressed by each of the architecture.

OSA and OAA are two ways to handle the offloaded traffic, and section 6.4 positions these architectures toward OSA and OAA. One example of alternative architectures is this with HIP. Section 6.5 defines how ISPs can take advantage of OSA, OAA to offload their end user traffic. In combination to OSA and OAA, ISPs may also use FWDA, the ForWarD Architecture that consists in forwarding the End User Traffic that does not require any protection. This section defines how ISP should combine these three architectures to properly offload their End User Traffic.

Once the ISP architecture has been defined, there are multiple ways to implement it. Section 6.6 proposed two alternatives, that are cost efficient. One is based on SCTP and MOBIKE-X, the other one is only using MOBIKE-X. The two implementations are described in this section. Performances of those two architectures, in a Mobile, Multihomed, and Multiple Interface environment are provided in section 6.7. In addition, based on performance measurements, this section provides inputs for ISPs to deploy their Offload architecture.

This chapter provided inputs for the paper [MPH+12a].

6.2 Offload Economics

6.2.1 Increasing Demand for Mobile Data

With raising popularity of tablets and M2M applications, mobile data increase by 92% per year, and will reach $6.3\,10^{18}\,bytes.month^{-1}$ by 2015 [Cis11]. Mobile-connected tablets will generate in 2015 as much traffic as the entire global mobile network in 2010 - that is to say, $248\,10^{15}\,bytes.month^{-1}$. Similarly Machine-to-Machine traffic is expected to reach $295\,10^{15}\,bytes.month^{-1}$ in 2015. As a result, in 2015, the average Smartphone will generate a traffic of $1.3\,10^{9}\,bytes.month^{-1}$, which represents a 16-fold increase over the 2010 average of $79\,10^{6}\,bytes.month^{-1}$. In other words, aggregate Smartphone traffic in 2015 will be 47 times greater than it is today.

6.2.2 Offloading traffic to WLAN: the only viable solution for ISPs

A large part of the ISPs revenues are provided by Services, and not facing this increasing demand on traffic represents lost of profits. As such ISPs have to make their infrastructure ready to deal with that traffic. To overcome this traffic growth, ISPs have three alternatives [LM03, Han10, RA10, Han09, NL11]:

- **Upgrade their infrastructure** by building the required number of cells.
- **Optimize their infrastructure** by improving the current technology and increasing each cell's capacity
- **Offload** the traffic on Alternate Networks such as WiFi.

To deal with foreseen mobile traffic, ISPs have two options 1) Optimize and Upgrade their infrastructure vs 2) Use an alternate WiFi Network. This section compares the cost of various scenarios and shows that the RAN infrastructure may not require to be upgraded nor optimized if 52% of the traffic growth can be offloaded, and that deploying indoor WiFi Access Points reduces

the costs by 4.8 over the Optimize and Upgrade scenario.

Radio access optimization and bandwidth optimization represents the fields to improve the global efficiency of a cell. Radio access may be optimized by migrating the HSDPA infrastructure to HSDPA+ and then to LTE. Migration from HSDPA to HSDPA+ increases download bandwidth from $14.Mbits.s^{-1}$ to $21Mbits.s^{-1}$, and migration to LTE is expected to increases by up to 4 times the HSDPA performances. Then, bandwidth optimization may increase the coverage of each cell which provides flexibility for the ISP to design its network. ISP can reduce the number of cells of under loaded areas and increase the number of cells of heavy loaded areas. In fact the area of the old GSM $800MHz$ is three times larger than with a $2.1GHz$ cell. However optimization increases the cell capacity by around 4, and so does not present a long term solution to face the forecast 16fold increase of traffic. According to the forecasts traffic growth, ISP must either optimize and upgrade their infrastructure or offload the traffic to an alternate network.

[NL11] evaluates the costs of three scenarios to deal with an eightfold traffic increase. [NL11] assumed that 95% of the labtop where used indoor (resp. 5% outdoor), and that 70% of the Smartphones and tablets were used indoor (resp. 30% outdoor). In conjunction to the great cost difference between indoor and outdoor WiFi equipments, [NL11] considered the following scenarios:

LTE-only : Consists in optimizing and upgrading the current ISP infrastructure from HSDPA to HSDPA+ and then to LTE.

indoor-WiFi : Consists in not upgrading the current HSDPA RAN infrastructure and deploying enough indoor WiFi Access Points so that 52% of the traffic growth can be offloaded. Indoor Access Points are localized in sheltered places, which makes the costs of those equipments very low.

in/outdoor-WiFi : Consists in not upgrading the current HSPD RAN infrastructure and deploying both outdoor and indoor WiFi Access Points are used to reach the 52% traffic growth to be offloaded. The number of outdoor WiFi Access Points is defined so there are enough Access Point to cover the whole HSDPA cell.

The *LTE-only* scenario requires optimizing the 3G cells, to build new LTE cells and to buy spectrum. Optimizing does not add opex estimated at $30\,000\,EUR$. However, building a new LTE cell is estimated at 110 000EUR, with an added $31\,000\,EUR$ for the opex. The spectrum cost is estimated at $0.15\,EUR/MHz/pop$. With a population per site $pop = 9600$ and a $20MHz$ bandwidth, the cost is estimated at $28000\,EUR$. As a result, the costs of *LTE-only* scenario are estimated to $600.000\,EUR$ per currently HSDPA existing site.

For *indoor-WiFi*, Access Points hardware is $120\,EUR$, backhaul Network is estimated to $100\,EUR$, but can be reduced if the provided by the ISP with a total of $450\,EUR$ per year and site. Of course, multiple sites are required with a total cost of $124\,910\,EUR$ per currently HSDPA existing site.

The *in/outdoor-WiFi* scenario consists in also deploying outdoor WiFi Access Point so that with a mesh organization, the outdoor Access Points provide an coverage equivalent to a $500\,m$ HSDPA cell. Outdoor Wifi Access Points estimated to $6.185\,EUR$. With a mesh organization, 10 Access Points are required. Hardware costs are around $61\,850\,EUR$, installation costs are $15\,400\,EUR$ and backhaul Network are $7\,950\,EUR$. Similarly backhaul network costs can be reduced if provided by the ISP. Operational and management costs are estimated to $22\,900\,EUR$, with a site rent estimated to $11\,000\,EUR$. Of course this depends of the place. This makes the total cost per 3G site of *in/outdoor-WiFi* is $260\,317\,EUR$.

As a result, the two offload scenarios *indoor-WiFi* (resp. *in/outdoor* scenarios reduces the

costs by 4.8 (resp. 2.3) times over the optimize and upgrade scenario. Despite a great economical advantage, offload architecture comes with an added complexity, that ISPs have to overcome so to take the full advantage provided by the offload architectures.

6.2.3 Offload Complex Environment

Currently, the model for Accessing the Internet has been quite simple. The End User subscribes to a Mobile Network Operator (MNO). This MNO has deployed its network of 3G / 4G Access Points, and the End User uses the Access Points of the MNO it has subscribed to. In case the MNO of subscription cannot provide access, the End User uses another Access Point, and MNOs have roaming agreements between each other. The number of MNOs is quite restricted (around 4 per countries), so the number of agreements to negotiate is quite small.

With WLAN deployment, Access Points cover smaller areas like a bar, a cafe, an airport, an hotel. Such coverage provides the access of on a localized area, which is fine while the End User remains in the same place. This does not match the offload requirements that take advantage of local coverage, but also need to provide coverage between those places. As a result, we have Local WLAN operators that are in charge of covering a place, and Aggregators that have multiple agreements with the multiple local operators providing a huge coverage for End Users and eventually MNOs. The Cloud [The] is an example of Local WLAN provider, covering multiple places all over Europe. In order to extend its coverage its has concluded partnerships with MNO such as Telenor [Tel], Sprint [Spr] AT&T [ATT] as well as Aggregators like iPass [iPa]. Example of aggregators are: iPass [iPa], Boingo [Boi], Trustive [Tru], WeRoam [WeR].

Main MNO have started deploying WiFi, and for example Orange is said to have deployed in 2011 30.000 Access Points. However, the major advantage of this ISP is that it may take advantage of its DSL boxes deployed in its DSL End User customer.

6.2.4 Current Offload Deployments

Verizon Communications chief technology officer Tony Melone announced on TIA 2011, that although VZC is deeply involved in deploying 3G/4G network, it also plans to deploy WiFi networks in stadiums, campuses so to offload mobile data [Fit11]. Furthermore, the main operators like AT&Ts have set a partnership with the hotspot provider Boingo Wireless, and most of the European ISPs (Orange, Free, Vodaphone) have deployed WiFi communities. On the ISPs side, multiple developments have already been performed. AT&T developed WISPr to switch iPhone applications from RAN the Hotspots. Deutsche Telecom concluded an agreement with iPass to benefit from multiple WLANs, KDDI and Ruckus Wireless deployed 10 000 indoors WiFi HotSpots, Republic Wireless provided a hybrid phones that switches to RAN when no WLAN is available, and states that users are offloaded in 60% of the time. Offloading mobile data to alternate WiFi networks represents the most efficient way ISPs can deal with the mobile data increase. However, ISPs' mission is to provide services to the EU, and the offload architectures MUST not degrade nor the Quality of Service, nor the Security of the services provided to the EU. More specifically, WLAN MAY not be trusted, and in this case, the communication must be secured. This adds an overhead that should not degrade the Quality of Service. Real Time Applications are especially sensitive to added overhead which adds latencies.

This part provides alternatives to seamlessly switch from one network to the other, with the

appropriated security, and with the minimum security overhead.

6.3 Offload Service Architecture (OSA) vs Offload Access Architecture (OAA)

When mobile End Users are offloaded from a Radio Access Network (RAN) to a WLAN, there are two ways to secure the communication. On way is to have End-to-End Security between the MN and the CN. E2E Security can be provided both by the Tunnel mode and the Transport mode. In this chapter we consider in that case that if Transport mode can be used, then Transport is used and not Tunnel. Furthermore, in offload scenarios, the MN is most of the time connected to a Service, thus we usually call Offload Service Architecture (OSA) the Architecture that corresponds to a MN connected to the Service with E2E IPsec protected communication. This, of course, requires the ISP to have a dedicated IPsec infrastructure for that Service.

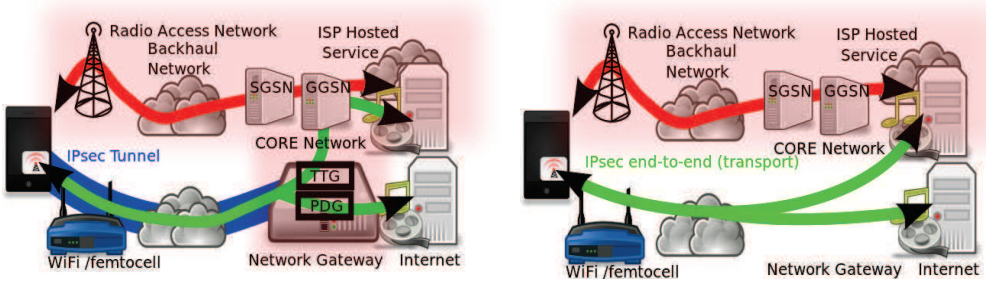
Another alternative to secure the offloaded communication is to secure the communications of the MN until a Security Entry Point in the ISP CORE Network. This architecture is based on a Security Gateway, and the MN tunnels its traffic to the SG. This is the architecture proposed by I-WLAN [3GP11]. Contrary to OSA, this architecture does not provide E2E security to the Service and is not dedicated to a Service. All traffic is tunnelled, and we call this architecture the Offload Access Architecture (OAA).

This section compares the OSA and the OAA Architecture. Section 6.3.1 compares OSA and OAA on an architecture point of view. Basically, it describes the differences between securing a Service to an Access. The architecture comparison should be as independent as possible from the protocols used. Although, the whole thesis is about IPsec, and I-WLAN is using IPsec, one may expect IPsec to be the Security protocol used for OSA and OAA. In fact Section 6.3.2 lists the Security Requirements of OSA and OAA and explains why IPsec is preferred compared to TLS/DTLS for example. Then because Mobility, Multihoming and Multiple Interfaces operations are required, section 6.3.3 explains why MOBIKE is not convenient and why MOBIKE-X is convenient for OSA and OAA. Section 6.3.4 considers the CPU consumption of the Transport mode and the Tunnel mode. Overall, the benefits of OSA are mostly load reduction and a better End User experience. First, OSA offloads the ISP CORE and backhaul Networks, then it uses IPsec Transport mode instead of Tunnel mode, which removes networking and security overhead. This reduces CPU load by 20% (cf. section 6.3.4), enhances Mobility and Multihoming operations by about 15%, and makes the system 2.9 times more reactive for detecting modifications of interfaces (cf part 5). On the other hand, OSA requires a layer that is able to handle Mobility and Multihoming (like SCTP or MPTCP), whereas OAA works with regular TCP connections. At last section 6.3.5 explains why OSA represents a new Service ISPs can propose to third party Service providers, and as such represents a new business opportunity.

6.3.1 OSA and OAA Architecture Comparison

This section compares our OSA to OAA and more specifically the 3GPP I-WLAN Offload Architecture [3GP11].

I-WLAN illustrated in figure 6.1a is the proposed 3GPP offload architecture. A Smartphone connected to a WLAN sets up an IPsec tunnel with the ISP Tunnel Terminating Gateway (TTG) which decapsulates and forwards the traffic. That is, communications with an ISP service hosted



(A) OAA (I-WLAN) (B) OSA

FIGURE 6.1: Description of the OSA and OAA Offload Architectures

application are forwarded to the Gateway GPRS Support Node (GGSN), otherwise Internet communications are forwarded to the Packet Data Gateway (PDG). MM is handled by the MOBIKE extension of IKEv2 [Ero06].

OSA, as illustrated in figure 6.1b, provides End-to-End security. That is, the communication is encrypted from the MN to the server hosting the service. That is, there is no Security Gateway.

Compared to the OSA, OAA (I-WLAN) suffers from the following drawbacks:

- **Cost overhead for the ISP:** The main reason for encrypting and redirecting traffic to a Security Gateway is to protect this traffic. This traffic overloads the ISP CORE Network, and ISPs have to deploy VPN concentrators, platforms and licenses. With current 3G RAN architecture, license costs are derived from Packet Data Protocol (PDP) context activation. When an MN is being offloaded, this requires a new PDP activation which adds unnecessary costs for the ISP. There are at least two types of traffic that are unnecessarily redirected to the Security Gateway: (1) Traffic that is not confidential and (2) traffic already protected like HTTPS for example. Note also that I-WLAN, in addition to the Security Gateway requires also a Mobile IP infrastructure [Per10] to move EU from RAN to WLAN.
- **Latency overhead:** The Security Gateway introduces extra latencies by doing extra processing over the packet (e.g. encapsulation, forwarding), and routing indirection. Traffic tunneling adds network complexity as each packet is forwarded twice in the IP stack. Furthermore, an overhead by at least 20 *bytes* in IPv4 and 40 *bytes* in IPv6, is introduced and leads to extra network load and network latency. Tunneling with IPsec requires extra encryption costs of the inner header. Of course, the smaller the application datagram is, the higher the cryptographic cost is. Section 6.3.4 evaluates the cost of encrypting the inner IP header.
- **Single point of failure :** The Security Gateway where all the traffic is going through is exposed to DoS or DDoS attacks.

On the other hand, the OSA security approach provides the following advantages:

- **Per service granularity:** The ISP secures only the services that need to be secured.
- **ISP Network load reduction:** End-to-End communications are not redirected to the CORE network of the ISP, and eventually not even on the Access Network nor the backhaul Network of the ISP. We use SCTP [Ste07] in conjunction of IPsec so that Mobility is handled by the Terminal, and does not require the ISP to deploy any infrastructure like with Mobile IP [Per10]. Similarly to OAA, OSA needs a Mobility protocol to move from RAN to WLAN.

However, IPsec Transport mode cannot be used for moving the traffic on its own and OSA requires a Mobility protocol as SCTP.

- **Security overhead reduction:** Avoiding some useless traffic encryption, smaller clusters of VPN concentrators are necessary for deploying the Security Gateway. Using Transport mode rather then Tunnel mode increases the capacity of each concentrator by reducing both the cryptographic and the network load.
- **Latency reduction:** Both in term of network latency and routing indirection.
- **Provision of new services:** Security can be considered as a service for third service providers (see section 6.3.5).

As a result, for a given traffic OSA requires a smaller infrastructure than OAA, and provides a better EU experience, especially for RTA with small datagram. In this chapter, we compare OSA and OAA and measure how OSA provides a better EU experience. However, OSA is not expected to replace OAA, we expect ISPs to deploy OSA for high value Services, and OAA for the remaining traffic of the EU.

6.3.2 IPsec: the Security Protocol for OSA OAA

In order to choose properly the Security protocol, this section lists General requirements the protocol must fulfil, followed by specific MM requirements.

- **Granularity:** With E2E, the traffic that is secured depends on the Service, the level of trust of the Network, so we must be able to define SP using selectors as IP addresses, ports, application protocols.
- **Security Layer:** With E2E, a Service Provider must be able to request the ISP to secure its traffic over untrusted networks like WLAN. The way the ISP secures the Service should be transparent for the Service Provider. In that sense TLS, for example, requires to modify the source code of the application.
- **Architecture:** E2E and I-WLAN are complementary Architectures. E2E addresses traffic of a specific service whereas I-WLAN address other traffic. For a given service, an ISP may start to use I-WLAN, and then evolves to E2E. It is thus recommended to use the same Security Protocol for E2E as the one used for I-WLAN.
- **Authentication:** Offload Security should support similar authentication mechanisms from the WLAN and the RAN, for homogeneous network access. This would provide the opportunity for an EU to initiate a connection directly from WLAN, rather than from RAN before being offloaded.

This list can be enriched with the MM Security Requirements of [Dan09a]:

- **Mobility:** A MN must be able to UPDATE the IP address of its interface.
- **Multihoming:** WLAN Access Point may not be maintained by the ISP, and so may be unreliable. The MN must be able to provide alternate IP addresses that may be used if the running IP address is not reachable anymore.
- **Multiple Interfaces:** Similarly, the MN may be attached to various WLAN Access Points simultaneously. The MN should be able to ADD, REMOVE or UPDATE an interface to a given communication.

Comparing TLS [DR08] / DTLS [Phe08] and IPsec shows that IPsec [KS05] is recommended to Offload Security. TLS/DTLS does not provide other granularity than a service granularity (port). In other words, DTLS/TLS provides a secure version of a given service. Moreover TLS/DTLS's main drawback is that it requires code modifications, and thus makes ISP Offload service of section 6.3.5 hard to be deployed for third party. Furthermore, TLS/DTLS has been designed for

End-to-End connectivity, and may not fit all requirements of a Security Gateway Architecture. At last, TLS/DTLS does not provide EAP [ABV+04] framework for authentication. On the other hand, IPsec defines Security Policies according to various Traffic Selectors that includes subnetworks, IP addresses, ports, and upper layer protocols. Furthermore it secures the IP layer in the kernel, which does not impact the service, and thus makes possible an ISP to provide a Secured Offload for a third party service. IPsec has two modes: the Transport mode for End-to-End connectivity and the Tunnel mode to secure the link between the MN and a Security Gateway. At last, IPsec [ETS10] provides an EAP framework making authentication mechanisms [HS06, AH06] on RAN possible on WLAN.

6.3.3 MOBIKE-X: Mobility, Multihoming and Multi Interface Security Protocol for OSA and OAA

6.3.3.1 MOBIKE Does Not Fulfil MM Requirements

MM Security Requirements are partially handled by IPsec MOBIKE [Ero06] extension. MOBIKE has been designed for a MN with a single interface and the Tunnel mode. More specifically, Transport mode and Multiple Interfaces are not considered.

MN and the Security Gateway agree to use MOBIKE by exchanging a MOBIKE_SUPPORTED Notify Payload while establishing the IKE channel. If the MN and the Security Gateway support MOBIKE, when the MN changes its IP address, it sends the Security Gateway an UPDATE_SA_ADDRESSES Notify Payload. When receiving this Payload, the Security Gateway looks at the IP source of the Packet, and for all Security Associations (SA) associated to the MN, the Security Gateway changes the outer header IP address of the Tunnel. Note that only the outer header of the SA is a parameter of the SA, and does not affect the Security Policy. The Security Policy —*Tunneling traffic from my inner IP address* —is not changed. Thus, the Security Association Database is impacted; the Security Policy Database remained unchanged. Note that changing the outer header, results in tunneling traffic to the Security Gateway from IP_{OLD} and then from IP_{NEW}. This results in a Mobility operation that is transparent to encapsulated traffic. This MOBIKE Mobility Hard Handover is used in WLAN and takes advantage of the Tunnel mode.

For Multihoming, the MN informs the Security Gateway with ADDITIONAL_IP4/IP6_ADDRESS Notify Payload that an Alternate IP address may be used, if the MN is not reachable on the Primary IP address. If the MN happens to be unreachable, the Security Gateway performs a Return Routability Check to check the MN is still reachable on the Alternate IP address, and in case of success, it sends an UPDATE_SA_ADDRESSES to the MN so it updates its SAs.

In order to fulfil MM Requirements, MOBIKE-X must:

- Extend MOBIKE Multihoming and UPDATE_SA_ADDRESSES with the IPsec Transport mode
- Extend MOBIKE Mobility for Multiple Interfaces for both IPsec Transport and Tunnel modes. Typically this includes functionalities such as ADDing / REMOVing and UPDATING an Interface to an existing SA.

6.3.3.2 MOBIKE-X Makes MOBIKE Fulfil MM Requirements

MOBIKE-X [Dan09b] extends MOBIKE [Ero06] to address MM Requirements of section 6.3.3. MOBIKE-X [Dan09b] extends MOBIKE on at least two aspects. MM operations are extended to the Transport mode, and to Multiple Interfaces by using ADD_SA_ADDRESS / REMOVE_SA_ADDRESS Notify Payload.

Modifications in Transport mode are a bit more complex than with the Tunnel mode because in Transport mode, the IP address impacts both the SAD and the SPD. Then Mobility with Transport mode does not result in moving the communication as in Tunnel mode. Mobility with Transport mode updates the SAD and SPD, but other protocols like SHIM6 [NB09], SCTP [Ste07], mpTCP [FRH+11] have to move the communication from one interface to the other.

Then, Multiple Interfaces requires ADD, REMOVE and UPDATE operations to specify what needs to be modified. More specifically, MOBIKE, with a single interface, the UPDATE_SA_ADDRESSES does not carry any information: the new IP addresses are those in the IP header. With Multiple Interfaces, we use IP PARAMETER to specify the old and new IP addresses. When not explicitly provided, MOBIKE-X derives the PARAMETERS so to remain compatible with MOBIKE.

Finally MOBIKE-X offers the following advantages over MOBIKE: (1) MOBIKE-X remains compatible with MOBIKE Payloads, then (2) MOBIKE-X supports Transport mode and makes E2E possible. (3) with Multiple Interfaces, MOBIKE-X makes Soft Handover possible which reduces packet loss over Hard Handover. Furthermore, (4) it supports interface traffic management as the selectors can be renegotiated.

6.3.4 CPU consumption OSA vs OAA

6.3.4.1 Theoretical Estimation

With IPsec, the traffic can be secured with the Tunnel mode as in I-WLAN or with the Transport mode as in E2E. This section estimates the gains of CPU consumption provided by the use of Transport instead of Tunnel. In the early design of IPsec, [FS00] recommended to remove the Transport mode because it was considered as a subset of the Tunnel mode, and added complexity that balance neither the gain on throughput ($\approx 3\%$) nor the cryptographic overhead ($\approx 1\%$ [Hob10]. This was right as long as communications consider large Ethernet datagram of 1500 bytes, but for smaller IP datagram of $64-200$ bytes as those used for RTAs, the overhead of the Tunnel mode is not any more negligible. [Hob10] measures for Security Gateways the performance impact of Intel AES New Instruction (AES-NI) for the cryptographic AES-GCM on Linux. For 60 to 180 bytes RTA payloads, it estimates that removing the encryption of the 20 bytes of the inner IP header reduces the number of CPU cycles by 10% to 31% with AES-NI and by 6% to 26% with regular software AES implementation. Furthermore, with AES-NI, for a 200-byte packet (resp. 1500 bytes), the cryptographic computation consumes 16% (resp. 35%) of the total computation capacity whereas the remaining CPU cycles are left to the networking process. Thus, this recent performance measurement paper shows that using Transport mode significantly improves performances, CPU consumption of RTA. Similarly, [Gar08] evaluates IPsec performances on the 3G/LTE architectures by considering the tunnel between the eNobeB and the Radio Node Controller. For large packet size traffic (512 - 1420 bytes), IPsec tunnel overhead is shown negligible. However, for traffic with small payload (64 - 500 bytes), IPsec tunnel overhead reduces performances by $60-80\%$. [Gar08, MS10] measure the effect of IPsec VPN over the offloaded RTA traffic and measures that as soon as network are

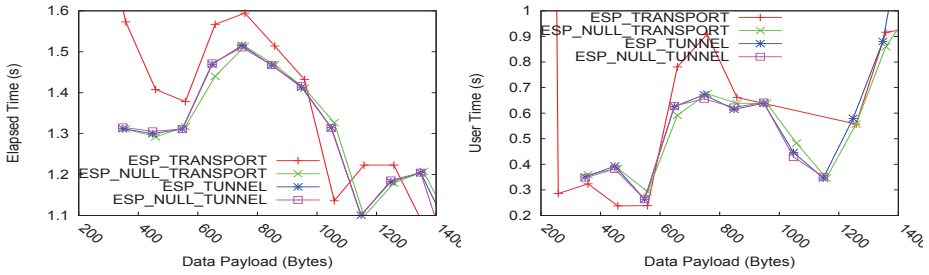
loaded or the Security Gateway is not highly available, the EU experience is impacted. This may be counter by prioritizing flows. However, prioritization reduces the Security Gateway impact, but ISPs have no impact the network congestion between the MN and our service. On the other hand Transport reduces network latencies and [Hob10, Gar08, MS10] conclude that Transport mode for RTA significantly reduces CPU consumption, and improves EU experience. Note also that with specific application IPsec links E2E eases flows prioritization in the ISP CORE Network.

6.3.4.2 Experimental Measurements

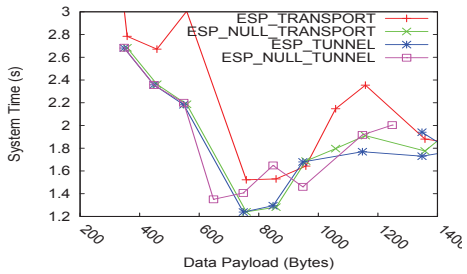
This section measures the CPU consumption of different IPsec mode and using different algorytms. Measurements do not take advanatge of the AES-NI for AES-GCM, as described in [Hob10]. For encryption, we use the default AES encryption (AES 128 CBC), and null otherwise. For integrity check we use the default SHA1 with 96 bits. A client download from the server a 1GB file with various IPsec configurations. The Server is a regular PC, and the client is a SAMSUNG NC10 with Intel(R) Atom(TM) CPU N270 1.60GHz. The reason we choose this client is that it is a low power consumption device, that are likely to be a MN. However, we do not generalize the experimental results for other architectures, and other tests must be performed. For all tests we checked the CPU is heavily loaded. For a MTU of 1000, we have for example, 46% with the NONE IPsec configuration, 60% for the ESP_TUNNEL IPsec configuration and 58% with the ESP_TRANSPORT mode. The Operating System are Ubuntu 11.10, and we use the system time command to evaluate the kernel time, the user time and the elapsed time.

What we are interested in is to evaluate the IPsec overhead on a communication, then, we want to isolate the networking part to the encryption part, and compare the Transport to the Tunnel mode. Figure presents the various measured time ratio with the NONE configuration, that is to say a regular non IPsec protect download.

Measurements are provided for various MTU, because we wanted to check the IPsec overhead on small datagram that are usually used on Real Time Applications. The MTU fixes the maximum size of the packet sent on the wire, which is different from the size of the datagram sent by the application. In fact the ESP [Ken05b] header adds the SPI (4 Bytes), the Sequence Number (4 Bytes), padding can be between 0 - 255 Bytes, the pad length (1 Byte) the Next Header (1 Byte), and the Integrity Check Value (12 Bytes in our case). The overhead is around 22 bytes for the Transport mode. With the Tunnel mode, the outer IP addresses are added which makes the IPsec overhead to 38 Bytes. Thus, in order to have consistent data with the NONE configuration, the data payload is derived from the MTU by subtracting 20 Bytes for the NONE, configuration, $20 + 22 = 42$ for the Transport mode and $32 + 20 = 52$ for the Tunnel mode.



(A) Elasped Time (B) User Time



(c) System Time

FIGURE 6.1: CPU consumption of IPsec protected links over non IPsec protected links

Figure 6.1 shows the IPsec overhead by measuring for an IPsec communication, the elapsed time, the user time and the kernel time. Elapsed time is the number of seconds it takes to download the file. User (resp. System) Time is the number of CPU seconds, the process spends in user mode (resp. in the kernel). For each IPsec protected communication, figure 6.1c presents the ratio with the non protected communication.

Elapsed Time represented in figure 6.2a is directly associated with the End User Experience. It shows a 30% overhead for Data Payload below 600 Bytes, a 50% overhead for Data Payload between 600 and 1100 Bytes, and a 20% overhead for Data Payload over 1100 Bytes. Because the overhead does not have a linear relation with the Data Payload, we can think that the overhead is highly impacted by the how the different processes are scheduled. Since the OS and the TCP stack has been optimized for a Ethernet MTU of 1500 Bytes, the IPsec overhead can be estimated between 10% and 20%. When scheduler are not working together, it looks that a process is waiting for the other, which leads to an overhead up to 60%.

Figure 6.1c shows the System Time, that is the time spent in the Kernel. The Kernel is used to forge and encrypt the IPsec packet [CB03]. Globally, the smaller the Data Packet is the more time the process requires system CPU time. One possible explanation is that small packet generates more interruptions for the kernel to perform some tasks. With IPsec, the process spends between 3 and 2 time more time in the kernel. For packets with Data Payload between 600 and 1100 the process spends almost no additional time in the Kernel for IPsec protected communications. Note that for Data Payload between 600 and 1100 Bytes, corresponds to the largest elapsed time in figure 6.2a, and relatively quite large time in the user land in figure 6.2b. Spending more time in the user land vs kernel means does not necessarily that takes are performed. More particularly, for synchronized tasks, one may regularly check the other is performed or not. Il looks that for this range of Data Payload, tasks in the user land are waiting for tasks performed in the kernel.

From figures 6.2a, 6.2b and 6.1c, we measured that encryption hardly influenced the IPsec overhead. This means that IPsec cost is mainly due to interruption. Similarly, Tunnel mode does not provide an additional overhead as expected.

6.3.5 OSA New Business Opportunities

Recent works investigated different behaviours for offloading traffic. [HHK+10, HHS10] consider social networking applications and offload over ad-hoc networks. [LRL+10] evaluates, based on live traffic, which download strategy saves battery. WLAN offers higher bandwidth than RAN

which reduces download time and saves battery. With a 1 hour timer before switching to RAN, the offloaded MN increases by 29% the traffic downloaded from WLAN. Because WLAN provides higher bandwidth, this reduces downloading time which results in reducing battery consumption by 20%.

With the *delayed* strategy, applications use the WLAN when available. When no WLAN is available, the application waits for a defined delay, and if no WLAN has been found, then the use the RAN. Using WLAN with greater bandwidth makes networking faster and results in saving battery. Simulations based on a one week traffic capture showed that already 65% of the traffic is already offloaded, with the *on-the-spot* strategy which saves 55% of the battery. With a $100\,ms$ delay, the gain is only about $2-3\%$, but raising the delay to 1 hour increases the offloaded data by 29% and saves the battery by 20%.

Similarly to application specific download strategies, our chapter optimizes the security according to the application security requirements and the network level of trust. More specifically, RAN is considered secured, and thus layer 2 security is enough. Switching on a WLAN may require layer 3 security depending on the level of trust of the network and application security requirements —i.e. what data are carried, is the communication secured at layer 4 by TLS for example. Our chapter considers three ways to secure a communication: (1) An optimized security channel using IPsec Transport mode, (2) a secure network access (OAA) and (3) no security at all. Since the network defines how security should be deployed, ISPs are good candidates for such services. Note that security also includes authentication of the MN. With offload, the ISP may also be able to authenticate a MN with RAN authentication method on behalf of some IP based services.

6.4 Related Works on IPsec based Architectures

Our chapter measures how the OSA secured communication performs during MM operations on to offloading traffic. We test communications with SCTP over IPsec, so we position our work toward IPsec, SCTP & IPsec, HIP and MIP.

6.4.1 IPsec Work

Several works [FS00, SGM07, Hob10, Gar08, MS10] analyse IPsec performances in several VPN configurations.

6.4.2 Interaction between SCTP / IPsec

Bellovin and al. [BIKS03] describes how IKEv1 establishes an IPsec SA with the multiple IP addresses of the SCTP association. MOBIKE(-X) is based on IKEv2 and [BIKS03] does not consider dynamic IP addresses management. Other works [CLW+08, HRUT06, CCC07, URJ04, LB08] evaluate different ways to secure SCTP communications and design TLS based protocol specific to SCTP: *Secure - SCTP* and *Secure Socket SCTP*. IPsec was rejected because of its 4 bytes overhead over TLS, leading to less than 3% throughput performance loss and a lack of flexibility for (1) different chunks (specific to SCTP) and (2) with Multihoming and Dynamic Address Configuration [SXT+07]. None of the previous work considers performance measurements for MM operations. Our work provides a generic solution MOBIKE-X, not SCTP specific and measures

performances over MM operations. Note that MOBIKE-X addresses MM IPsec limitations.

Noriega-Vivasand and al [NVCGRGL11] analyses a Home Node B (HNB) in a I-WLAN/3GPP architecture, with multiple WLAN/WIMAX/UMTS interfaces that use SCTP over MOBIKE so to select the best interface. This work differs from ours since (1) the architecture is WLAN and Tunnel based and (2) multihoming is never used for Soft Handover which is reported as a missing feature. Note that MOBIKE-X addresses that problem.

6.4.3 Alternative protocols: HIP - SHIM6 - MIP6

Other protocols than SCTP could have been selected. We give a special attention to HIP [MN06, MNJH08, Gur08] that provides both security with IPsec BEET mode [NM08] and MM facilities. HIP communications are established between crypto identifiers (Host Identity Tags or HIT). HIT are bound to an IP address. Since HITs remain fixed during the communication, IP addresses can be changed / added transparently to the application.

Actually HIP takes advantage of the Tunnel and Transport mode with the BEET mode. MM is transparent to the applications, and there is no tunnel header. More specifically, HIP splits the IP layer between Identifiers (HITs) and locator (IP addresses). As, described in [NGH10, GKLN08], this make possible to handle Mobility Multihoming and Multiple Interfaces transparently to the transport layer. Moreover this makes also possible to simultaneously handle both IPv4 and IPv6 addresses. Despite this new architecture [HG12] reports multiple HIP experimentations, and show that HIP deployment can be successful.

However, HIP suffers from two drawbacks: (1) Communications are always IPsec protected and (2) HIP breaks the current IP oriented communications. Protecting all communications adds an extra overhead on RAN for example, even though it could be reduced by using ESP_NULL.

HIP breaks the current IP-oriented communication model and the non-incremental characteristic imposes HIP to be deployed between the MN and the server on the RAN. Thus MOBIKE-X provides the IPsec characteristics of HIP to the IP oriented communications. On the other hand MOBIKE-X only considers the IPsec layer, and the MM features of the communication must be provided by other protocol. SHIM6 and SCTP are very good candidates. We choose SCTP because our platform is IPv4 only, as most of over infrastructure has not been yet deployed in IPv6.

6.4.4 Alternative Architectures

The OSA architecture differs from traditional Mobile IP based architectures [Per10, Per02] as the MN is managing the MM operation. This may add complexity at the terminal side, but experimentations have concluded that using MM aware terminal optimizes the MIP mobility operation [SFA04]. Thus we do not consider it is a major constraint. Then, MIP and MOBIKE-X do not address the same issue. MIP makes the MN reachable with its Home Address, whereas the OSA provides MM operation for a given communication. As a result simultaneous mobility of the nodes is not possible with OSA.

[HJH+10] is quite close to OSA architecture, as specific applications benefit from an HIP end-to-end communication with security and MM features. The remaining communications are tunneled to a Security Gateway located in the EU private network rather than in its ISP's core network.

As mentioned earlier MM can be performed at different layers [Rat04]. Here IPsec MM is handled at the IPsec layer, independently to other Mobility / Multihoming protocols. Therefore, we hope MOBIKE-X can be compatible with most of the Mobile / Multihomed architectures.

6.5 ISP FWDA, OSA and OAA combined Offloading Architecture

Mobility and Multihoming operations are closely linked to security in the case of offload because the EU interacts between networks with different levels of trust, and thus security requirements. Mobility and Multihoming Security Requirements are fulfilled if the EU is able to:

- MM_1 Move a communication from RAN to WLAN.
- MM_2 Move a communication between WLAN Access Points.
- MM_3 Move a communication from WLAN to RAN.
- MM_4 Provide Alternate IP addresses to recover from a connection fail over.

These Requirements are fulfilled for each type of traffic and each FWDA, OSA and OAA architectures. Furthermore for a given architecture Mobility may be handled by various protocols (SCTP, MOBIKE(-X), Application) in various ways (Soft Handover, Hard Handover). For each case, we measure how it impacts the EU experience so that ISP can choose the appropriate way to offload the EU traffic.

Section 6.5 presents FWDA, OSA and OAA. Then section 4.1.2 presents SCTP which provides Multihoming and Mobility features at the transport layer. Section 4.6 presents the different IPsec Multihoming and Mobility extensions: MOBIKE [Ero06] and MOBIKE-X [Dan09b]. Then we compare and position SCTP, MOBIKE and MOBIKE-X for Mobility and Multihoming handling.

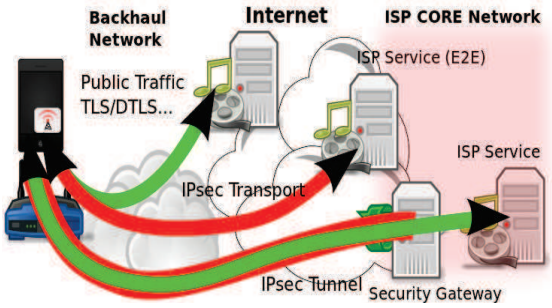


FIGURE 6.2: ISP Offload Infrastructure combining FWDA, OSA and OAA

FWDA: Public Traffic that does not require any protection, or traffic that is already protected at upper layers is directly forwarded by the WLAN Access Point to the Internet, and is not redirected on the ISP Network (cf figure 6.2).

OSA: Figure 6.1b presents the Offload Service Architecture. Unlike OOA where all the EU traffic is offloaded, in OSA, each service offloads its own traffic. Advantages of OSA are that 1) Offload is adapted to the Service, and avoids unnecessary encryption. For example, offloaded

traffic may be secured with IPsec or not depending on upper layer security such as TLS/DTLS (HTTPS) or whether the traffic needs or not to be secured. 2) Furthermore OSA provides End-to-End Security which reduces the load especially reduces load on ISP and backhaul network. In fact, by securing traffic from the EU to the Service with a direct communication avoids going through the ISP CORE and backhaul network. The EU may benefit from the WLAN Access Point it is attached to. Moreover OSA 3) reduces the ISP Security Architecture and improves the EU experience by avoiding Security Gateway. With End-to-End communications, ISP does not have to deploy Security Gateways, which introduced latency and routing indirection for EU traffic. At last 4) OSA reduces the Security Overhead by not only reducing the traffic that needs to be IPsec secured, but also by using Transport mode rather than the Tunnel mode. In fact, End-to-End communication makes possible the use of Transport mode, which reduces the number of bytes to be encrypted, as well as the networking overhead required by encapsulation.

OOA: Figure 6.1a presents I-WLAN [3GP11] as an example of Offload Access Architecture. The current model is that EU trusts its ISP so that - with the exception of legal interception - none is likely to intercept the EU traffic from its terminal to the Internet. The RAN Security architecture thus secures the Radio Access Link Layer from the EU Terminal to the eNodeB. Layer 2 security is sufficient since the eNodeB is a trusted entry point to the trusted ISP network. On the other hand, when the EU is offloaded, WLAN Acces point may not belong to the ISP, and thus may be not trusted. The EU is attached to the ISP Network via a Security Gateway, at layer 3. I-WLAN [3GP11] in figure 6.1a establishes an Secure IPsec VPN between the EU and the Security Gateway, thus providing a Secure Access to the EU. Then the EU uses the MOBIKE extension of IKEv2 [Ero06] to move from the tunnel from WLAN Access Points, and MIP to move from the RAN to WLAN.

In I-WLAN, the Security Gateway is called Tunnel Terminating Gateway (TTG). Communications with an ISP service hosted application are forwarded to the Gateway GPRS Support Node (GGSN), otherwise Internet communications are forwarded to the Packet Data Gateway (PDG).

I-WLAN is the standardized Offload architecture. In this chapter, we propose OAA which is similar to I-WLAN, except that we uses SCTP as the Mobility protocol to move from RAN to WLAN, and MOBIKE-X to extend MOBIKE features. A Mobility protocol other than MOBIKE is required because MOBIKE(-X) only works for IPsec protected communication. The advantage of using SCTP is that ISP don't have to deploy Home Agent required by a MIP infrastructure.

6.6 Deploying FWDA, OSA and OAA

The *ISP Offload Infrastructure* defines 3 different classes of traffic and associates each class to an offload architectures. FWDA is associated to traffic without any security requirements, OSA for traffic that provides End-to-End security, and OOA that tunnels all remaining traffic that needs to be protected (cf figure 6.2). When possible, Mobility is handled with MOBIKE(-X). However MOBIKE(-X) provides Mobility only for IPsec with Tunnel mode protected links, and so cannot deal with FWD and OSA. SCTP can deal with Mobility, and section 6.6.1 how FWDA, OSA and OAA can be deployed by combining SCTP and MOBIKE(-X). On the other hand, there may be multiple reasons an ISP does not want to deploy SCTP on its EU Terminal or on its Application. In that latter case, we detail in section 6.6.2, how ISP can still provide an infrastructure based on MOBIKE(-X), and/or application ability to deal with Mobility. The performance measurements of the two alternatives in section 6.7 will provide inputs for ISPs to decide whether SCTP worth

to be deployed or not. In both sections 6.6.1 and 6.6.2, we define for FWDA, OSA and OAA how the EU moves between RAN and WLAN as well as between WLAN Access Points.

6.6.1 Deploying FWDA, OSA and OAA with SCTP & MOBIKE(-X)

This section supposes the ISP has deployed SCTP and MOBIKE(-X), and details how FWDA, OSA and OAA can be deployed.

While connected on the RAN, SCTP Mobility and Multihoming features are not used. SCTP Multiple Interface ability is used to associate the IP address acquired on the RAN and the WLAN to a given connection. As represented in figure 4.1, the EU configures the new Interfaces (authentication, eventually IPsec IKE negotiation...) before sending an ASCONF ADD IP Payload. When the EU is connected through both IP_{RAN} and IP_{WLAN}, it defines its Primary Address to perform a Soft Handover. The EU may choose to REMOVE the old IP address, however, we recommend to keep IP_{RAN} when being offloaded and eventually REMOVEs IP_{WLAN} when it is not reachable any more. SCTP Multihoming feature is used when the EU is connected on both IP_{RAN}, and one or multiple IP^{i}_{WLAN}. When none of IP^{i}_{WLAN} are reachable anymore, then SCTP is used to switch on IP_{RAN}

The way Mobility is handled between various WLAN Access Points varies according to the Offload Architecture

FWDA: Forwards traffic on the Internet. SCTP is used to move the traffic from IP^{OLD}_{WLAN} to IP^{NEW}_{WLAN} with Soft Handover. SCTP Multihoming is also used to prevent breaking the communication if a WLAN Access Point fails. If the EU has only one WLAN Interface, to avoid breaking the SCTP connection, it may performs two Soft Handover from IP^{OLD}_{WLAN} to IP_{RAN}, then ADD IP^{NEW}_{WLAN} with an ASCONF before moving again from IP_{RAN} to IP^{NEW}_{WLAN}.

OSA: provides End-to-End IPsec Security using Transport mode. With the Transport mode, MOBIKE-X Mobility features must be used in conjunction of SCTP. More specifically, MOBIKE-X is not used to move the communication, but only to configure IPsec so that IP^{NEW}_{WLAN} is IPsec ready to protect the communication on IP^{NEW}_{WLAN}. Moving the communication from IP^{OLD}_{WLAN} to IP^{NEW}_{WLAN} is performed by SCTP Soft Handover. MOBIKE-X Multiple Interface feature is used to prepare the Soft Handover and avoid blocking the communication (cf figure 4.11b), and Multihoming is used in case the Primary Address does not work, and Alternate IP address is used. Multihoming works for the IKEv2 application, otherwise, with TRANSPORT, SCTP Multihoming must be used to move the communication from Primary to the Alternate Address.

OAA: The communication is Tunneled to the Security Gateway, and MOBIKE is used to move within the WLAN with Hard Handover. MOBIKE-X makes Soft Handover possible as presented in figure 4.11b, which improves the EU experience, by reducing the number of lost packets. MOBIKE(-X) Multihoming works as with OSA, but with the Tunnel mode, both IKEv2 and the communication are moved to the Alternate Address. MOBIKE(-X) Multihoming is only used with the WLAN, the Alternate IP address is on the RAN, then SCTP Multihoming is used to switch from WLAN to RAN.

6.6.2 Deploying FWDA, OSA and OAA with only MOBIKE(-X)

This section defines how FWDA, OSA and OAA can be deployed without SCTP.

In section 6.6.1 SCTP is used to switch from RAN to WLAN. Without SCTP, RAN to WLAN and WLAN to RAN Mobility relies on other mechanism. Most applications like FTP/HTTP, have session resumption mechanisms to avoids re-downloading a whole file when a connection is restarted with *first-byte-pos* options. Similarly TLS provides also Session Resumption [RRDO10, KES06, SCAC09] mechanisms. The main difference with SCTP is that application performs Hard Handover, whereas SCTP performs Soft Handover. Hard Handover results in longer interruption and more packets lost. Then SCTP provides the Mobility and Multihoming framework for all applications whereas without SCTP each application has to deal with its own session resumption mechanism. However, most EU applications have been designed to be robust to network failure, and interrupted connectivity: HTTP, HTTPS, FTP have session resumption mechanisms, Peer-to-peer Files download don't block when a peer is not reachable, downloading time during Web Browsing is very short, Video Streaming buffers up to a few seconds of videos. As a result, only few real time applications like games, chat, VoIP may be impacted by a small interruption.

FWDA: Mobility and Multihoming is completely handled by applications. If an application cannot handle them, then it should be offloaded with OAA or OSA with Tunnel mode.

OSA: With Transport mode, Mobility must be handled by the application. Thus, moving from RAN to WLAN is performed as in figure 6.3a, except that Transport mode is used instead of Tunnel. However, if the application cannot provide Mobility features, the ISP may use OSA with Tunnel mode. This adds a Security overhead, but still provide End-to-End connectivity as well as Mobility and Multihoming features on WLAN. In that case it is recommended that IP_{RAN} may be globally routable, as detailed for OAA.

OAA: OAA tunnels traffic to a Security Gateway. There are three types of traffic: 1) traffic addressed to mobility aware application that requires to be encrypted 2) traffic addressed to applications that are not mobility aware application that requires to be encrypted and 3) traffic of FWDA of applications that are not mobility aware. Figure 6.2 presents the RAN to WLAN Mobility with OAA, indicating between horizontal lines and with a yellow background the time the communication is stalled. With 1) application moves from RAN to WLAN and WLAN to RAN, and exchanges are illustrated in figure 6.3a. On WLAN Mobility is handled with MOBIKE(-X) as in section 6.6.1 (cf figure 4.11a and 4.11b).

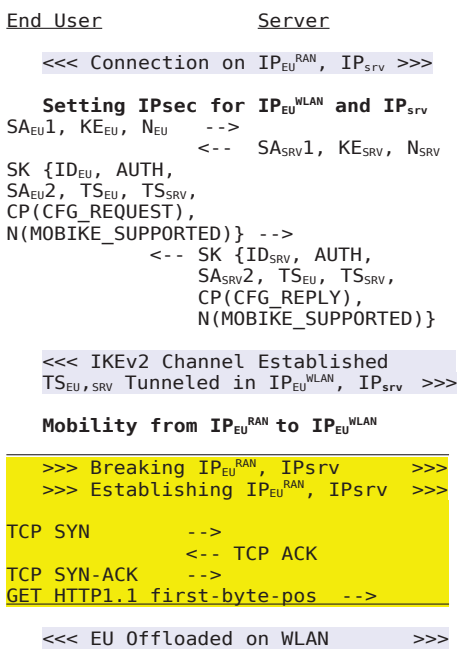
For 2) and 3) MOBIKE-X must be used to move the communication between RAN and WLAN. With 3), the Tunnel may not be encrypted, IPsec is only used to provide Mobility and Multihoming. To move from RAN to WLAN, the main idea is to establish an IPsec Tunnel and then use MOBIKE-X to move to the WLAN. The EU modifies its SPD from BYPASS to PROTECT for the given traffic, and initiates an IKE negotiation. The Security Gateway may be configured with a light authentication, since the EU has been authenticated by the RAN. Then the Security Gateway must configure the ISP CORE network to become a border router for the IP_{RAN}. In fact, the EU encapsulates between IP_{RAN} and $IP_{SECURITY\,GATEWAY}$ the traffic from IP_{RAN} to IP_{SERVER}. IP_{RAN} and IP_{SERVER} are designated as Traffic Selectors (TS). The Security Gateway decapsulates the traffic and sends it to IP_{SERVER} on the Internet. In return, when the Server responds to IP_{RAN}, the IP datagram must be routed to the Security Gateway so that it can encapsulates it back to the EU. Once the IKEv2 negotiation is finished and the SA configured, the EU can use MOBIKE(-X) Multihoming / Mobility mechanisms to move on WLAN. On

WLAN, Traffic Selectors are not modified, only the Tunnel outer IP addresses are modified. This scenario is presented in figure 6.2b. Note that IP_{RAN} must be routable and that the connection is stalled during the whole IKEv2 negotiation. On the other hand MOBIKE can be used for that scenario, even though MOBIKE-X may be preferred for Multiple Interface and Soft Handover.

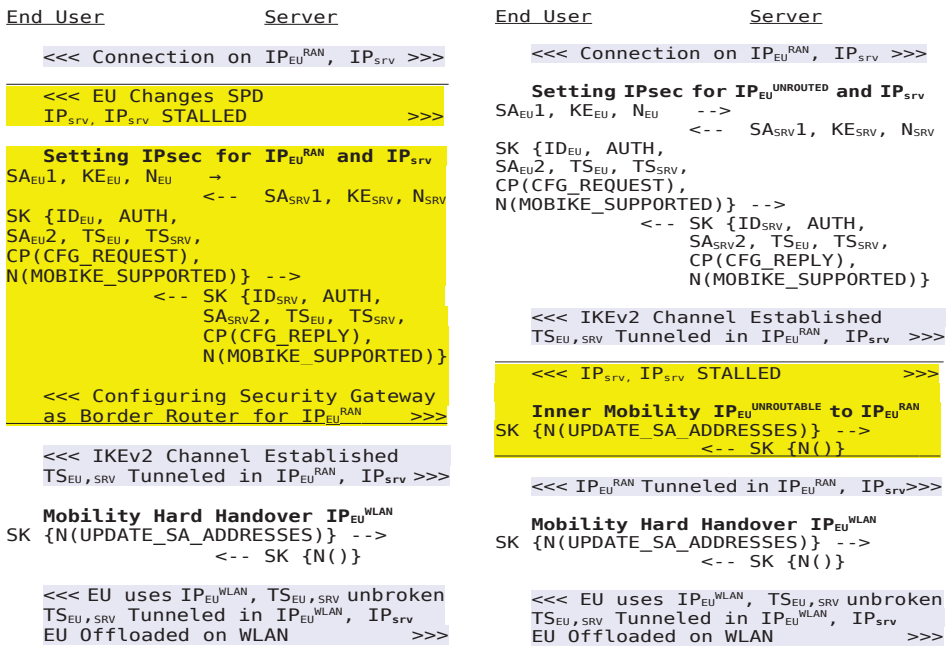
Figure 6.2c shows how to take advantage of MOBIKE-X to avoid stalling the communication during the IKEv2 negotiation. MOBIKE-X does not require any Routing configuration in the ISP Network, nor IP_{RAN} to be globally routable. However, using non globally routable IP_{RAN} results in breaking the connection and rely on application Mobility or recovery mechanisms. The main idea is to use MOBIKE-X to negotiate the tunnel with an non routable IP address as the inner IP address, than, to change the inner IP addresses —TS. Once the tunnel is configured with the proper inner IP addresses, the EU proceeds to the regular MOBIKE(-X) Mobility by changing the outer IP addresses.

The stalled time will be in the order of a $3\,RTT$, one RTT to change the Traffic Selectors on the Security Gateway and on the EU, and two RTT to restablish the connection between the EU the Server. Note that Mobility between RAN and WLAN is always Hard Handover.

WLAN to RAN Mobility is performed using MOBIKE(-X) as if RAN required to be secured. The EU can either re-negotiate the SA, and remove the encryption in case of another offload. The EU can also choose to DELETE the SA and provide direct connectivity between the inner IP address and IP_{SERVER}. MOBIKE-X Inner mobility may be required to change the IP_{WLAN} inner IP address to IP_{RAN}.



(A) Application Mobility



(b) MOBIKE (c) MOBIKE-X

FIGURE 6.2: Messages Exchanges for OAA RAN to WLAN Mobility

6.7 Performance Measurements

Section 6.6 describes various ways to deploy FWDA, OSA and OAA by considering either SCTP traffic or non SCTP traffic. On the WLAN, when the connection is IPsec protected with the Tunnel mode, OAA uses the standard MOBIKE(-X) for Mobility. OSA with Transport mode combines MOBIKE(-X) with SCTP, or with application Mobility or Session Resumption mechanisms. The main issue comes with connections that are not IPsec protected. In that case, standard MOBIKE(-X) is not supposed to be used. Section 6.6.1 describes how to deploy FWDA, OSA and OOA with SCTP and section 6.6.2 describes how FWDA, OSA and OAA can be deployed without SCTP, with only MOBIKE(-X).

This section presents the platform we use to measure how the various Mobility mechanisms affect the connectivity. Then, given how the EU experience is affected by the various WLAN Mobility, we provide recommendations for deploying progressively the *ISP Offload infrastructure* in section 6.7.3. This is followed in section 6.7.4 by recommendations on the protocols to use to move between RAN and WLAN, and whether Mobility should be performed by the application or by SCTP.

6.7.1 Experimental Platform

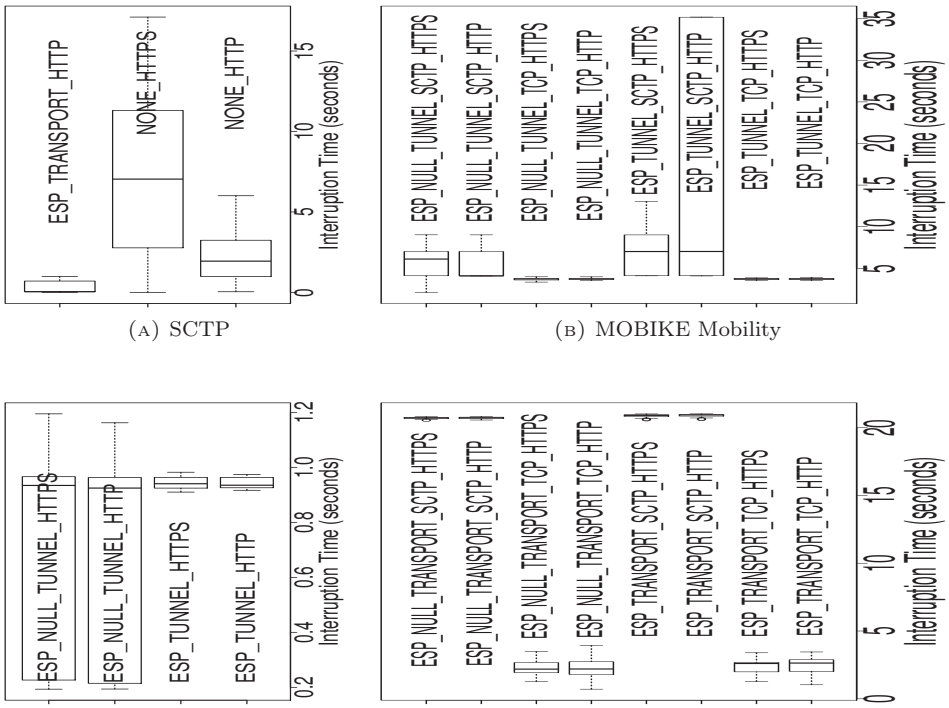
Our experimental platform is composed of a EU with Multiple Interface running on *Fedora 17 Linux OS 2.6.38-rc7*. SCTP implementation is *LKSCTP-2.6.28-1.0.10* [LKS] patched with *fastmsctp-2.6.34-rc5.patch* to enable ASCONF. IKEv2 implementation uses *strongSwan 4.3* [str] and implemented MOBIKE-X [Dan09b] on this version. All tests are performed with HTTP(S) traffic over Ethernet.

Measurements are performed on Ethernet because, TCP/ SCTP have been configured for this

Network. We provide realistic estimation for EU connected on WLAN, and RAN by measuring FTP download on various Networks over a day. Public ISP WLAN (*Public HotSpot*, $RTT = 15\,ms$, $Data\ BR = 1021.30\,kbs^{-1}$) Home WLAN with 1Mbits ($HWLAN_{1Mb}$, $RTT = 2.14\,ms$, $Data\ BR = 10199.42\,kbs^{-1}$) and Home WLAN with 10 Mbits ($HWLAN_{10Mb}$, $RTT = 9.466\,ms$, $Data\ BR = 2131.87\,kbs^{-1}$). On our $LabPlatform_{Ethernet}$ $RTT = 0.355\,ms$, $Data\ BR = 34715.04\,kbs^{-1}$.

Measurements show statistical results, and present median as well as quartiles.

6.7.2 Experimental Mobility Measurements



(A) SCTP (B) MOBIKE Mobility

(C) MOBIKE Multihoming (D) MOBIKE-X Mobility

FIGURE 6.2: Experimental Measurements for WLAN Mobility with different protocols (SCTP, MOBIKE, MOBIKE-X) and different configurations (Mobility, Multihoming)

FWD: is associated to HTTPS traffic as well as HTTP traffic with no security requirements. SCTP Mobility is performed with a Multihomed EU by putting down the Primary Interface. The EU discovers the Primary Interface is down, and switches to the Alternate Interface. Figure 6.3a sums up the results, and shows that SCTP interrupts the HTTP communication for $1.94\,s$. SCTP Multihoming does not require extra messages, thus this time is more or less the same on *Public HotSpot*. Figure 6.3a also shows that TLS interacts with SCTP, which results in an $7.01\,s$ interruption. Finally, Mobility handled with SCTP impacts the EU experience for Real Time Application. Then LKSCTP and TLS interactions shows that porting application to SCTP is not straight forward and that resources must be dedicated for that task.

OSA: Figure 6.2d shows that Mobility with MOBIKE-X and Transport mode is independent from TLS or the use of non encrypted IPsec. On the other hand SCTP and IPsec results in a 20 s interruption compared to a 2.57 s with TCP. With the Transport HTTP and HTTPS re-initiates the TCP connection. Comparing the HTTP/HTTPS connection re-establishment to SCTP Mobility shows a difference of $2.57 - 1.94 = 0.63\,s$ in favor of SCTP.

OAA: Figures 6.3b (resp. 6.2c) measures MOBIKE Mobility (resp. Multihoming) impact on the communication. Mobility is performed by changing the IP address the running Interface, as if the EU had a single Interface. Multihoming requires Interface down detection before switching to the Alternate Address. Figures 6.3b and 6.2c show that Mobility is neither impacted by the use of TLS nor the use of null encryption Tunnel. However, SCTP interrupts the communication around 7.02 s compared to 3.72 s with TCP. In fact SCTP and Tunnel interact with the Kernel routing policies. Figure 6.2c shows that Multihoming performs better than Mobility, probably because Multihoming orders Kernel events to prepare the Mobility.

6.7.3 Recommendation for Offload Deployment

This section compares the EU experience on WLAN Mobility over FWDA, OSA or OAA, and provides deployment strategies for the *ISP Offload Infrastructure*.

FWDA: From figure 6.2d and figure 6.3a SCTP Mobility takes $\approx 1.94\,s$ for non TLS traffic. For TLS traffic, session resumption would add 3 RTT, and with a measured $RTT = 0.015\,s$ on *Public HotSpot*, SCTP Mobility for TLS is $\approx 2\,s$.

OSA: MOBIKE-X and Transport Mobility takes $\approx 2.7\,s$. For MOBIKE(-X) one can eventually add 1 or 2 RTT depending whether Return Routability Check is performed or not. This adds a 40% increase over FWDA, but MOBIKE-X Multiple Interface ability may reduce even further this delay.

OAA: MOBIKE(-X) and Tunnel Mobility takes $\approx 3.8\,s$, adding 100% over FWDA which affects the EU experience. This considers neither routing indirection nor latency introduced by the Security Gateway.

Given how the EU is affected by the various Offload Architecture, we recommend ISPs to firstly deploy FWDA for all TLS and traffic with no protection requirements. Secondly to deploy OSA for service with Real Time requirements, and OAA for the remaining traffic. Note those recommendations match the recommendation to optimize costs deployment of the architecture. In the second phase, ISP may improve the EU experience by taking advantage of MOBIKE-X Multiple Interface and deploy Soft Handover. In fact measurements indicate that interruption mostly results from OS Network Layers configuration rather than Network latencies.

6.7.4 Recommendation on Mobility between RAN and WLAN

This section compares the various ways to move between RAN and WLAN. Then it provides recommendations on whether Mobility may use SCTP, MOBIKE or MOBIKE(-X) only architecture.

FWDA: SCTP interrupts the communication around 2 s for both non-TLS and TLS traffic.

OSA: MOBIKE-X and Transport mode interrupts the communication for $1\,s$ with SCTP and 2.74 when the application handles the Mobility.

OAA: SCTP presents similar performances in OAA and FWDA. With MOBIKE we estimate IKE negotiation $T_{IKE} \approx 60\,ms$ with the PSK authentication. PSK authentication is very fast and including T_{IKE} does not affect MOBIKE Mobility which is $\approx 3.8\,s$. To avoid T_{IKE}, MOBIKE-X must modify inner IP address first. This does not affect the Network Layers of the OS as the outer IP addresses are not modified. In fact only IPsec databases are updated, which should add a $1-2\,s$ delay, leading to a $5.5\,s$ interruption. MOBIKE, even though it includes an IKE negotiation, provides faster Mobility.

Thus, the first recommendation is to keep the architecture as simple as possible by using dedicated protocols or limiting Network Layer modifications. If SCTP is available then using SCTP is recommended, otherwise MOBIKE with fast authentication is recommended (like PSK). If there are specific needs for the application, or the authentication cannot use PSK, then MOBIKE-X should be used. In a second phase, ISPs may take advantage of MOBIKE-X Soft Handover, and inner mobility.

6.7.5 Recommendations on Mobility with SCTP vs Application

This section discusses whether Mobility should be handled by SCTP or by the application itself.

SCTP Mobility is around $2\,s$ for HTTP connection, and around $2.73\,s$ for HTTP with MOBIKE-X TRANSPORT. SCTP provides a 30% advantage over application. For example wget can be configured for HTTP, HTTPS and FTP with the *–read-timeout* option that defines, during a download, the idle after which the TCP connection breaks, *–tries* specify the number of tries before reporting a failed download. *–wait* defines the seconds between retires and *–continue* avoids restarting the download from begining. Every timers are configured in second, with MOBIKE-X, Mobility takes $2.73\,s$, but we may expect slightly better performances with tunned client. However, SCTP advantage must be balanced with the fact that HTTP(S)/FTP clients are very easily configured, and that slight modifications can generate long delays with SCTP ($7\,s$ in TLS, $20\,s$ with TRANSPORT, routing interaction with Tunnel).

Thus recommendations are to configured Mobility at the application layer for most ISPs applications, and consider SCTP when the EU experience is impacted by Mobility.

6.7.6 RAN to WLAN Mobility

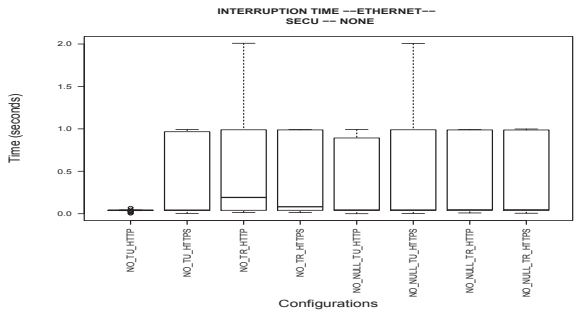


FIGURE 6.3: Experimental Measurements for RAN to WLAN Mobility

Figure 6.3 shows Interruption Time when moving from non secured communication on a RAN to a WLAN where the communication is secured with IPsec. Although links are different, the time the communication is interrupted is around 2 times smaller than in WLAN mobility with SCTP.

6.8 Conclusion

This chapter describes the *ISP Offload Infrastructure*. Contrary to I-WLAN [3GP11] where ALL EU traffic is tunnelled to a Security Gateway, the *ISP Offload Infrastructure* defines 3 types of traffic which are associated to a specific and dedicated Offload Architecture. Traffic that does not require any security, or that is already protected by other layers like TLS is associated to FWDA and is forwarded directly on the Internet. Traffic for specific ISP hosted services is associated to OSA that provides End-to-End security with Transport mode. The remaining traffic is tunnelled to the Security Gateway with IPsec Tunnel mode.

By providing an adapted Offload Architecture to each type of traffic, we expect to reduce drastically the infrastructure ISPs have to deploy, as well as to improve the EU experience. OSA and OAA are IPsec based architectures and so require the ISP to deploy an IPsec infrastructure. The IPsec infrastructure can handle Mobility with MOBIKE(-X), but other Mobility protocols are required when the communication is not protected with the Tunnel mode. This chapter considers deploying OSA, OAA and FWDA by either using SCTP or by relying on the application session resumption mechanisms.

For all possible scenarios we measured how Mobility interrupts the communication and may affect this EU experience, and balance it with the cost of porting application to SCTP.

We find out that ISPs may deploy in a first phase FWDA. Then ISP should deploy OAA and OSA. To ease OSA deployment, ISPs may start by deploying OSA with the Tunnel mode which makes MOBIKE(-X) deal with Mobility on the WLAN. To move from RAN to WLAN, MOBIKE may also be used at first. This would correspond to a first deployment version of the *ISP Offload Infrastructure*. For version 2, we recommend ISPs decide to port applications to SCTP or to configure properly the session resumption mechanisms so that OSA can migrate from Tunnel to Transport mode and do not rely on MOBIKE for Mobility between RAN and WLAN. For version 3, we recommend to optimize MOBIKE(-X) so perform Soft Handover. For version 4, we recommend ISP to focus on the RAN to WLAN optimization with MOBIKE-X for application that are not ported to SCTP.

Future work includes interactions between IPsec and SCTP, especially for the OSA architecture, SCTP and MOBIKE-(X) performs Mobility simultaneously for IPsec and SCTP. We believe SCTP may take advantage of IPsec signaling, using a cross layer communication. This would require the definition of an IPsec API.

Conclusion

Designing a Resolving Platform for DNSSEC

This thesis is focused on enhancing security for End Users in a Mobile, Multihomed, and Multiple Interfaces environment.

The first part of this thesis provides a Resolving Platform architecture. This enables ISP to provide their End Users a Secure Naming Resolution Service. We started our work with experimental measurements to evaluate the cost of migrating a DNS resolving platform to DNSSEC. By measuring the CPU consumptions of different implementations we show that DNSSEC requires, depending on the implementations, between 169% and 500% more resources for the Resolving platforms and 130% more resources for Authoritative Platforms. ISPs can hardly afford moving from a 100 node Resolution Platform to a 500 node Resolution Platform. Such an increase of the number of nodes represents too much Operation and Management (OAM), power supply or space issues. Then, we design a platform that optimizes the Resolving Platform for DNSSEC.

In order to optimize the resources of the platform, we attempt to limit DNS(SEC) operations that require a lot of CPU. We identify two major operations : DNS cache lookups and DNSSEC Resolutions. Current DNS Resolution platforms are composed of a load balancer that splits the traffic between the nodes by XORing the IP addresses of the DNS query. Each node of the platform resolves the received DNS query. Such load balancers distribute uniformly the load between the nodes, but the same FQDN may be sent on different nodes, thus triggering parallel resolutions.

At first, we estimate how splitting the DNS(SEC) traffic between the nodes can significantly reduce the load of the platform. We used a load balancer that splits the DNS(SEC) traffic by hashing the FQDN and showed that globally the DNSSEC Resolution Platform requires 30% fewer nodes, than with the current architecture. On the other hand, using such a load balancer, results in a very non uniform distribution of the resources among the nodes of the platform. More specifically, some nodes receive more queries or perform more resolutions than the others. As a result FQDN load balancing reduces the necessary resources, but the remaining challenge is to make the resources uniformly distributed among the nodes.

Resulting in a uniformly distribution of the resources can be done in two ways: one way is to define specific rules for the load balancer, that result in uniformly distributing the resources between the nodes. With this solution, all the intelligence is placed in the load balancer, and the nodes keep on responding to the queries that are sent to them. This alternative has been developed in [MHS+a], but not in this thesis. In this thesis, we adopted the opposite view, that is to say, leaving the load balancer unchanged, and placing the intelligence on the nodes of the Resolving platform. With this architecture, the nodes are cooperating. We based the cooperation

on Distributed Hash Table (DHT) mechanisms like Pastry. Our motivation for this architecture is mainly that load balancers are sensible equipments under heavy load which makes operational teams reluctant to modify the load balancers. Furthermore, modifying the nodes provides much flexibility for innovation, and we could benefit from multiple mechanisms that have been already elaborated. We evaluated various Pastry architectures for our platform and showed that pro-active cache sharing architecture reduces the number of nodes by more than 3.5.

The pro-active mechanism takes advantage of the Zipf distribution of the FQDNs' popularity. More specifically, the 2000 most popular FQDNs correspond to 70% of the traffic. Thus, populating the cache of the nodes with the most popular FQDNs, results in increasing the Cache Hit Rate, and in distributing the load of the DNS queries between the nodes. In fact, for these queries cached, the DNS query is handled by the receiving node and not necessarily its Responsible Node. This results in uniformly distribute the load associated these cached most popular FQDNS, while still avoiding parallel resolutions.

However, with the proactive mechanism, the DHT node considers a common cache for the FQDNs that are pro-actively cached and the FQDNs the node is responsible for. The number of FQDNs the DHT node is responsible for is quite large, thus increasing the resources for a cache lookup. A first optimization would be to introduce some levels of caches. A first cache would only contain the most popular FQDNs —that is to say the pro-actively cached FQDNs —, a second cache would contain the FQDNs of the DHT. However, this adds complexity to the DHT process. In order to keep the DHT process simple, and to take advantage of hardware acceleration, we looked how to extract the proactive mechanism from the DHT process. This makes possible to provide a light process that can be placed in the front end of the platform, dealing with 70% of the traffic. This leads to the $PREFETCH_X$ architecture. We show with simulations that the performance could at least be improved by 4 over the traditional DNS Resolution Platform architecture. Furthermore, this architecture is expected to improve much more the performances by taking advantage of the hardware acceleration. As a result, it is the most promising architecture.

MOBIKE-X: Seamless IPsec Security in a Mobile, Multihomed and Multiple Interfaces environment

The second part of this thesis is focused on providing mechanisms so End Users can keep their IPsec protected communication in a Mobile, Multihomed and Multiple Interface environment.

At first, we identified how Mobility Multihoming and Multiple Interfaces impact IPsec configuration. Then, we positioned MOBIKE, the currently defined protocol that deals with IPsec Mobility and Multihoming. MOBIKE considers mobile nodes with a single interface, and only considers the IPsec Tunnel mode. We designed MOBIKE-X to overcome these restrictions. Then we implemented and measured the performances of MOBIKE-X in a lab environment. Because IPsec mobility and the Transport mode require a protocol that handles the mobility of the communication in combination with IPsec mobility, we used SCTP.

Tests with SCTP show that SCTP Mobility over IPsec protected links take 666% more delay. Then comparison between using Transport and Tunnel mode shows that securing Real Time applications with IPsec would favor the Transport mode over the Tunnel mode. Most probably, strong Real Time requirements will result in a dedicated IPsec Transport architecture, mainly to avoid traffic congestion and avoid traffic indirection. Transport mode reduces CPU load by 20%, enhances Mobility and Multihoming operations by about 15%, and makes the system 2.9 times more reactive for detecting modifications of interfaces. Transport mode optimizes Mobility Multihoming and Multiple Interfaces compared to the Tunnel mode. However, to reduce drastically Mobility

Multihoming and Multiple Interfaces, the MN may anticipate its operations and prefer Soft Handover to Hard Handover as performed by the current MOBIKE. In fact, Multihoming relies on system interface detection and requires further network verifications such as Return Routability Check which stalls the communication around 5.7% longer than Mobility. As a result, the Network Manager is encouraged to perform Mobility operations rather than relying on failover mechanisms like Multihoming.

To face the huge demand on mobile traffic, ISPs are looking to offload traffic of their Radio Access Network to WLAN. Currently I-WLAN is the proposed offload architecture by 3GPP which tunnels the traffic to a Security Gateway.

Thus, we propose for ISPs an *ISP Offload Infrastructure* which minimizes the infrastructure cost deployment, and which can be deployed in a very short term. The *ISP Offload Infrastructure* classifies the EU traffic into 3 distinct classes and assigns each class a specific and adapted offload architecture: *FoWarD Architecture* (FWDA), *Offload Service Architecture* (OSA) and *Offload Access Architecture* (OAA). This thesis shows how to deploy each Offload Architecture by using SCTP in conjunction to MOBIKE(-X) or only MOBIKE(-X). Then we measure how each Offload Architecture may affect the EU experience, and provide recommendation on how to deploy and implement the *ISP Offload Infrastructure*

Perspectives

This thesis focused its effort on two areas, provides significant results, but a lot of work remain to make these effort deployed on operational networks. Here are some identified topic that worth being investigated:

Hash Functions for FQDNs: In this thesis we did not pay much attention to the hash functions that are either used to split the DNS traffic between the nodes of the platform, or to associate a Responsible Node to a FQDN. In fact we mostly considered cryptographic hash functions, because they provide a good avalanche effect. However these hash functions may not be very efficient for our purpose, and may require too much resources. We already pointed out that using XOR or SHA1 on the IP addresses of the DNS query results in similar distribution of the load among the nodes of the platform. The advantage of XOR is that is requires less resource than SHA1. As such, we believe work should be done to measure, compare and define a proper hash function for the DNS traffic. We are looking to a hash function that is very fast and uniformly spread the FQDNs between the nodes. Here we are not interested in the number of queries and resolutions associated to each FQDNs.

Light Pastry for Small Managed Networks: Similarly, we re-used the original version of Pastry, but functions, like routing protocol, are not involved on our platform. We encountered some performance issues with our Pastry implementation, and believe that a Light Pastry protocol may be designed for the Small Managed Network. From this thesis, we have identified the two following points: (1) removing unnecessary functions —like routing —, and (2) improve failover recovery mechanisms. However, re-designing a Light Pastry protocol would require to also reconsider the Operations and Management's requirements.

Light prefetching daemons Similarly to an adapted Pastry protocol, some work also needs to be done to finalize the prefetching mechanisms. Implementation work should be done to optimize the prefetching processes that may run on a Network Hardware Acceleration Card. Optimization should consider the card architecture. If one supposes the card has multiple cores, one may consider that different processes running on different cores. The idea is to design the process so that one

does not block the others. Among all processes we have the "Cache lookup" process that performs a cache lookup. In case of a cache hit it sends the response, otherwise, it forwards the query to the DHT process. One should for example, design the cache so to enhance the cache lookup. Then we have the "Resolving process", which does not have much constraints. In fact, resolutions are performed in advance, and we would strongly recommend to re-use existing libraries developed for BIND or NSD. Another process is the "Filling Cache" process that fills every nodes's cache with the updated responses. The process may be optimized in various ways. First cache updates should be optimized because during the update the cache is locked. One may check the best time to trigger the update, what data to send —incremental versus all data —, the most efficient data to send —byte code, vs objects, vs abstract description —. Then, multiple processes may be added, so to check the validity of the prefetched FQDN list, to check the reachability of the nodes. The issues raised by this item are both design issues that may require simulations and implementation issues. With an optimized Light prefetching daemon, one should experiment either with real time measurements the number of FQDNs that can reasonably be prefetched.

Interaction between MOBIKE-X and MPTCP MOBIKE-X has only been implemented in a beta version, so at first, a finalized implementation should be made available. Then tests we performed with SCTP should be re-done with MPTCP. We use SCTP in this thesis because MPTCP was in its early development at the time we performed the tests. Now MPTCP developments are more advanced. MPTCP has been designed and developed to match our use cases. Furthermore, it has been designed to enable MPTCP connections to go across middle boxes, which makes MPTCP compared to SCTP, more likely to be deployed. SCTP in fact has originally been designed to transmit signalization, and despite efforts has not been adopted by applications. Instead it is widely deployed in backhaul Networks which slows down the effort on Mobility. Results are not expected to differ much from MPTCP and SCTP. However, during the tests, we noticed that kernel implementations provide unexpected interactions. This is the main motivation to make MPTCP and IPsec work together. In other word, this thesis has provided by combining IPsec and SCTP, an IPsec proof-of-concept, and combining IPsec and MPTCP would provide an End User usability proof-of-concept.

Interaction between IKEv2 and Upper layer applications With MOBIKE-X, IPsec is able to send information to the peer like adding an interface, removing an interface, performing mobility with soft / hard Handover or peer unreadability detection. Upper Layer protocols like SCTP or MPTCP or Connection Manager have their own protocols to provide similar signaling. Effort should also be made to make upper layer protocols interact with the IPsec layer. More specifically, IPsec is the lowest layer, and may be used to advertise Interface event to local daemons. In return these local daemon may use IPsec to provide this information to the peer in a secure way. This requires to design and develop an IKEv2 API.

Adaptative Security With an IKEv2 API, an application is likely to interact with IKEv2 and IPsec. However, one needs to know when to interact and how. One domain of research would be to define which security rules should be set in a given environment. The security rule may depend on the level of security required by the communication, this may be combined with the level of trust of the network, the type of used terminal or the way the end user has been authenticated. According to these inputs, the connection manager may define which interfaces should be used, as well as the security policies to set up.

Part III

ANNEX

Appendix A

DNS(SEC) Measurements Complements

A.1 Mathematical Expressions of Experimental Measurements

This section provides for the different configurations and the different implementations numerical expressions of the curves provided by section 1.5 *Experimental Work*. Equations provided in this section are based on experimental results only. Their purpose is only to provide graphical resolution since equations are easier to handle than graphs.

A.1.1 Expression of Maximum Load

This section provides CPU Load value as a function of the Query Rate per seconds. Equations are derived from figures 1.4 in Section 1.5.2.

$$cpu_{DNS}^{BIND,\,auth}(q) = 0.012147 \times q + 3.638841$$

$$cpu_{DNSSEC}^{BIND,\,auth}(q) = 0.015833 \times q + 3.842773$$

$$cpu_{DNS}^{BIND,\,resol}(q) = 0.090918 \times q + 1.402989$$

$$cpu_{DNSSEC}^{BIND,\,resol}(q) = 0.102750 \times q + 0.187266$$

$$cpu_{DNSSEC-validation}^{BIND,\,resol}(q) = 0.192542 \times q - 1.897584$$

$$cpu_{DNS}^{NSD,\,auth}(q) = 0.005263 \times q - 2.293101$$

$$cpu_{DNSSEC}^{NSD,\,auth}(q) = 0.006093 \times q - 0.083391$$

$$cpu_{DNS}^{NSD,\,auth}(q) = 0.025499 \times q - 0.899292$$

$$cpu_{DNSSEC}^{UNBOUND,\,resol}(q) = 0.031408 \times q - 1.354852$$

$$cpu_{DNSSEC-validation}^{UNBOUND,\,resol}(q) = 0.101004 \times q + 0.554175$$

A.1.2 Network Latency & Response Time

This section provides the Server Response Time as a function of the CPU Load. Equations are derived from figure 1.5 in Section 1.5.3.

$$t_{DNS}^{BIND,\,auth}(cpu) = 81.910 \times 10^{-6} e^{3.3287 \times cpu^{3.2711}} + 155.99 \times 10^{-6}$$

$$t_{DNSSEC}^{BIND,\,auth}(cpu) = 29.004 \times 10^{-6} e^{3.8496 \times cpu^{1.7598}} + 187.11 \times 10^{-6}$$

$$t_{DNS}^{BIND,\,resol}(cpu) = \begin{cases} 2 \times 10^{-3} \times cpu + 1 \times 10^{-3} & \text{if cpu} \in [0\%, 60\%] \\ 4 \times 10^{-3} \times cpu + 3.9 \times 10^{-3} & \text{if cpu} \in [60\%, 100\%] \end{cases}$$

$$t_{DNSSEC}^{BIND,\,resol}(cpu) = \begin{cases} 0.8 \times 10^{-3} \times cpu + 1 \times 10^{-3} & \text{if cpu} \in [0\%, 60\%] \\ 8.5 \times 10^{-3} \times cpu + 1.1 \times 10^{-3} & \text{if cpu} \in [60\%, 100\%] \end{cases}$$

$$t_{DNSSEC-validation}^{BIND,\,resol}(cpu) = \begin{cases} 2.3 \times 10^{-3} & \text{if cpu} \in [0\%, 60\%] \\ 9 \times 10^{-3} & \text{if cpu} \in [60\%, 90\%] \\ 5 \times 10^{-3} & \text{if cpu} \in [90\%, 95\%] \\ 16 \times 10^{-3} & \text{if cpu} \in [95\%, 100\%] \end{cases}$$

$$t_{DNS}^{NSD,\,auth}(cpu) = 27.930 \times 10^{-6} e^{3.5903 \times cpu^{3.2748}} + 63.739 \times 10^{-6}$$

$$t_{DNSSEC}^{NSD,\,auth}(cpu) = 12.772 \times 10^{-6} e^{5.1367 \times cpu^{8.7832}} + 117.65 \times 10^{-6}$$

$$t_{DNS}^{UNBOUND,\,resol}(cpu) = \begin{cases} 1 \times 10^{-3} \times cpu + 0.2 \times 10^{-3} & \text{if cpu} \in [0\%, 95\%] \\ 7 \times 10^{-3} & \text{if cpu} \in [95\%, 100\%] \end{cases}$$

$$t_{DNSSEC}^{UNBOUND,\,resol}(cpu) = \begin{cases} 0.8 \times 10^{-3} \times cpu + 0.2 \times 10^{-3} & \text{if cpu} \in [0\%, 80\%] \\ 30 \times 10^{-3} \times cpu - 23 \times 10^{-3} & \text{if cpu} \in [80\%, 100\%] \end{cases}$$

$$t_{DNSSEC-validation}^{UNBOUND,\,resol}(cpu) = \begin{cases} 2 \times 10^{-3} & \text{if cpu} \in [0\%, 60\%] \\ 9 \times 10^{-3} & \text{if cpu} \in [60\%, 75\%] \\ 3 \times 10^{-3} & \text{if cpu} \in [75\%, 95\%] \\ 22 \times 10^{-3} & \text{if cpu} \in [95\%, 100\%] \end{cases} \tag{A.1}$$

A.1.3 Update Operation Cost

This section expresses the number of additions that can be accomplished per second according to n_{add}, the number of `add` sent per `nsupdate` message. Equations are derived from figures 1.6 in Section 1.5.4.

$$addNbr_{DNS}(n_{add}) = 5000 \times (1 - e^{-n_{add}/40})$$
$$addNbr_{DNSSEC}(n_{add}) = 24 - 10 \times e^{-n_{add}/2.5}$$

A.1.4 Impact of Cache Hit Rate

This section measures the impact of the Cache Hit Rate of the traffic. For traffic with various CHR, we measure the query rate when the CPU load reaches 100%. Equations express the Added Query Rate (AQR) for a given CHR, the percentage of added queries that can be treated as compared to the number of queries treated with a $CHR = 0$. Equations are derived from figure 1.8 in Section 1.5.5.

$$AQR_{DNS}^{BIND}(CHR) = 1.35\,10^{-4}\,e^{15.3\,CHR^{0.278}} + 1.25\,10^{-4}$$

$$AQR_{DNSSEC}^{BIND}(CHR) = 1.37\,10^{-4}\,e^{15.3\,CHR^{0.291}} + 1.51\,10^{-4}$$

$$AQR_{DNSSEC\_validation}^{BIND}(CHR) = 1.66\,10^{-4}\,e^{15.7\,CHR^{0.429}} + 1.5\,10^{-4}$$

$$AQR_{DNS}^{UNBOUND}(CHR) = 1.27\,10^{-4}\,e^{15\,CHR^{0.210}} + 1.5\,10^{-4}$$

$$AQR_{DNSSEC}^{UNBOUND}(CHR) = 1.18\,10^{-4}\,e^{15\,CHR^{0.204}} + 1.5\,10^{-4}$$

$$AQR_{DNSSEC\_validation}^{UNBOUND}(CHR) = 1.48\,10^{-4}\,e^{16.3\,CHR^{0.472}} + 1.5\,10^{-4} \tag{A.2}$$

AQR is derived from equations below :

$$q_{DNS}^{BIND}(CHR) = \left[1.35 \times 10^{-4}\, e^{15.3\, CHR^{0.278}} + 1.25 \times 10^{-4}\right] q_{DNS}^{BIND}(0)$$

$$q_{DNSSEC}^{BIND}(CHR) = \left[1.37 \times 10^{-4}\, e^{15.3\, CHR^{0.291}} + 1.51 \times 10^{-4}\right] q_{DNSSEC}^{BIND}(0)$$

$$q_{DNSSEC-validation}^{BIND}(CHR) = \left[1.66 \times 10^{-4}\, e^{15.7\, CHR^{0.429}} + 1.50 \times 10^{-4}\right] q_{DNSSEC-Val}^{BIND}(0)$$

$$q_{DNS}^{UNBOUND}(CHR) = \left[1.27 \times 10^{-4}\, e^{15.0\, CHR^{0.210}} + 1.50 \times 10^{-4}\right] q_{DNS}^{UNBOUND}(0)$$

$$q_{DNSSEC}^{UNBOUND}(CHR) = \left[1.18 \times 10^{-4}\, e^{15.0\, CHR^{0.204}} + 1.50 \times 10^{-4}\right] q_{DNSSEC}^{UNBOUND}(0)$$

$$q_{DNSSEC_validation}^{UNBOUND}(CHR) = \left[1.48 \times 10^{-4}\, e^{16.3\, CHR^{0.472}} + 1.50 \times 10^{-4}\right] q_{DNSSEC_validation}^{UNBOUND}(0)$$

A.2 Configuration files

A.2.1 Authoritative server configuration

For `BIND`, the following DNSSEC configuration file is used. `zone.signed` is the zone file, `key-directory` defines the directory where keys are located for dynamic updates, and `dnssec-enable` indicates the server has to deal with security extensions. This option is set to *yes* in current BIND9 version. The DNS configuration file can easily be derived from this one.

```
zone "."{
    type master;
    file "/etc/root_zone";
};
zone "test."{
    type master;
    file "/root/conf/zone.signed";
    key-directory "/root/conf/";
};
options{
    dnssec-enable yes;
};
```

For `NSD`, the following configuration is used. `NSD` does not support dynamic updates so the key repository does not need to be specified, and the server enables DNSSEC by default.

```
server:
    username: ""
    chroot: ""
    verbosity: 1
    debug-mode: yes
    port: 53
    ip4-only: yes
    ip6-only: no
zone:
    name: .
    zonefile: "/etc/root_zone"
zone:
    name: test.
    zonefile: "/root/conf/zone.signed"
```

A.2.2 Resolving server configuration

For BIND, the following configuration file is used. `dnssec-validation` indicates the server has to proceed to signature checks and other DNSSEC operations. `trusted-keys` specifies the keys that can be trusted by the cache to perform DNSSEC validation.

```
zone "."{
    type hint;
    file "/root/conf/cache/db.root";
};
options{
    dnssec-enable yes;
    dnssec-validation yes;
};
trusted-keys {
    "test." 257 3 7 "AwEAA[...]aM/";
};
```

For `UNBOUND`, the following configuration file is used. `validator iterator` specifies that the resolving server has to proceed to DNSSEC verifications and `trust-anchor-file` specifies the trusted keys for the validation.

```
stub-zone:
    name: "."
    stub-addr: 192.168.2.2
    stub-prime: no
server:
    username: ""
    chroot: ""
    verbosity: 1
    logfile: ""
    access-control: 0.0.0.0/0 allow
    interface: 0.0.0.0
    port: 53
    do-ip4: yes
    do-ip6: no
rrset-cache-size: 20m
    module-config: "validator iterator"
    trust-anchor-file: "/root/conf/dsset-test."
```

A.2.3 Zone files

The `zone.signed` file for `BIND` and `NSD` is as follows. We did not indicate the full value for the keys and the signatures, and values might differ depending on the tests being performed. We mentioned in the file the key id used to generate the signature. We can easily derive the zone file used without DNSSEC.

```
For DNSSEC, the zone files
; dnssec_signzone version 9.6.0-P1
test.  1      SOA   ns.test. username.test. (
            42666      ; serial
            86400      ; refresh (1 day)
            86400      ; retry (1 day)
            2419200    ; expire (4 weeks)
            3600       ; minimum (1 hour)
                              )
       1      RRSIG   SOA [...]; 28550
       1      NS      ns.test.
       1      RRSIG   NS  [...]; 28550
       3600   DNSKEY  256 [...] ; key id = 28550
       3600   DNSKEY  257 [...] ; key id = 64095
       3600   RRSIG   DNSKEY [...] ; 28550
       3600   RRSIG   DNSKEY [...] ; 64095
       0      NSEC3PARAM 1 0 100 42
       0      RRSIG   NSEC3PARAM [...]; 28550
ns.test.    1      A       192.168.1.2
            1      RRSIG   A [...];  28550
test0.test. 1      A       1.2.3.4
            1      RRSIG   A [...]; 28550
test1.test. 1      A       1.2.3.4
            1      RRSIG   A [...]; 28550
[...]
VV[...]G4.test. 3600  NSEC3 [...] VV[...]IA A RRSIG
                3600  RRSIG  NSEC3 7 2 [...]; 28550
VV[...]IA.test. 3600  NSEC3 [...] VV[...]7R A RRSIG
```

Appendix B

Résumé Étendu en Français

B.1 Introduction

B.1.1 L'objectif de la Thèse

Une des problématiques majeure de sécurité pour les opérateur est de permettre à ses utilisateurs de maintenir la sécurité d'une communication même au travers d'un réseau qui ne soit pas de confiance. Pour l'utilisateur, une communication est établie entre deux identifiants, et ceci indépendamment des mouvements et changements de réseau de l'utilisateur. Autrement dit, l'opérateur doit permettre cette communication entre identifiants possible grâce au système DNS [Moc87a, Moc87b], et fournir les mécanismes réseaux nécessaires afin que la communication puisse être maintenue quand le client bouge et change d'adresse. Dans cette thèse nous nous somme concentrés sur les aspects sécurités et plus exactement:

- **DNSSEC:** DNSSEC [AAL+05a, AAL+05c, AAL+05b] définit comme sécuriser la résolution d'un nom de domaine. La sécurité a un coût que nous commençons par évaluer avant de proposer des architectures permettant aux ISPs de migrer des plateformes de Service de Résolution de DNS vers DNSSEC.
- **IPsec:** IPsec [KS05, Ken05a, Ken05b, KHNE10, Ero06] définit comment sécuriser une communication IP. Dans cette thèse nous définissons une extension qui permet à un utilisateur de maintenir une communication sécurisée par IPsec pour un terminal mobile, Multihomé, et avec de Multiples Interfaces.

DNSSEC: Un utilisateur a rarement la notion de l'adresse IP utilisée. Il établit généralement des communications sur la base de noms. Le service DNS se charge alors de faire le lien entre le nom et l'adresse IP. Le nom est généralement appelé Fully Qualified Domain Name (FQDN), et le Service qui fait la correspondance entre le FQDN et l'adresse IP est le Domain Name System (DNS) [Moc87a, Moc87b]. Le DNS, défini dans les années 80, n'a pas été conçu afin de sécuriser l'échange DNS entre l'utilisateur et le correspondant. Il en résulte que le DNS peut être utilisé afin d'intercepter des communications en fournissant une mauvaise adresse IP. Ainsi, une personne souhaitant se connecter à un site donnée *www.monsite.fr* va demander au service DNS l'adresse IP correspondante. Un service corrompu, va fournir une adresse non légitime, et l'utilisateur se connectera à un site non légitime, pensant être connecté à *monsite.fr*. Pour éviter cette situation, et permettre à tout utilisateur de s'assurer que l'adresse IP renvoyée par le Service DNS est bien légitime, l'IETF a défini une extension de sécurité DNSSEC [AAL+05a, AAL+05c, AAL+05b]. Cette extension utilise de la cryptographie pour, entre autre, établir une chaîne de confiance entre les différents domaine DNS et sous domaine —dans notre exemple entre le *.fr* et *www.monsite.fr* —et pour certifier l'information envoyée —dans notre exemple certifier que l'adresse IP envoyée est bien celle hébergée par le responsable de la zone. Pour un service comme le DNS, défini pour être "léger", l'ajout de calculs cryptographiques impacte fortement les performances des serveurs. Entre autres pour les ISP qui doivent traiter un grand nombre de requêtes simultanée, la migration vers DNSSEC parait difficilement envisageable. Dans cette thèse, nous nous sommes attachés à mesurer le coût d'une telle migration et à définir et évaluer des architecture permettant d'optimiser les plateformes de résolution DNSSEC.

IPsec: L'objectif de cette thèse est d'étudier comment établir et conserver une communication sécurisée de bout en bout dans un environnement Mobile, Multihomé et avec des Interfaces Multiples. On cherche à sécuriser les communications au niveau du réseau de façon à ce que la sécurité soit indépendante des applications et puisse être implémentée de manière générique et transparente pour toutes les applications. Cette capacité à fournir un accès sécurisé et transparent pour les applications intéresse les opérateurs qui peuvent ainsi adapter la sécurité d'une communication en fonction du réseau utilisé, et ce, de manière transparente pour l'application et l'utilisateur.

Ainsi une communication au sein d'un réseau de confiance comme le réseau de l'opérateur pourra se contenter d'une sécurité au niveau radio et sécuriser l'attachement à ce réseau de confiance. En revanche, lorsque l'utilisateur utilisera des réseaux intermédiaires qui ne sont pas de confiance, une sécurité au niveau IP ou au niveau des couches supérieures est alors indispensable. Une sécurité au niveau IP n'impacte pas les applications, alors qu'une sécurité au niveau transport (TLS [DR08]/DTLS [Phe08]) ou au niveau des applications ne se fait pas de manière transparente pour les applications.
Ainsi, au cours de cette thèse, nous nous sommes concentrés sur le protocole IPsec [KS05] afin de sécuriser les communications au niveau IP. De manière générale, ces besoins en sécurité sont nécessaires lorsque l'utilisateur se trouve dans une situation de mobilité, i.e. lorsqu'il n'est ni chez lui ni sur son lieux de travail. Lorsque l'utilisateur utilise un réseau qui n'est pas maintenu par son opérateur, l'utilisateur considère ce réseau comme un réseau qui n'est pas de confiance. D'un point de vue sécurité, cela signifie qu'il doit protéger les communications qui transitent par ce réseau. D'un point de vue transport, ces réseaux non maintenus par l'opérateur peuvent fournir une qualité de service moindre. Par exemple, un point d'accès peut se rebooter à tout moment, la mobilité de l'utilisateur n'est pas gérée comme elle l'est dans les Réseaux d'Accès Radio. Afin de palier à ces inconvénients, des protocoles de transport comme Stream Control Transmission Protocol (SCTP) [Ste07] ou Multi Path TCP (MPTCP) [FRH$^+$11] ont été définis. Il permettent entre autres à un utilisateur de changer l'adresse IP d'une communication (Mobilité), de s'attacher à plusieurs points d'accès simultanés (Multiples Interfaces), ou de prévenir le correspondant qu'en cas de rupture de communication, l'utilisateur reste joignable sur une Interface Alternative (Multihoming). Ces protocoles reposent sur une sécurité du mode transport qui reste très liée à l'application. Par exemple pour SCTP, la sécurité s'inspire beaucoup de TLS. L'idée de cette thèse est de doter IPsec de fonctionnalités similaires de sorte que les mêmes fonctionnalités puissent être effectuées avec une communication sécurisée par IPsec.

IPsec et DNSSEC sont complémentaires. Cela signifie, que si DNSSEC n'est pas utilisé et que le DNS renvoie une adresse non légitime, alors la phase d'authentification d'IPsec devrait détecter la non légitimité de l'hôte et annuler l'établissement de la communication. En considérant ce cas de figure, c'est bien parce qu'ils sont complémentaires qu'ils assurent la sécurité de la communication. En effet, DNSSEC évite toute indirection due au DNS. Ce sont des attaques "facilement" réalisables, dans la mesure où elles reposent sur une faiblesse de conception du protocole DNS [Kam08b, Kam08a], et DNSSEC est la seule alternative à cette faiblesse. DNSSEC n'est pas suffisant, car si j'envoie un paquet IP vers une certaine destination, IP n'étant pas protégé, je n'ai aucune garantie que ma communication n'est pas usurpée au niveau IP. Ces attaques semblent toutefois plus "difficiles" à mettre en œuvre, car elles se font au niveau du routage. IPsec authentifie le nœud correspondant, et donc peut détecter des attaques faites au niveau DNS. En théorie, c'est effectivement le cas, mais la pratique montre qu'un certificat corrompu ou le bug d'une implémentation sont toujours existants. Ceci est aussi vrai pour les implémentations de DNSSEC, mais la probabilité de bugs ou failles sur deux protocoles distincts augmente la sécurité. De plus, les équipes opérationnelles administrant IPsec et DNSSEC sont distinctes. DNSSEC utilise sa propre structure de clé, alors que IPsec repose sur une PKI pour les authentifications à base de certificats. Dans cette thèse, nous n'avons pas pas exemple essayer de combiner IPsec à DNSSEC, par exemple en utilisant DNSSEC pour héberger. Bien entendu, lorsque les communications ne sont pas protégées par IPsec ou TLS / DTLS, DNSSEC tient lieu de garde fou, et vice et versa. Aussi nous est-il apparu dans cette thèse aussi important de maintenir des communications sécurisées au cours d'opérations de Mobilité que d'établir une relation de confiance entre le FQDN et l'adresse IP.

B.1.2 Organisation du résumé

La suite du résumé est construite de la façon suivante. La section B.2, constitue la première partie de ce résumé. Dans un premier temps, elle mesure les différences de performance entre DNS et DNSSEC sur plusieurs implémentations. Ensuite, on se base sur ces mesures afin d'évaluer des architectures qui optimisent les ressources engagées par la plateforme. Le principe des architectures proposée, est de répartir les FQDNs entre les nœuds de la plateforme, afin d'éviter que deux nœuds effectuent des requêtes similaires. La difficulté consiste à trouver une façon de répartir les FQDNs parmi les nœuds pour que chaque nœud est une charge équivalente et ainsi éviter les déséquilibres de charge. La seconde partie est abordée dans la section B.4. Elle est consacrée à l'utilisation d'IPsec afin de maintenir un niveau de sécurité des communications indépendante de la confiance associée au réseau utilisé pour les communications de l'utilisateur. Cette section expose les principes d'IPsec, décrit les impacts de la Mobilité, du Multihoming et des Interface Multiples sur IPsec. Ensuite elle présente MOBIKE-X [Dan09b], le protocole que nous avons défini et implémenté ainsi que les performances de ce dernier. Enfin, cette section B.4.5 étudie comment les ISPs peuvent tirer parti des réseaux WLAN afin de limiter la charge des réseaux 3G / 4G. La section B.5 conclut ce résumé.

B.2 Optimisation des Architecture de plateformes Résolution pour le déploiement d'un nommage sécurisé: DNSSEC

Cette partie est essentiellement consacrée à l'étude de la migration des plateformes de résolution DNS vers DNSSEC. La section B.2.1 présente brièvement DNSSEC, son déploiement actuel, et montre qu'une migration vers DNSSEC est quasi inévitable. La section B.3 présente des mesures expérimentales, et évalue le coût de la migration de DNS à DNSSEC. Dans cette section, on se concentre essentiellement sur les plateformes de résolution. Les deux sections qui suivent proposent des architectures qui permettent d'optimiser les ressources nécessaires aux plateformes de résolution. Dans les deux cas, le principe de base est d'éviter que deux nœuds distincts de la plateforme n'effectuent simultanément la résolution d'un même nom de domaine. Ainsi, on cherche à attribuer à chaque FQDN un nœud dit *Nœud Responsable*. La section B.3.3 évalue les performances d'une plateforme où les nœuds qui la composent sont organisés sous la forme d'un Distributed Hash Table (DHT) et se répartissent la charge des résolutions. Cette plateforme compare différentes architectures de DHT. Suite aux bonnes performances des architectures introduisant des mécanismes pro-actifs, la section B.3.4 propose et décrit le déploiement, sur la base d'une capture de trafic DNS, d'une architecture qui maintient un cache réparti pour les noms de domaines les plus fréquents, et qui utilise une architecture DHT pour la résolutions des autres noms de domaines.La section B.3.5 conclut cette partie.

B.2.1 Description de DNSSEC

Le protocole DNSSEC [AAL+05a, AAL+05c, AAL+05b] est l'extension de sécurité qui permet de sécuriser le système DNS [Moc87a, Moc87b]. Les mécanismes introduits par DNSSEC sont:

- la constitution d'une chaîne de confiance qui permet à une zone père de désigner de manière sûre sa sous-zone. Pour ce faire, la zone père héberge dans sa zone un enregistrement qui contient le hash de la clé publique identifiant la zone fille qui est la *Key Signing Key* ou *KSK*. La zone fille prouve alors la possession de la clé privée en signant des enregistrements.

- la signature des enregistrements d'une zone avec la *Zone Signing Key* ou *Zone Signing Key.*

- la preuve de non existence, qui permet à un serveur autoritaire de prouver, en cas d'absence de réponse, que le FQDN recherché n'existe pas. Cela évite par exemple qu'un tiers ne réponde qu'un domaine n'existe pas à la place du serveur autoritaire.

Actuellement, la standardisation de DNSSEC au sein de l'IETF est finalisée. DNSSEC a été implémenté au sein de nombreuses distributions - Internet Systems Consortium[1] (`BIND9`), NLnetLabs[2] (`NSD`[3] et `UNBOUND`[4]), Microsoft, Nominum (`ANS` and `CNS`), Secure64. DNSSEC complexifie également la gestion des zones et rend les erreurs de signatures très contraignantes dans la mesure où la zone est inaccessible. Des outils de management sont également disponibles comme Opendnssec[5] par exemple. Au niveau du déploiement de DNSSEC, la plupart des Top Levels Domains (TLD) ont déployé DNSSEC, avec au moins la mise en place de la chaîne de confiance. Côté ISP, seul Comcast [Com] a effectué officiellement des expérimentation avec DNSSEC.

B.3 Coût d'une migration de DNS vers DNSSEC

Si l'Internet semble progressivement migrer de DNS vers DNSSEC, les Opérateurs qui effectuent les résolutions DNS, ne semblent pas les plus actifs dans cette migration. Or c'est bien pour eux que la migration est la plus délicate. Par ailleurs, les ISPs ont également une forte relation de confiance avec leurs utilisateurs et sont alors poussés à déployer DNSSEC afin de se prémunir contre les vulnérabilités du DNS dévoilées par Dan Kaminsky [Kam08b, Kam08a].

Des mesures expérimentales nous ont permis de mettre en évidence le coût de la migration de DNS vers DNSSEC. Les mesures ont été faites à la fois dans le cas d'un serveur autoritaire, et d'un serveur cache. Nous avons testé plusieurs implémentations `BIND9` et `UNBOUND`/`NSD` [MGL10, Mig10]. A partir des performances mesurées, nous pouvons en déduire l'impact sur les ressources plateforme, i.e. le nombre de nœuds qu'il faudrait ajouter à la plateforme afin qu'elle puisse traiter un même trafic. Le tableau B.1 met en évidence le nombre de nœuds de la plateforme pour les différentes configurations ainsi que les différents ratios, entre les configurations ou entre les implémentations. Notons que seul les ratios sont à considérer car le nombre de nœuds dépend du type de serveur. Le tableau montre que pour les plateformes de résolution, la migration vers DNSSEC peut nécessiter jusqu'à 4.25 fois plus de ressources, et que les implémentations présentent des variations de performances qui vont du simple au double. Pour les serveurs autoritaires, le coût de la migration vers DNSSEC est d'environ 30%.

Node	DNS	DNSSEC	DNSSEC-VAL
BIND9	80 (1*)	87 (1.09*)	160 (2.00*)
UNBOUND	20 (1*)	24 (1.20*)	85 (4.25*)
$\frac{\text{UNBOUND}}{\text{BIND9}}$ (IR**)	0.25	0.28	0.53

(A) plateforme de Resolution

Node	DNS	DNSSEC
BIND9	29 (1.00*)	38 (1.31*)
NSD	9 (1.00*)	12 (1.33*)
$\frac{\text{UNBOUND}}{\text{BIND9}}$ (IR**)	0.31	0.32

(B) plateforme Authoritaire

TABLE B.1: Evaluation du nombre de nœuds – (*) PR), (**) IR

[1] `http://www.isc.org`
[2] `http://www.nlnetlabs.nl`
[3] `http://www.nlnetlabs.nl/projects/nsd/`
[4] `http://www.unbound.net`
[5] `http://www.opendnssec.org`

B.3.1 Intérêts d'optimiser des plateformes de résolutions

Il est difficilement concevable pour un ISP de multiplier par 5 la taille de ses plateformes. D'une part, les plateforme DNS sont assez importantes et, pour les captures que nous avons considérées, les plateformes de résolution comprennent 4 clusters d'une 20^{aine} de nœuds chacun —La capture que nous avons utilisé provient d'un cluster de 18 nœuds. De plus les trafic DNS sur les 15 dernières années n'a cessé d'augmenter d'environ 8% par mois, et il est a craindre qu'avec le déploiement des CDNs, ou des DNS load balancer qui se basent sur le DNS, ce trafic continue d'augmenter. En effet, CDNs et DNS Load Balancers utilise le DNS pour rediriger l'utilisateur vers le serveur de contenu le plus proche ou le plus disponible. La disponibilité d'un serveur varie rapidement, et pour bénéficier de cette dynamicité, la validité des réponses renvoyées par le serveur DNS ne doivent pas être trop longue. Il en résulte que les FQDNs sont stockés beaucoup moins longtemps dans les caches et doivent plus régulièrement effectuer des résolutions pour les noms de domaines très demandés. Le temps durant lequel la réponse est stockée dans le cache est définie par le *Time To Live* (TTL). Ainsi les perspectives d'évolutions du trafic DNS nous poussent à repenser l'architecture des plateformes de résolutions, et à considérer comment optimiser ces plateformes.

B.3.2 Première Optimisation de la plateforme de Résolution en répartissant le trafic selon les FQDNs

L'architecture actuelle d'une plateforme DNS est un ensemble de nœuds indépendants entre eux auxquels un Load Balancer envoie des requêtes DNS. Le Load Balancer répartit généralement les requêtes DNS entre les nœuds en effectuant une fonction de hachage sur les adresses IP. Dans le cas de nos plateformes opérationnelles, la fonction de hachage est un XOR entre les 24 bits de poids faibles des adresses source et destination. On nommera IP_{XOR} un tel type de Load Balancer. L'avantage de IP_{XOR} est qu'il offre une bonne répartition du trafic sur les nœuds, i.e. chaque nœud reçoit une charge à peu près équivalente. De plus la fonction de hachage est très peu coûteuse, et travailler sur les adresses IP permet de travailler sur des données fixes et ne nécessite pas d'analyser le paquet. Il en résulte que les requêtes sont les noms de domaine les plus populaires, sont réparties de manière uniforme parmis les nœuds. Chaque nœud effectue les résolutions pour ces noms de domaine, et stockent ces noms de domaines au sein de leur cache. De plus, ces architectures ont du mal à passer à l'échelle. En effet, lorsque l'on ajoute un nouveau nœud, ce dernier va effectuer des résolutions en doublon pour les noms de domaine les plus populaires, les stocker dans son cache, avant de véritablement apporter une plus value en ressources.

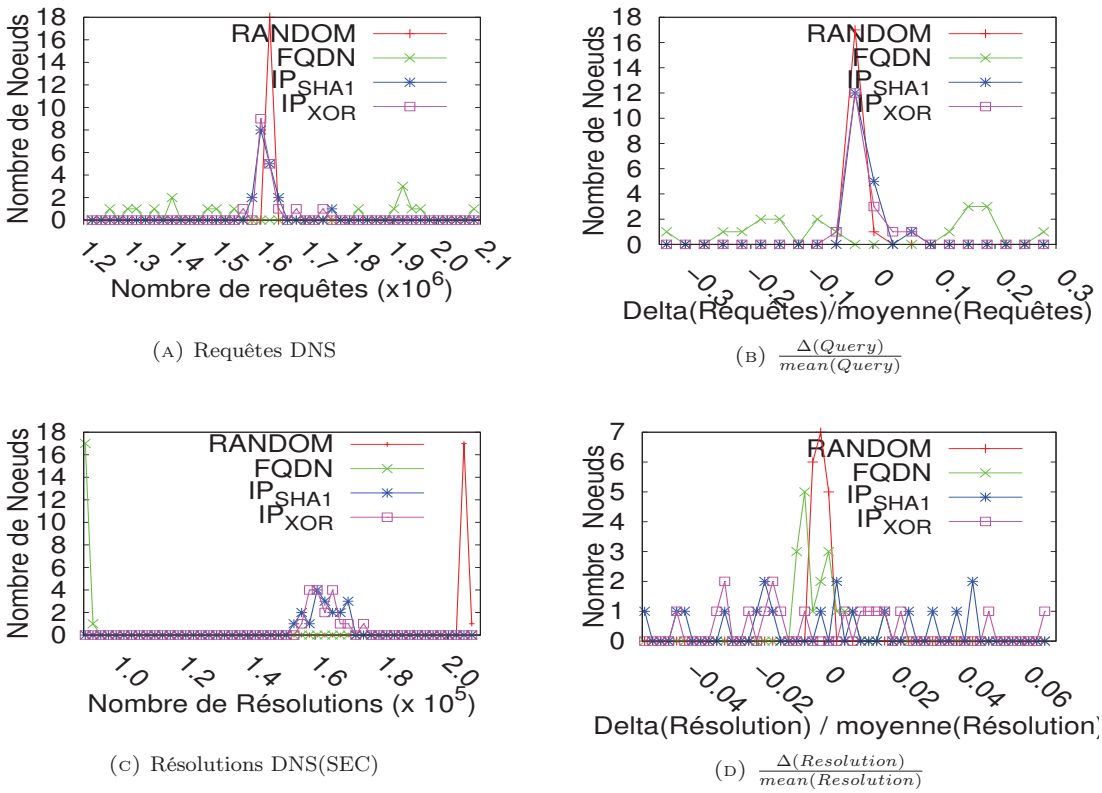
Il est clair qu'une telle architecture n'est pas optimale. Pour éviter des résolutions redondantes, une possibilité est d'assigner chaque nom de domaine à un nœud particulier. Dans une architecture avec un Load Balancer, —tel que IP_{XOR}—, ceci peut être réalisé en effectuant un hash du FQDN. On appellera une telle architecture *FQDN*. Une autre possibilité est de faire coopérer les nœuds entre eux, et, dans le cas où un nœud reçoit une requête DNS, dont il n'est pas responsable, il redirigera la requête, ou effectuera une requête vers le nœud responsable. Ce type d'architecture correspond aux architectures *Distributed Hash Table* ou DHT. L'avantage de ces deux architectures est qu'elles réduisent le nombre de FQDNs au sein des caches, et donc accélèrent les cache lookups. De plus, elles limitent le nombre de résolution DNS à effectuer sur l'Internet. Pourquoi alors ce type d'architecture n'a-t-elle pas été utilisée dans le cas du DNS? Il y a sans doute plusieurs raisons à cela. Premièrement le service de résolution DNS, fut pendant longtemps considéré comme un service nécessitant peu de ressources, fournit gracieusement par l'ISP, et n'ayant pas ou peu de valeur ajoutée. Ensuite, effectuer avec le protocole DNS une requête sur l'Internet ou auprès d'un autre nœud de la plateforme est à peu près similaire d'un point de vue ressources consommées,

à ceci près que les résolutions sur l'Internet introduisent une grande latence. La taille des caches a complètement été sous-estimée. Par contre, le très gros avantage d'une architecture de type IP_{XOR} est sa simplicité. Les nœuds sont indépendants, donc ajouter une nœud ou en retirer un n'impacte pas les autres. En revanche, les architectures de type DHT doivent tenir compte des interactions entre les nœuds, et les architectures de type *FQDN* nécessitent de déployer des Load Balancer spécifiques, et de définir des fonction de répartition du trafic.

La figure B.1 présente les caractéristiques d'un trafic DNS que l'on répartit de différentes façons. A partir d'une capture de trafic réel prise en heure de pointe traité par une plateforme de 18 nœuds, on rejoue le trafic et représente en ordonnée le nombre de nœuds pour lesquels on mesure X —nombre de requêtes, nombre de résolutions, ainsi que leur écart relatif à la valeur moyenne—. IP_{XOR} répartit le trafic en appliquant une fonction de hachage SHA1 sur les adresses IP source et destination, et *FQDN* applique cette même fonction de hachage sur les FQDNs de la requête DNS. XOR répartit le trafic selon les adresses IP et en utilisant la fonction de hachage XOR. Cette fonction est actuellement déployée dans des Load Balancers de notre plateforme opérationnelle. La simulation nous permet de comparer l'efficacité de XOR avec SHA1. Enfin *RANDOM* représente un Load Balancer qui répartirait le trafic de manière aléatoire, pour chaque requête DNS. Notons que *FQDN* définit un Load Balancer qui effectue un hash du FQDN. La répartition des requêtes et des résolutions qui en résulte est similaire dans le cas où l'on utilise un Load Balancer ou une architecture DHT.

Synthèse: La figure B.1c montre que *FQDN* réduit de 60% le nombre de résolutions et que les résolutions sont réparties de manière uniforme sur les nœuds de la plateforme. La figure B.1d représente la distribution de l'écart relatif et montre que *FQDN* présente une très bonne distribution. En revanche les figures B.1a et B.1b montrent que *FQDN* présente une très mauvaise répartition des requêtes. Au final, une répartition basée sur les FQDN permet de réduire de 30% les ressources de la plateforme. Toutefois, pour permettre ce déploiement, il faut trouver un moyen d'uniformiser la charge parmi les nœuds. Pour cela nous avons envisagé trois méthodes :

- Définir une table de routage qui oriente chaque requête vers un nœud de la plateforme. La table de routage est calculée de façon à ce que les ressources soient réparties de manière uniforme [MHS+a]. Cette table de routage peut être déployée au sein d'un Load Balancer.
- Utiliser une architecture DHT qui permet de répartir la charge de requêtes entre les différents nœuds, et la charge des résolutions soit grâce à une table de répartition définie par [MHS+a], soit en utilisant des mécanismes de caching proactifs pour les noms de domaine les plus populaires. Ceci est développé dans la section B.3.3.
- Définir une architecture qui présente en front end des caches avec les noms de domaine les plus populaires, et en back end, une architecture de type DHT qui traite les autres FQDNs. On peut considérer cette architecture de deux façons. Une première où les caches et les nœuds DHT sont distincts. Une seconde façon considère une architecture DHT où le cache est distribué sur l'ensemble des nœuds. Cela signifie alors que le trafic est uniformément réparti sur les nœuds DHT via un load balancer, que chaque nœud consulte un premier cache qui contient les FQDNs les plus populaires, et en cas de cache miss, la requête est adressée à la partie DHT du nœud qui va se charger de la résolution. Cette architecture peut tirer partie des cartes réseau accélératrices par exemple. Cette architecture est décrite dans la section B.3.4.



(A) Requêtes DNS

(B) $\frac{\Delta(Query)}{mean(Query)}$

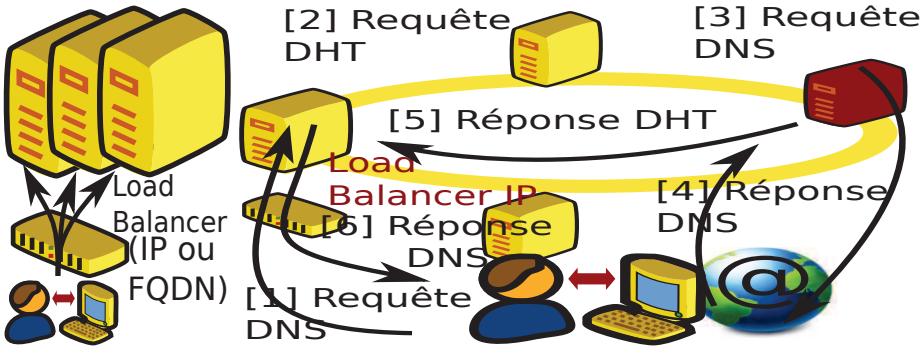
(C) Résolutions DNS(SEC)

(D) $\frac{\Delta(Resolution)}{mean(Resolution)}$

FIGURE B.1: Analyse d'une capture DNS: Distribution des Requêtes, Résolution et Cache Hit Rate (CHR), sur les nœuds de la plateforme

B.3.3 Seconde Optimisation de la plateforme de Résolution DNSSEC à l'aide de DHT

Si pour le DNS, les architectures de type DHT présentaient relativement peu d'intérêts d'un point de vue performances, avec DNSSEC, les ressources économisées peuvent compenser leur complexité. En effet, avec DNSSEC, effectuer une requête DNS auprès d'un nœud nécessite entre 20 et 40 fois moins de ressources CPU. En ce qui concerne l'architecture *FQDN*, avec l'arrivée de cartes accélératrices hardware, un load balancing qui nécessite une analyse du paquet est envisageable. La figure B.2 présente les deux types d'architectures. La figure B.2a représente les architectures de type load balanncer comme *FQDN* et IP_{XOR}, et la figure B.2b représente les architectures à base de DHT, qui impliquent une coopération entre les nœuds. En l'occurrence, la figure B.2b illustre le protocole DHT *Pastry* [RD01a].



(A) *FQDN*, IP_{XOR}, IP_{SHA1}

(B) DHT (Pastry)

FIGURE B.2: Plateformes de résolution DNS(SEC) basées sur un un Load Balancer et une coopération entre les nœuds (DHT)

Dans cette section, on cherche à évaluer les performances obtenues en utilisant une architecture DHT pour la plateforme de résolution. On se base sur le protocole *Pastry* [RD01a]. Les raisons pour lesquelles on privilégie une architecture DHT plutôt qu'une architecture utilisant un Load Balancer sont:

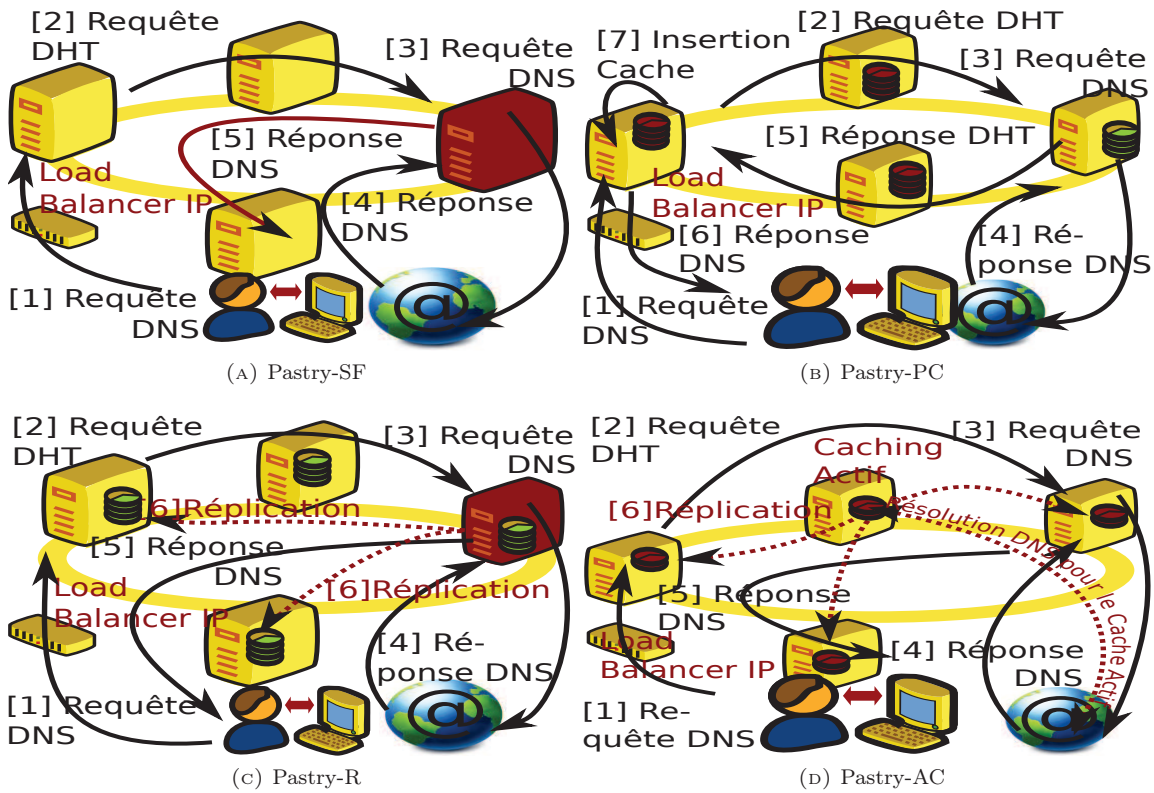
- **Fragilité :** Les Load Balancers sont des éléments fragiles qui établissent une interconnexion entre deux réseaux, soumis à un très fort trafic. En cas de dysfonctionnement, toute la plateforme est hors d'accès.
- **Flexibilité :** Les contraintes auxquelles les Load Balancers sont soumis réduisent les fonctionnalités de ces derniers. En revanche, les serveurs permettent l'installation de services très complexes, et n'ont quasiment aucune limitation.
- **Auto-configuration :** *Pastry* est un protocole qui offre des fonctionnalités d'auto configuration. Ainsi lorsque l'on ajoute un nœud ou que l'on retire un nœud, le système s'auto-configure.

En revanche lorsque l'on parle de *Pastry* dans cette thèse, on ne considère qu'un certain nombre de fonctionnalités de *Pastry*. En effet, *Pastry* a été conçu pour héberger des contenus sur des plateformes comprenant des millions de nœuds, qui pouvaient dynamiquement joindre la plateforme ou la quitter. Chaque nœud étant administré de manière indépendante, les nœuds ne pouvaient alors pas avoir une connaissance de l'ensemble des nœuds de la plateforme. Ceci nécessita la mise en place d'un protocole de routage assez complexe. Dans notre cas précis, notre plateforme de résolution n'est composée que d'un nombre restreint de nœuds, ces nœuds ne sont pas sensés quitter et joindre la plateforme à tout moment et sont administrés par le même administrateur, au sein d'un même LAN. Ces différences font que le protocole de routage initialement prévu pour *Pastry* peut être simplifié. Un autre modification importante est que *Pastry* répartit des contenus au sein des nœuds de la plateforme en fonction de la valeur de l'empreinte du contenu. Dans notre cas, on se permet d'utiliser une fonction différente de celle d'un SHA1, comme par exemple la combinaison d'une table de routage et d'une fonction de hachage afin d'équilibrer la charge au sein des nœuds de manière uniforme.

L'architecture *Pastry* est présentée dans la figure B.2b. L'utilisateur émet une requête DNS vers un nœud de la plateforme (flèche (1)). Ce nœud effectue une recherche dans son cache afin

de vérifier s'il ne possède pas la réponse. Si la réponse se trouve dans son cache, il la renvoie à l'utilisateur. Si la réponse ne se trouve pas dans son cache, le nœuds détermine qui est le *Nœud Responsable* associé à ce FQDN. Il émet ensuite une requête vers le *Nœud Responsable* (flèche (2)). Ce dernier est chargé de renvoyer la réponse attendue, qui peut se trouver dans son cache ou nécessiter une résolution sur l'Internet (échange (3) et (4)). La réponse est renvoyée vers le nœud auquel l'utilisateur s'était adressé (flèche (5)), pour être renvoyée à l'utilisateur (flèche (6)) et éventuellement cachée. Dans cette thèse, lorsque nous parlons de *Pastry*, nous considérons que la réponse n'est pas cachée par un nœud non responsable.

Dans cette section, nous envisageons différentes variantes de *Pastry* représentées dans la figure B.3. *Pastry Straight-Forwarding* ou *Pastry-SF* dont le principe de fonctionnement est décrit dans la figure B.3a fonctionne comme *Pastry* mise à part que le *Nœud Responsable* envoie la réponse directement à l'utilisateur. *Pastry-Passive Caching* ou *Pastry-PC*, dont les échanges sont indiqués dans la figure B.3b, fonctionne comme *Pastry* mais ici le nœud cache les réponses qu'il reçoit du *Nœud Responsable* et qu'il renvoie à l'utilisateur. Avec *Pastry-Replication* ou *Pastry-R*, dont les échanges sont décrits dans la figure B.3c, chaque fois que le *Nœud Responsable* effectue une résolution sur l'Internet, il transmet la réponse sur ses k nœuds voisins. Enfin la figure B.3d décrit *Pastry-Active Caching* ou *Pastry-AC*, une architecture où chaque nœud cache pro-activement les γ FQDN les plus populaires dont il est responsable. Pour toutes ces architecture, on considère que la répartition des FQDNs est telle que les ressources sont uniformément réparties parmi les nœuds de la plateforme [MHS+a]. Pour *Pastry-AC*, cette condition est éventuellement plus discutable.

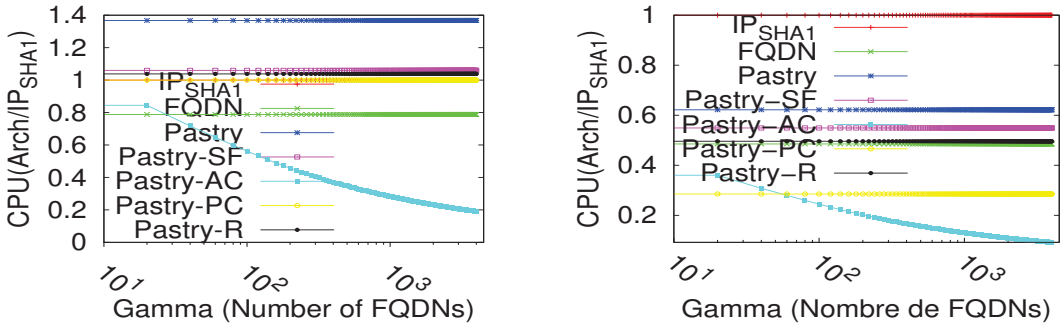


(A) Pastry-SF (B) Pastry-PC

(C) Pastry-R (D) Pastry-AC

FIGURE B.3: Principe des architectures DHT

En modélisant le fonctionnement d'un nœud et en rejouant une capture de trafic réel, on comptabilise les opérations effectuées par chaque nœud et d'après les mesures expérimentales [MGL10, Mig10], on en déduit le pourcentage CPU nécessaire pour chaque nœud, pour traiter le trafic. La figure B.4 représente pour DNS et DNSSEC, le ratio des différentes architecture par rapport à IP_{XOR}, en fonction de γ le nombre de FQDNs que les *Nœuds Responsables* se chargent de mettre dans les caches des autres nœuds. La figure B.4 montre que, dans une configuration DNS, les architectures *Pastry-SF*, *Pastry-PC*, *Pastry-R* ne sont pas rentables en terme de CPU par rapport à une architecture de type IP. Seules *Pastry-AC* et *FQDN* présentent un avantage sur IP_{XOR}. D'après notre modélisation, la comparaison entre *FQDN* et IP_{XOR} coïncide avec les résultats de la simulation en section B.3.1. En revanche, la figure B.4b se place dans le cas où DNSSEC est utilisé, et montre que les architectures basées sur Pastry sont toutes avantageuse sur IP_{XOR}, et que *Pastry-AC* est encore l'architecture la plus avantageuse si les 2000 FQDNs les plus populaires sont cachés de manière pro-active.



(A) γ_{Nbr} (DNS)

(B) γ_{Nbr} (DNSSEC)

FIGURE B.4: Comparaison du ratio CPU des architectures Pastry par rapport à une architecture traditionnelle de Load Balancer en fonction de γ

Pastry-AC est l'architecture la plus avantageuse car elle tire parti de la distribution en loi de puissance de la popularité des FQDN. La figure B.5 classe les FQDNs en fonction de leur popularité et indique en ordonnée le pourcentage du trafic qu'il représente. Avec cette distribution, l'architecture *Pastry-AC* permet raisonnablement de diviser par 4 les ressources nécessaires afin de garantir un service de résolution. Même si les résultats de la figure B.4b montrent que l'on peut améliorer les performances de *Pastry-AC* en augmentant le nombre de FQDNs destinés à enrichir les caches des nœuds, il faut toutefois tenir compte des approximations faites dans nos modèles. En l'occurrence, nous avons considéré que les mises à jour du cache par les *Nœuds Responsables* étaient faites par une seule requête. Si le nombre de FQDNs à cacher augmente, cette approximation n'est plus vraie. De plus nous n'avons pas pris en compte les coûts dus à l'augmentation de la taille des caches. Les résultats expérimentaux que nous avons considérés utilisaient des caches avec peu d'enregistrements.

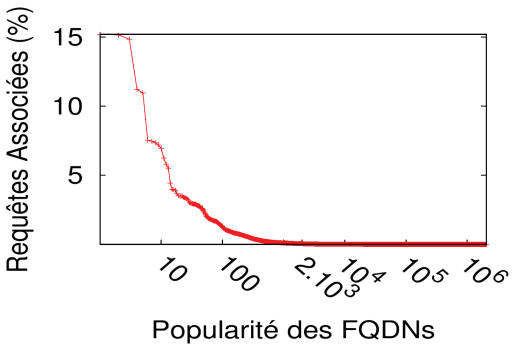


FIGURE B.5: Distribution des FQDNs selon leur popularité

B.3.4 Troisième Optimisation de la plateforme de Résolution DNSSEC avec une architecture PREFETCH

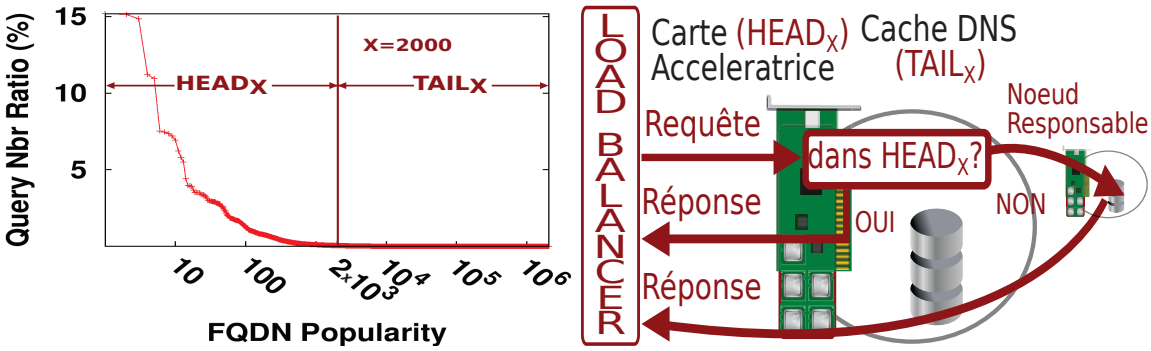
Dans la section B.3.3, nous avons défini que l'architecture la plus efficace consistait à cacher les FQDNs les plus populaires sur l'ensemble des nœuds de la plateforme. Ceci est réalisé par l'architecture *Pastry-AC* qui tire parti du fait que la distribution de la popularité des FQDNs suit une loi de puissance.

Dans cette section, nous considérons ce principe mais nous voulons également tirer parti des performances matérielles et notamment des cartes accélératrices [cav, end], afin d'optimiser les performances de cache lookup au niveau de nœuds. En effet, les deux opérations principales effectuées par un nœud sont la résolution des requêtes DNS(SEC) sur l'Internet qui nécessitent une vérification de signature et les opérations sur les caches. L'architecture *Pastry-AC* optimise la consommation CPU, c'est à dire qu'elle limite le nombre de résolutions à effectuer et permet de répartir uniformément les ressources. *Pastry-AC* optimise également les opérations de cache lookup en réduisant la taille des caches des nœuds grâce à la désignation de *Nœud Responsable* pour chaque FQDN. Toutefois, les caches restent conséquents, et une consultation de cache reste un facteur limitant les performances de la plateforme de résolution.
La figure B.5 montre que les 2000 FQDNs les plus populaires représentent environ 70% du trafic. Par conséquent, pour une architecture de type *Pastry-AC* 70% des requêtes vont correspondre à un cache hit. Une opération de cache lookup est coûteuse pour le nœud Pastry car son cache reste important. Par contre 2000 enregistrements représente peu pour un cache. L'idée de cette section est d'introduire des niveaux de cache, ou d'installer sur chaque nœud une carte accélératrice qui pour chaque requête DNS va consulter si la requête correspond à l'un des 2000 FQDN les plus populaires. Si c'est le cas, la carte renverra la réponse, dans le cas contraire, elle redirigera la requête au nœud Pastry. Le processus de vérifier dans une liste est très simple et peut être poussé sur une carte multi-cores, qui aurait alors plusieurs processus légers et indépendants. Ces processus prendraient en charge 70% du trafic, et les nœud Pastry ne géreraient alors que l'équivalent de 30% du trafic.

A ce stade là, on ne cherche plus à se ramener à une architecture de type *Pastry-AC*. On considère plutôt une architecture Pastry dans laquelle chaque nœud possède une carte qui prend en charge une grande partie du trafic. Les caches des nœuds ne sont pas significativement réduits, mais le nombre de consultations l'est. Pour des raison de coûts, combinés à une grande efficacité des cartes accélératrices, on peut souhaiter mutualiser les cartes accélératrices, dans la mesure où les ressources limitatives sont celles utilisés pour la résolution. La mutualisation des cartes ac-

céléatrices revient alors à considérer un réseau front end qui prend en compte les 2000 FQDNs les plus populaires, et un réseau back end constitué de nœuds Pastry. Cette architecture est appelée $PREFETCH$.

La figure B.6a reprend la distribution de la popularité des FQDNs et définit $HEAD_X$, les FQDNs les plus populaires qui sont cachés soit au sein d'une carte accélératrice, soit au sein d'un réseau front end. $TAIL_X$ désigne l'ensemble des FQDNs qui ne sont pas gérés par la carte accélératrice ou l'architecture front end. X désigne le nombre de FQDNs populaires contenus dans $HEAD_X$, i.e. le nombre d'occurrence de $HEAD_X$. La figure B.6b présente un nœud de l'architecture $PREFETCH$.



(A) X, $TAIL_X$, $HEAD_X$ (B) PREFETCH Node Achitecture

FIGURE B.6: Description de l'architecture $PREFETCH$

Pour déterminer X, $HEAD_X$ et $TAIL_X$ on analyse une capture de trafic réel. On définit alors X comme étant le nombre de FQDNs à ôter du trafic afin d'observer une distribution uniforme des ressources. L'uniformité des ressources peut être estimée selon plusieurs critères. Dans notre cas, nous avons définit X tel que le nombre de requêtes DNS et le nombre de résolutions effectuées par chacun des nœud ne présente pas un écart supérieur au cas où une architecture de type Load Balancing est utilisée. Une fois les X FQDNs ôtés du trafic, l'architecture $PREFETCH_X$ est assimilable à *Pastry*. Pour que X soit stable vis à vis de la fonction de hachage utilisée, on utilise plusieurs fonction de hachage comme SHA1, MD5, CRC32. Ensuite, on vérifie la stabilité de X par rapport au temps, i.e. si la valeur trouvée par l'analyse de notre capture reste toujours valable au cours du temps. Dans notre cas, nous avons vérifié sur une journée que les données déduites de notre capture de 10 minutes restent valides.

X, $HEAD_X$ et $TAIL_X$ définissent l'architecture $PREFETCH$. Les FQDNs les plus populaires $HEAD_X$ sont gérés par la carte accélératrice ou par un front end. $TAIL_X$ est géré soit par le nœud Pastry, soit par le réseau en back end. Pour le réseau en back end, on choisit de déployer une architecture Pastry, essentiellement afin de bénéficier des mécanismes d'auto-configuration. On peut alors reprendre les modèles sur lesquels nous avons appuyé notre étude dans la section B.3.3. Toutefois, nous devons les reprendre avec le trafic $TAIL_X$. En effet, le trafic de $TAIL_X$ présente des caractéristiques différentes du trafic $HEAD_X + TAIL_X$. Si la figure B.6a laisse à penser que $TAIL_X$ est uniforme, un test du χ^2 avec l'hypothèse nulle "$TAIL_X$ *est uniformément répartie*" rejette cette hypothèse. Les résultats obtenus par nos modèles montrent que l'architecture *Pastry-AC*, *Pastry-SF* et *Pastry* présentent les meilleures performances. L'avantage d'utiliser *Pastry-AC*

pour le réseau back end est que l'on peut alors aussi jouer sur le back end afin d'augmenter le nombre de FQDNs cachés. Avec DNSSEC, la différence entre *Pastry* et *Pastry-SF* n'est pas flagrante. Comme les implémentations de *Pastry* sont davantage disponibles que des implémentations de *Pastry-SF*, nous choisissons *Pastry* dans la suite de cette thèse.

Jusqu'à présent, nous nous sommes appuyés sur une modélisation de notre architecture, mais nous n'avons jamais validé cette modélisation à l'aide d'une expérimentation. Nous avons implémenté une plateforme de 10 nœuds Pastry à l'aide de la libraire FreePastry [Fre]. Avec un trafic Uniforme, nous chargeons la plateforme, et relevons le trafic maximum traité par la plateforme lorsque le nombre de nœuds actifs est compris entre 1 et 10. Ces résultats sont alors comparés avec ceux de nos modèles grâce à la mesure d'un coefficient de corrélation. Le coefficient de corrélation proche de 1 valide nos modèles.

B.3.5 Discussion

Suite à cette partie sur les architectures de plateformes de Résolution DNSSEC, les opérateurs sont en mesure de déployer DNSSEC. Les plateformes optimisées ne nécessite pas plus de nœuds que les architectures actuelles déployées afin d'assurer un service de résolution avec DNS. De plus, les optimisations décrites dans cette sections permettent me de diminuer, malgré le déploiement de DNSSEC, la taille des plateformes car à la mutualisation des opérations de résolutions et l'amélioration des caches.

Ainsi à l'issue de cette section, la problématique de : comment migrer un service de Résolution DNS vers DNSSEC est résolue. Notons toutefois, que dans cette partie nous avons résolu ce problème d'un point de vue technique, en dégageant des principes d'architecture. La mise en place et le déploiement doivent encore prendre en compte des critères économiques, qui dépendent des constructeurs.

B.4 MOBIKE-X: Extensions IPsec permettant le support simultané de la Mobilité, du Multihoming et des Interfaces Multiples

La seconde partie est dédiée à la sécurité IPsec dans un contexte de Mobilité, de Multihoming et de Multiples Interfaces. La section B.4.1 décrit brièvement le protocole IPsec, et la section B.4.2 met en évidence l'impact de la mobilité, du multihoming et de Multiples Interfaces sur la configuration IPsec. La section B.4.3 présente les protocoles IPsec liés à la Mobilité et au Multihoming comme MOBIKE [Ero06] et positionne notre protocole MOBIKE-X [Dan09b]. La section B.4.4 mesure les différentes performances en termes de mobilité et de multihoming. Plus précisément, on mesure le temps de réaction d'un système pour se configurer correctement par rapport aux changements réseau et pour adapter une communication protégée par IPsec d'une interface à une autre. L'objectif de ces mesures de performances est d'évaluer comment MOBIKE-X permet d'introduire de nouvelles fonctionnalités et d'améliorer la qualité de service dans un cas de mobilité. Ces performances sont mesurées selon différentes architectures. En effet, les architectures que nous considérons sont :

- un client connecté à un serveur avec une communication protégée de bout en bout.
- un client connecté à un serveur via une passerelle de sécurité, et tunnel le trafic jusqu'a la

Passerelle de Sécurité.

Commecons par considérer l'architecture avec la sécurité de bout en bout. Cette configuration est un peu semblable à l'utilisation de TLS. La seconde architecture que nous considérons est celle plus classique d'un client VPN connecté à une *Passerelle de Sécurité*. L'utilisation d'une communication avec une protection IPsec de bout en bout n'est pour l'instant pas couramment déployée. Cependant nous considérons cette architecture car elle présente certains avantages sur l'utilisation du mode tunnel. D'une part, elle réduit la bande passante ainsi que les opérations de tunneling et de chiffrement. D'autre part, la communication point à point évite les indirections et permet, par rapport aux architectures VPN, de dimensionner la plateforme de sécurité par rapport aux services. Les services cibles sont bien entendu des services avec de fortes exigences temps réel, comme les services de voix ou de jeux. La section B.4.5 se concentre sur la problématique de l'offload. Les opérateurs ne pourront pas supporter la demande de trafic mobile d'ici 5 ans, et envisagent de migrer les flux qui circulent actuellement sur les Réseau d'Accès Radio comme la 3G / 4G vers des réseaux WLAN. Ces derniers n'appartiennent pas forcément à l'opérateur et donc nécessitent de sécuriser la communication. Selon la nature du service, l'une ou l'autre des architecture peut être utilisée. En effet, l'offload pose un double problème. D'une part, un problème de sécurité car l'on utilise un réseau tiers, et d'autre part une problématique plus générale de mobilité de communication. Dans cette section, on s'attache à donner des recommandations sur la manière de gérer l'offload à la fois au niveau des politiques d'offload, et au niveau de la manière de gérer la mobilité de l'application. Enfin, la section B.4.6 conclut cette partie.

B.4.1 Rappel sur IPsec et définition de la Mobilité, Multihoming et Multiples Interfaces sur IPsec

Dans cette partie, nous nous attacherons à effectuer un bref rappel sur IPsec et à définir ce que l'on entend par Mobilité, Multihoming et Multiples Interfaces.

On parle souvent d'IPsec comme un protocole, or il faudrait plutôt le voir comme un ensemble de protocoles permettant de sécuriser les paquets IP d'une communication. Le principe de base d'IPsec est de protéger chacun des paquets IP d'une communication. IPsec définit alors une architecture [KS05] qui définit les différents éléments intervenant pour le chiffrement de chacun des paquets. IPsec définit un protocole IKEv2 [KHNE10] qui permet aux différents nœuds de négocier le matériel cryptographique nécessaire à la protection de la communication. Enfin, IPsec définit également des protocoles qui permettent d'authentifier les paquets de la communication: IP Authentication Header (AH) [Ken05a] ou de chiffrer les paquets IP de la communication avec IP Encapsulating Security Payload (ESP) [Ken05b].
Lorsque nous parlons, dans cette partie de Mobilité, Multihoming et Multiples Interfaces, nous entendons comment garder cohérente la configuration IPsec par rapport aux modifications d'interfaces d'un terminal. C'est donc essentiellement définir d'une part les modifications qui doivent être effectuées sur les deux nœuds communicants et, d'autre part, comment les nœuds peuvent signaler entre eux les modifications de la configuration IPsec à effectuer. Ainsi, on se focalisera sur l'architecture IPsec qui met en évidence l'impact, sur la couche IPsec, d'une modification au niveau des interfaces réseaux, comme un changement d'adresse IP, un ajout d'interface, l'injoignabilité d'une interface par exemple. Ensuite on définira les modifications à apporter au protocole IKEv2 afin de permettre la signalisation. Ce protocole de signalisation sera appelé MOBIKE-X, et fait l'objet de plusieurs drafts présentés à l'IETF [Dan09b, Dan09a, DC12].

Le principe de fonctionnement de l'architecture IPsec est résumé par la figure B.7. IPsec permet

à deux zones de confiance de communiquer via un réseau qui n'est pas de confiance. La zone de confiance est dite *Trusted* et le réseau est dit *Untrusted*. Pour cela, la couche IPsec doit protéger tous les flux allant de la zone *Trusted* vers la zone *Untrusted* et inversement, vérifier les paquets allant de la zone *Untrusted* vers la zone *Trusted*.
Pour chaque paquet IP envoyé sur le réseau, la couche IPsec doit d'abord vérifier le traitement à effectuer. Le traitement est défini par la *Security Policy* (SP) qui se trouve au sein de la *Security Policy Database* (SPD) et qui définit les politiques de sécurité. La SPD définit grâce à des *Traffic Selectors* (TS) comme les adresses IP ou les ports si le paquet doit être rejeté (DISCARD), envoyé sur le réseau en clair, i.e. sans être protégé (BYPASS) ou si le paquet doit être protégé (PROTECT). Nous nous intéressons, dans cette partie, au cas où le paquet doit être protégé, et la Politique de Sécurité, pointe vers une structure indiquant comment le paquet doit être protégé, avec notamment le matériel cryptographique nécessaire. Cette structure est appelée *Security Association* (SA) et est stockée au sein de la *Security Association Database* (SAD). Notons qu'il se peut que la SP ou la SA ne soit pas encore définie au sein du noyau lorsque le paquet est envoyé sur le réseau. Dans ce cas IKEv2 va négocier la SA et la SP tel qu'elles doivent être stockées au sein du noyau.
Lorsqu'un paquet, par exemple chiffré, arrive sur l'interface réseau, la couche IPsec doit le déchiffrer pour l'envoyer vers la zone *Trusted*. Les paquets qui arrivent ne possèdent par comme pour le trafic sortant les même sélecteurs. En effet, pour un paquet chiffré, les ports ne sont pas visibles par exemple, et ne peuvent permettre de sélectionner la SA qui va servir à déchiffrer le paquet. Pour ce faire, on utilise un index: le Security Parameter Index (SPI) qui doit indexer de manière unique les associations de securité. Pour un paquet entrant chiffré, la couche IPsec identifie le SPI, déchiffre le paquet. Outre les clés nécessaires au chiffrement / déchiffrement, la SA contient les *Traffic Selectors* qui ont permis de selectionner la SP. La couche IPsec vérifie alors que le paquet possède bien les bons *Traffic Selectors* avant de finalement renvoyer le paquet vers la zone *Trusted*.

Il apparaît donc clairement que les modifications des paramètres réseau comme les adresse IP vont impacter à la fois les associations de sécurité (SA) et politique de sécurité (SP).

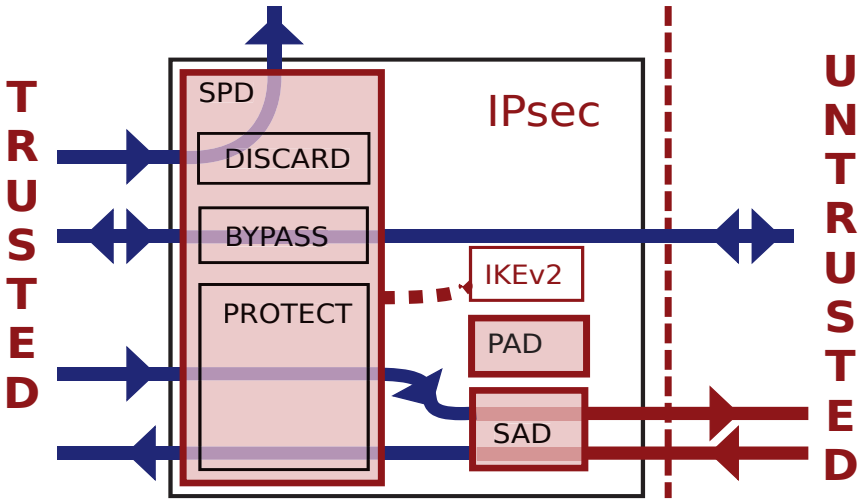
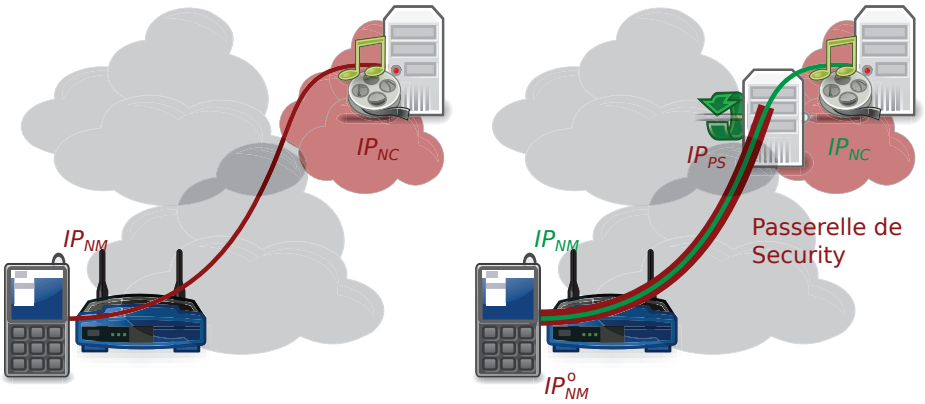


FIGURE B.7: Principe de fonctionnement d'IPsec

IPsec possède deux modes : le mode Transport et le mode Tunnel. La figure B.8 décrit les deux scénarios que nous considérons dans cette thèse, et la table B.2b décrit les configurations

respectives pour les SAD et SPD. Ainsi, La figure B.8a illustre le cas où le Nœud Mobile (NM) est connecté directement au Nœud Correspondant (NC). Avec une connexion directe, on s'attend à ce que ce soit le mode Transport d'IPsec. Les SP et SA correspondantes sont indiquées par la table B.2a. La SPD indique que tout trafic entre l'adresse IP_{NM} et l'adresse IP_{NC} doit être protégé, et ce, peu importe le port. La SP indique que c'est le mode Transport qui est utilisé, et que le chiffrement se fait avec ESP qui utilise $ENCR$ pour l'encryptage et $AUTH$ pour l'authentification. Nous avons mentionné le SPI dans la SPD comme lien permettant de pointer vers la bonne SA pour le trafic sortant, plutôt que le SPI, la plupart des implémentations utilisent un lien propre à l'implémentation comme une adresse de structure par exemple. La SA est indexée par les adresses IP IP_{NM} et IP_{NC} ainsi que le SPI. Ce sont les seuls champs visibles du trafic entrant. La SA contient le matériel cryptographique ainsi que les sélecteurs qui ont servi pour la sélection de la SP.
De manière analogue, l'utilisation du mode Tunnel est représentée par la figure B.8b. Le NM est connecté au NC et leur communication est établie entre IP_{NM} et IP_{NC}. Chaque paquet IP avec IP_{NM} comme IP source et IP_{NC} comme IP destination est encapsulé vers la *Passerelle de Sécurité* en utilisant comme adresses externes IP^o_{NM} et IP_{PS}. Comme indiqué par la table B.2b, les sélecteurs de la SP sont identiques à ceux du mode Transport, mais dans le cas du mode Tunnel, le SP mentionne les adresses externes. Ces adresses externes sont également mentionnées dans la SA, afin de vérifier, une fois le paquet décapsulé qu'il correspond bien à la bonne SP. L'important, dans le mode Tunnel, est de noter que les sélecteurs correspondent aux adresses internes.



(A) ESP Transport (B) ESP Tunnel

FIGURE B.8: Configuration réseau IPsec initiales

B.4.2 Description de l'impact de la Mobilité, Multihoming et Multiple Interfaces sur IPsec

Dans cette thèse, lorsque l'on parle de Mobilité, cela signifie que le NM change d'adresse. Dans le cas de la figure B.8a, en mode Transport, le NM utilise IP^*_{NM} au lieu de IP_{NM}. Dans le cas du mode Tunnel, représenté par la figure B.8b, le NM change son adresse IP externe et utilise $IP^o_{NM}*$ au lieu de IP^o_{NM}.
Au niveau IPsec, dans le cas du mode Transport, la Mobilité nécessite de changer la SP et la SA. En effet, la SP doit être modifiée de sorte qu'un paquet utilisant IP^*_{NM} puisse être protégé exactement comme s'il l'était auparavant avec IP_{NM}, et qu'il puisse donc être associé à la même SA. La SA doit également être mise à jour pour que le trafic entrant puisse être traité. Tout d'abord

SPD (Trafic Sortant)	
Sélecteurs	Info Chiffrement
IP_{dst} : IP_{NC}	PROTECT
IP_{src} : IP_{NM}	Mode : Transport
P_{src} : ANY	Proto. : ESP
P_{dst} : ANY	Algo. : ENCR, AUTH
	SPI

SAD (Trafic Sortant)	
SPI : SPI	Crypto. : K_e, K_a
IP_{src} : IP_{NM}	CompteursC_{esp}, C_w
IP_{dst} : IP_{NC}	Sélecteurs SPD
	IP_{src} : IP_{NM}
	IP_{dst} : IP_{NC}
	P_{src} : ANY
	P_{dst} : ANY

(A) Transport ESP

SPD (Trafic Sortant)	
Sélecteurs	Info Chiffrement
IP_{dst} : IP_{NC}	PROTECT
IP_{src} : IP_{NM}	Mode : Tunnel
P_{dst} : ANY	IP^{Tunnel}_{src} : IP_{NM}
P_{src} : ANY	IP^{Tunnel}_{dst} : IP_{PS}
	Proto. : ESP
	Algo. : ENCR, AUTH
	SPI

SAD (Trafic Sortant)	
SPI : SPI	Crypto. : K_e, K_a
IP_{src} : IP^{o}_{NM}	CompteursC_{esp}, C_w
IP_{dst} : IP_{PS}	IP^{Tunnel}_{src} : IP^{o}_{NM}
	IP^{Tunnel}_{dst} : IP_{PS}
	Sélecteurs SPD
	IP_{src} : IP_{NM}
	IP_{dst} : IP_{NC}
	P_{src} : ANY
	P_{dst} : ANY

(B) Tunnel ESP

TABLE B.2: Configuration IPsec des SPD & SAD

l'index de la SA doit être mis à jour avec les bonnes adresses IP. Ensuite, la SA doit mettre à jour les champs correspondants aux Sélecteurs SPD, afin qu'une fois déchiffré, la couche IPsec puisse vérifier que le paquet déchiffré correspondait bien à la bonne SP.
Au niveau IPsec, dans le cas du mode Tunnel, la Mobilité ne modifie que les adresses externes. Cela nécessite de modifier la SA et éventuellement le SP. En effet, si l'on ne modifie pas la SP, pour le trafic sortant, les adresses internes n'étant pas modifiées, la bonne SP sera toujours sélectionnée après la mobilité. On peut éventuellement vouloir mettre à jour le contenu de la SP qui précise les adresses externes du mode Tunnel. Par contre, la SA nécessite d'être mise à jour. En effet, les adresses IP du Tunnel mentionnées dans la SA doivent être mises à jour afin de permettre la vérification de la SP une fois le paquet déchiffré. Il est également préférable de mettre à jour les adresses IP externes utilisées pour identifier la SA, même si le SPI peut suffire.
On notera que dans le cas du mode Tunnel, une Mobilité permet de rediriger une même connexion —entre IP_{NM} et IP_{NC} —depuis l'interface IP^{o}_{NM} puis depuis l'interface $IP^{o}_{NM}*$. Grâce au mode Tunnel, le changement de configuration permet, sans interrompre la communication, d'effectuer une opération de Mobilité. Ceci n'est, par exemple, pas le cas lorsque l'on utilise le mode Transport. En effet, avec le mode Transport, la mobilité de la communication ne peut être effectuée par IPsec. La Mobilité IPsec est nécessaire, mais doit être couplée avec un protocole permettant d'effectuer la mobilité de la communication. SCTP ou MPTCP effectue cette mobilité au niveau de la couche transport, mais elle peut également être effectuée par l'application avec des mécanismes de résilience.

Lorsque l'on parle de Multihoming, dans cette thèse, cela signifie que la couche IPsec du NM fournit au NC une liste d'adresse IP appelée adresses alternatives, qui peuvent être utilisées si l'adresse initiale ne permet plus de joindre le NM. Si l'adresse IP initiale IP_{NM} n'est plus joignable, alors le NC utilisera l'adresse alternative IP^{*}_{NM}. Cette opération est assimilable à une opération de mobilité, à la différence que la mise à jour peut être déclenchée non pas par le nœud sur qui le changement d'interface intervient, i.e. le NM, mais sur le NC.

Lorsque l'on parle d'Interfaces Multiples, dans cette thèse, cela signifie qu'un NM communique avec le NC sur une interface, notée IP_{NM} dans le cas d'une connexion en mode Transport et IP^{o}_{NM} dans le cas d'une connexion en mode Tunnel. Le NM s'attache à une seconde Interface, notée IP^{*}_{NM} en mode Transport et $IP^{o}_{NM}*$ en mode Tunnel. L'idée est de configurer IPsec de manière à ce que le NM puisse utiliser simultanément les deux interfaces pour la communication déjà établie.
Pour résoudre ce problème, nous tenons compte des implémentations IPsec existantes qui ne permettent pas, par exemple, d'associer plusieurs adresses IP à une SP ou une SA. Aussi, dans la cas du mode Transport, l'ajout d'une interface nécessite la création d'une nouvelle SP, avec les Sélecteurs associées à la nouvelle Interface. De même pour le trafic entrant, une nouvelle SA doit être créée, avec les sélecteur SPD ainsi que les adresses IP indexant la SA associée à la nouvelle Interface.
De manière similaire, pour le mode Tunnel, une nouvelles SA doit être créée et associée à la nouvelle interface, et comme une SPD pointe vers une unique SA, une nouvelle SPD doit être également créée. Ainsi, pour le mode Tunnel, deux SP avec les même trafic Sélecteurs vont pointer vers deux SA différentes. Ceci peut alors poser le problème de savoir qu'elle SP sélectionner lorsque le NM émet du trafic sur le réseau. La SPD est une base ordonnée, et la SP choisie sera celle du premier match avec le paquet. Si une seule SPD est disponible dans le système, une seule interface seulement pourra être choisie. Cette propriété peut être utilisée afin de faire un Soft Handover. En revanche, si l'on veut faire du load balancing, il faut séparer les SPD par interface.

B.4.3 MOBIKE-X

MOBIKE-X [Dan09b] est le protocole qui permet au NM d'effectuer au niveau de IPsec une opération de Mobilité, du Multihoming, ou d'ajouter une Interface. MOBIKE-X étend les fonctionnalité de MOBIKE l'extension de MOBilité et de Multihoming de IKEv2. MOBIKE a été défini pour les VPNs, la Mobilité ou le Multihoming ne permettent que de faire du Hard Handover. En effet MOBIKE suppose que le NM ne possède qu'une seule interface active à la fois. MOBIKE-X étend donc les fonctionnalités de MOBIKE-X et permet par la gestion de plusieurs interfaces de faire du Soft Handover pour les VPNs. De plus, MOBIKE-X étend également ses fonctionnalité au mode Transport.

Lorsque la connexion entre le NM (IP_{NM}) et le Serveur (IP_{SRV}) est tunnelée vers la *Passerelle de Sécurité*, MOBIKE [Ero06] permet d'effectuer un Hard Handover comme représenté à la figure B.9a. Pendant la phase d'initialisation de la communication (Phase 1), le NM et la *Passerelle de Sécurité*, annoncent le support du protocole MOBIKE via le Notify Payload MOBIKE_SUPPORTED. Les autres Payload mentionnées sur la figure permettent la négociation du matériel de sécurité et l'établissement de la SA ainsi que la IKE_SA, qui est le canal sécurisé permettant l'échange de messages IKEv2 entre le NM et le NC. Le NM s'attache à une autre réseau et utilise IP^{*}_{NM}. Il va alors avertir la *Passerelle de Sécurité*, qu'il utilise IP^{*}_{NM} à la place IP_{NM} par l'envoie d'un UPDATE_SA_ADDRESSES Notify Payload . Dans MOBIKE, ce message ne comprend aucune indication. En effet, MOBIKE suppose que le NM ne possède qu'une seule interface, donc la nouvelle adresse IP est indiquée dans le header IP du paquet. A la réception de ce paquet la *Passerelle de Sécurité* procède à la mise à jour de sa SAD, puis éventuellement procède à un test de joignabilité, le *Return Routability Check*. L'adresse IP indiquée dans le header IP n'est pas protégée par IPsec et donc peut être forgée. De plus rien n'indique que le NM est joignable à cette adresse IP. Afin de s'assurer que le NM est joignable et que cette nouvelle adresse appartient bine au NM, la *Passerelle de Sécurité* envoie un COOKIE2 Notify Payload.
Avec MOBIKE-X, la Mobilité Hard Handover peut être effectuée même pour le mode Transport. A la différence de MOBIKE, il faut négocier MOBIKE-X plutôt que MOBIKE. Pour cela on réutilise

le Notify Payload MOBIKE_SUPPORTED, mais on spécifie la version de MOBIKE, et on attribue à MOBIKE-X la version 2. Nous avons également spécifié que MOBIKE-X supporte les interfaces multiples. Si le NM possède par exemple deux interfaces, et que sur l'une des interfaces est utilisée par deux connections protégée par IPsec. Cette interface supporte alors également le canal IKEv2. Le NM peut vouloir bouger une connexion sur la seconde interface, sans que l'autre connexion ou le canal IKEv2 ne soit impacté. MOBIKE ne permet pas cela, car, par défaut tout le trafic utilise une unique interface. MOBIKE-X permet une telle opération grâce à des SELECTORs Notify Payloads. Ces Notify Payload permettent de cibler un trafic, et donc de restreindre la mobilité à un trafic spécifique.

La figure B.9b et la figure B.9c mettent en évidence la gestions des Interfaces Multiples par MOBIKE-X. En effet MOBIKE-X permet d'ajouter et de retirer une adresse IP à une ensemble d'associations de Sécurité. Comme précisé précédemment, MOBIKE-X permet de sélectionner l'ensemble des SA et SPD concernés par l'ajout ou le retrait d'une interface. Ainsi, le NM envoie en Phase 1 de la figure B.9b un ADD_SA_ADDRESS Notify Payload. Rappelons que lorsque le NM ajoute une adresse IP, une nouvelle SA est crée. Lorsque le NM envoie un tel message il doit alors impérativement spécifier la valeur des SPIs des nouvelles SA. Les SA sont unidirectionnelles, donc une communication comprend deux SAs, et chaque direction possède son propre SPI. Ce qui importe, c'est que le SPI du trafic entrant soit unique.

```
Initiateur (NM)                    Respondeur (PS)
-------------------------------------------------
    <<< IPsec est non configuré >>>

1) Mise en place du canal IKEv2
SA1_EU , KE_NM , N_NM  -->
                        <--  SA1_PS , KE_{PS} , N_{PS}
SK {ID_NM , AUTH,
SA2_NM , TS_NM , TS_PS ,
CP(CFG_REQUEST) ,
N(MOBIKE_SUPPORTED)} -->
                <-- SK {ID_PS , AUTH,
                        SA2_PS , TS_NM , TS_PS ,
                        CP(CFG_REPLY) ,
                        N(MOBIKE_SUPPORTED)}

    <<< Canal IKEv2 établi
    TS_NM,SRV Tunnelé dans IP_NM , IP_PS >>>

2) Mobilité Hard Handover IP*_NM
SK {N(UPDATE_SA_ADDRESSES)} -->
                        <-- SK {N()}

    <<< EU utilise IP*_NM , TS_NM,PS non modifiées
    TS_EU,PS Tunnel\'{e} in IP*_NM , IP_PS >>>

3) Return Routability Check
                        <-- SK {N(COOKIE2)}
SK {N(COOKIE2)}  -->
```

(A) Hard Handover

```
Initiateur (NM)                    Respondeur (PS)
-------------------------------------------------
    <<< Canal IKEv2 établi
    TS_EU,PS Tunneled in IP_NM , IP_PS >>>

1) NM acquière IP*_NM
SK {N(ADD_SA_ADDRESS IP*_NM)} -->
                        <-- SK {N()}

    Return Routability Check
                        <-- SK {N(COOKIE2)}
SK {N(COOKIE2)}  -->

    <<< NM possè plusieurs interfaces IP*_NM , IP_NM
    IP_NM est l'interface primaire
    TS_NM,PS Tunnelé dans IP*_NM , IP_PS >>>

    <<< Après T1 secondes >>>

2) NM met la préférence pour IP*_NM
    à la fois pour TS_NM,PS , et IKE
SK {SET_PRIMARY} -->
                        <-- SK {N()}

    <<< EU reçoit le trafic sur IP*_NM .
    TS_NM,PS Tunnelé et non modifiées
    NM IP_NM est toujours active >>>

    <<< Après T2 secondes >>>

3) Remplacement de IP_NM
SK {N(REMOVE_SA_ADDRESSES IP_NM)} -->
                        <-- SK {N()}

    <<< EU utilise seulement IP*_NM >>>
```

(B) Soft Handover

```
Initiateur (NM)                    Respondeur (PS)
-------------------------------------------------
    <<< Canal IKEv2 établi
    TS_EU,PS Tunnelé dans IP_NM , IP_PS >>>

1) NM acquière IP*_NM
SK {N(ADD_SA_ADDRESS IP*_NM ) , -->
     N(SOFT_HANDOVER, T1, T2, IP*_NM , IP_NM )

    <<< NM possède plusieurs interfaces IP*_NM , IP_NM
    IP_NM est l'interface primaire
    TS_NM,PS Tunnelé dans IP*_NM , IP_PS >>>

    <<<  Après T1 secondes >>>

2) NM Sets Preference for IP*_NM
    For both TS_NM,PS , and IKE

    <<< EU receives traffic on IP*_NM .
    TS_NM,PS Tunnelé et non modifiées
    NM IP_NM est toujours active >>>

    <<< Après T2 secondes >>>

3) Remplacement IP_NM
                        <-- SK {N()}

    Return Routability Check
                        <-- SK {N(COOKIE2)}
SK {N(COOKIE2)}  -->

    <<< EU utilise seulement IP*_NM >>>
```

(C) SOFT_HANDOVER Notify Payload

FIGURE B.9: Description des échanges de mobilité avec MOBIKE(-X)

B.4.4 Étude des performance d'IPsec dans un environnement Mobile

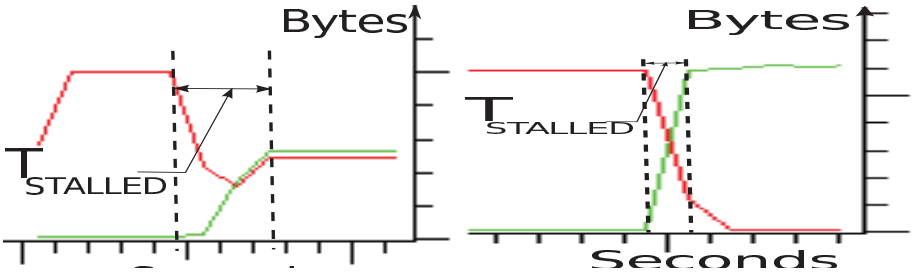
Ainsi MOBIKE-X étend les fonctionnalités de MOBIKE, et dans cette section, nous souhaitons évaluer dans quelle mesure cette extension améliore la Qualité de Service des utilisateurs.

La figure B.10 présente les conditions de tests et les résultats que nous obtenons. La figure B.10a évalue comment la communication est interrompue dans la cas d'une mobilité avec MOBIKE. On se place alors dans le cas d'une mobilité de VPN, qui utilise donc le mode Tunnel d'IPsec. La plateforme expérimentale et les mesures se font sur Ethernet. On utilise Wireshark afin de visualiser et mesurer les temps qui nous permettent de comparer MOBIKE et MOBIKE-X, et en particulier $T_{STALLED}$ le temps d'interruption de la communication. La figure B.10b mesure l'impacte d'une mobilité sur une communication protégée par le mode Transport, et où la mobilité est gérée par MOBIKE-X.
MOBIKE-X utilisé avec le mode Transport, nécessite, lorsqu'une opération de mobilité a lieu, de mettre à jour à la fois la SAD et la SPD. MOBIKE, en revanche ne nécessite de mettre à jour que sa SPD. Ainsi, le temps des mises à jour des bases IPsec est quasiment doublé dans le cas du mode Transport, par rapport au mode Tunnel. La mise à jour des bases de données IPsec met environ $36.6035\,ms$ en mode Transport, ce qui est 2.69 plus important que dans le cas du mode Tunnel. Durant la mise à jour, les bases de donnée IPsec sont bloquées, et donc la communication est interrompue. Afin d'estimer si ce temps est susceptible d'affecter la communication, on le compare au Round Trip Time (RTT) qui définit la latence du réseau.
Dans la cas d'un réseau Ethernet, on mesure $RTT = 0.355\,ms$, dans le cas d'une réseau WLAN connecté à une connexion ADSL de 1M, on mesure $RTT = 9.466\,ms$, dans le cas d'une connexion ADSL à 10M on mesure $RTT = 2.14\,ms$, et, dans le cas d'un HotSpot Public, on mesure $RTT = 15\,ms$. La comparaison du temps de mise à jour par rapport au RTT montre que cette dernière correspond à $2RTT$ dans le cas d'une mobilité entre les points d'accès public, et ne devrait donc pas impacter la Qualité de la Communication.
Si les mises à jour, dans le cas du mode Transport, sont 2.65 fois plus longues que dans le cas du mode Tunnel, en revanche, l'utilisation du mode Transport simplifie considérablement les opérations réseau, et permet au système d'être beaucoup plus réactif. En effet, plus le système comporte de couches successives, plus les modifications ou les changements prennent du temps avant d'être effectives. On peut ainsi voir sur la figure B.10 que le temps d'interruption de la communication est supérieur avec le mode mode Tunnel qu'avec le mode Transport. Plus exactement, $T_{STALLED} \approx 264\,ms$ est entre 9.3% et 15.6% plus rapide avec MOBIKE-X qu'avec MOBIKE. En fait, avec le noyau Linux, le mode Tunnel impose les paquets de repasser deux fois dans la couche IP, ce qui complique ou ajoute des opérations réseau supplémentaires. L'impact de la complexité des opérations réseaux sur les opérations de mobilité est encore plus flagrant lorsque l'on compare la mobilité effectuée par SCTP sur une communication non protégée par IPsec, avec une communication protégée par IPsec. Pour une communication non protégée par IPsec, la Mobilité SCTP prend $30\,ms$, alors que si les liens sont protégés avec IPsec, elle prend $200\,ms$.

Ainsi MOBIKE-X permet de réduire l'impact d'une Mobilité Hard Handover de deux façons. MOBIKE-X permet l'emploi du mode Transport qui réduit la complexité réseau, et donc augmente la réactivité du système. Par ailleurs MOBIKE-X permet le Soft Handover, pour une communication protégée à la fois par le mode Transport et par le mode Tunnel. Avec la réception simultanée sur deux interfaces, le Soft Handover diminue la perte de paquets au moment de la mobilité. Ensuite, pour le mode Transport, avec lequel IPsec doit être couplé à un autre mécanisme de mobilité, le Soft Handover permet de configurer IPsec en avance, et ainsi s'abstraire de la configuration IPsec au moment de la mobilité.

B.4.5 Architecture d'offload

Dans cette section, on se concentre sur la problématique de l'offload, et on montre comment MOBIKE-X pourrait être utilisé. On parle d'offload lorsque l'ISP redirige le trafic d'un utilisateur sur des réseaux WLAN afin de réduire la charge du réseau d'Accès Radio. A la différence du cas où un utilisateur choisit d'utiliser un réseau WiFi afin d'éviter par exemple de consommer son

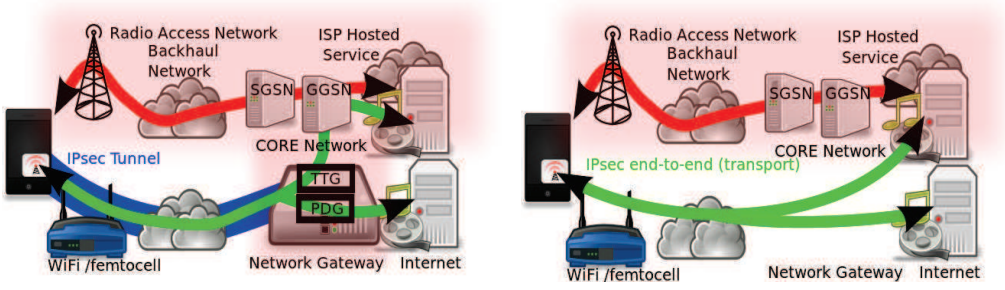


(A) *MOBIKE* (B) *MOBIKE – X*

FIGURE B.10: Comparaison d'une Mobilité Hard Handover en mode Tunnel (MOBIKE) et en mode Transport (MOBIKE-X)

forfait data 3G, dans le cas de l'offload, la décision est prise par l'opérateur. Cela signifie que l'ISP doit fournir une architecture sécurisée et dimensionnée de manière à ce que la qualité de service de l'utilisateur ne soit pas impactée. Pour ce faire, on détaille dans la figure B.11 les deux types d'architectures que nous considérons:

- **OAA:** Offload Access Architecture. Cette architecture fait intervenir une *Passerelle de Sécurité*. Son principe est de tunneler le trafic de l'utilisateur jusqu'à un point d'entrée d'un réseau de confiance. L'intérêt d'une telle architecture est qu'elle est indépendante du service. L'inconvénient est qu'elle ne prend pas en compte les spécificités du service, introduit des indirections, et utilise le mode tunnel qui rajoute du traitement et des overheads. Ce type d'architecture est similaire à celle définie par IWLAN [3GP11].
- **OSA:** Offload Service Architecture. Cette architecture utilise le mode Transport et est véritablement le pendant IPsec d'une communication IPsec. L'avantage est qu'elle est très spécifique au service, qu'elle évite les indirections et les mécanismes de tunnels.



(A) OAA (I-WLAN) (B) OSA

FIGURE B.11: Description des architecture OAA et OSA

Les architectures OSA et OAA sont de natures différentes, et la spécificité de OSA pour un service revient à poser une première question: *Quel service nécessite une architecture OSA spécifique et ne peut se contenter d'une architecture OAA?*. De tels services nécessitent une qualité de service qui souffrirait des indirections dues à une *Passerelle de Sécurité*, et dont l'overhead du tunnel augmenterait la latence. Nécessairement ces services ont une composante temps réel importante, et / ou utilisent des paquets de petite taille. En effet, pour un paquet dont le MTU est de 1500 octet, l'overhead du tunnel est négligeable et est inférieur à 3%. Ainsi, les services cibles, sont ceux

de type Voix ou de jeux où l'aspect temps réel et la confidentialité des données sont primordiaux. Inversement, on peut également se poser la question: *Quels sont les services qui nécessitent une architecture d'offload sécurisée?*. En effet, si un service ne nécessite pas une architecture spécifique, et peut se contenter d'une architecture OAA, le déploiement de ces architectures restent couteux, et une architecture OAA continue de charger le réseau CORE de l'opérateur. Ainsi, on peut définir le trafic qui nécessite une architecture OSA, celui qui ne nécessite aucune sécurité, et par défaut, le reste du trafic utilisera une architecture OAA en situation d'offload. Il est difficile de caractériser le trafic qui nécessite d'être protégé, de celui qui ne le nécessite pas, mais on peut supposer que les services protégés par HTTPS par exemple, ne nécessitent pas de protection supplémentaire. Une protection via OAA masquerait l'utilisation d'un service par un individu. D'autres trafics comme du trafic vidéo *youtube*, constitue une part importante du trafic mobile, et peut également être jugé comme ne nécessitant pas d'être protégé. Le trafic non protégé est directement traité par le réseau d'accès et est pris en charge par l'architecture ForWaRD Architecture (FWDA). Cette différentiation du trafic à laquelle on attribue une architecture d'offload différente est illustrée par la figure B.12.

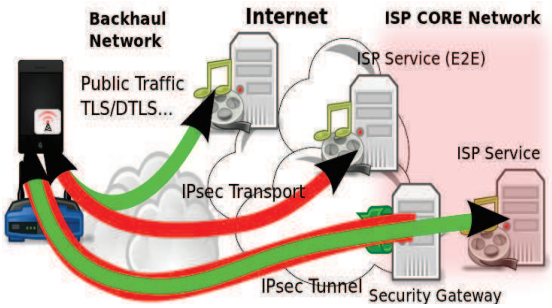


FIGURE B.12: Stratégie d'Offload pour ISPs combinant FWDA, OSA and OAA

Une fois l'architecture d'offload décrite, il reste dans le cas de l'architecture d'OSA, à voir comment déployer des applications adaptées à cette architecture. En effet, OSA n'utilise pas le mode Tunnel, mais le mode Transport. Cela signifie aussi que la mobilité doit être gérée pour le flux, parallèlement à IPsec. Cela peut se faire au niveau applicatif, c'est à dire par l'application, ou au niveau de la couche Transport. Dans cette thèse nous avons étudié le cas où la mobilité s'effectuait au niveau Transport, et nous avons utilisé, pour cela le protocole SCTP. En effet, si l'opérateur choisit de faire bénéficier une application de la mobilité au niveau transport, SCTP permet un portage rapide, soit par l'intermédiaire de librairie, soit par l'implémentation de SCTP au niveau du noyau. Par exemple, les applications TCP traditionnelles peuvent être portées sur SCTP en utilisant des primitives du style *withsctp*. Toutefois, aussi petite soit la modification, porter une application sur SCTP nécessite une validation, des testes fonctionnels, des testes de charges, ce qui représente un coût. Ainsi l'offload des applications critiques pour les opérateurs doit tenir compte de l'infrastructure à déployer ainsi que du portage de l'application.

B.4.6 Discussion

Dans cette partie, nous avons généralisé le protocole MOBIKE afin de permettre à des communications utilisant le mode Transport de bénéficie de la Mobilité, du Multihoming. Enfin, MOBIKE-X permet également la gestion d'Interfaces Multiples, et permet de gérer le Soft Handover. Nous nous sommes concentrés sur la problématique d'offload afin d'illustrer l'utilisation du protocole

MOBIKE-X. Les aspects Soft Handover semblent les plus prometteurs, et l'extension au mode Transport pourrait être utilisée pour des services critiques qui nécessitent à la fois une haute protection des informations ainsi que des besoins en temps réel. Les services qui pourraient bénéficier de ces fonctionnalités sont vraisemblablement les services voix et jeux. Toutefois, les problèmes auxquels on se heurte sont: 1) Le faible déploiement d'IPsec, 2) IPsec représente une solution "Opérateur", qui ne sera vraisemblablement pas privilégiée par les fournisseurs d'applications.

B.5 Conclusion

Dans cette thèse nous avons à la fois exploré comment migrer les plateforme de résolutions DNS vers DNSSEC. Comparées aux plateformes déployées, les architectures proposées permettent de réduire au moins par 5 le nombre de nœuds. De plus les architectures proposées sont à base de DHT et facilitent le management de la plateforme grâce aux fonctionnalités d'auto-configuration. Dans cette thèse nous nous sommes concentrés sur le protocole DNSSEC, mais les principes peuvent également être repris pour d'autres services. En particulier, l'opérateur peut appliquer les méthodes à la distribution de contenu de manière générale. Pour notre part, le travail effectué dans cette thèse a été transmis aux équipes opérationnelles qui peuvent tirer parti des résultats. Outre les résultats de l'étude, nous avons développé un certain nombre d'outils qui permettent de simuler des architectures à partir d'un trafic réel et d'analyser le trafic DNS(SEC). Ces outils ont également apporté une valeur ajoutée car d'autres études les réutilisent. Il est prévu de les publier en open source, et de leur fournir une interface graphique.
Si la mise à disposition des outils constitue l'étape immédiate après avoir rendu le manuscrit, d'autres études restent a explorer pour optimiser la plateforme de résolutions. Ainsi, on pourra effectuer une étude sur la qualité et les ressources demandées par les fonctions de hachage. En effet, dans les architectures proposées, nous avons principalement considérer SHA1 comme la fonction de hachage par défaut. SHA1 est une fonction de hachage cryptographique surdimensionnée par rapport à nos besoins. Il reste donc à définir une fonction de hachage adaptée à nos besoin spécifiques qui optimise la répartition du trafic et minimise le calcul. Par exemple, la comparaison des répartitions utilisant, sur les adresses IP, une fonction XOR et un SHA1, montre que l'on obtient une répartition équivalente du trafic. Il reste à mesurer "l'uniformité" des fonctions de hachage et implémenter une fonction de hachage optimale. Une autre perspective est de définr et normaliser, à partir de Pastry, le protocole DHT, nécessaire à la plateforme. Plus particulièrement, nos besoins permettent de simplifier Pastry. Le design doit également soigneusement prendre en compte les aspects auto-configurations. Une implémentation permettrait de mesurer les limites de l'architecture en particulier, le nombre optimal de FQDNs à considérer comme populaires.

Ensuite, nous avons décrit, implémenté et testé un protocole qui permet la continuité d'une communication protégée par IPsec dans une environnement Mobile, Multihomé et avec des Interfaces Multiples. La faisabilité a été prouvée au cours de cette thèse, et les avantgaes et inconvénients entre les différentes configurations ont été identifiés. Toutefois l'utilisation d'IPsec, la mise en place d'une architecture dédiée à des services, la modification des applications en situation d'offload sont des choix qui doivent mesurer les gains financiers avec les coûts de déploiement. Cette thèse achève la partie R&D. La suite immédiate de cette thèse considère le développement et la standardisation de MOBIKE-X à l'IETF. En particulier des drafts sur le Soft Handover doivent être publiés, ainsi que sur la gestion des SPIs. Nous souhaitons également développer un client VPN optimisé pour la mobilité et le distribuer à la fois pour les plateformes Androids, Windows, et Linux.
Aux travaux immédiats, nous pouvons également distinguer d'autre études plus conséquentes. L'une consisterait à faire ce qui a été fait avec SCTP avec le protocole plus actuel MPCTP. Il s'agirait de faire fonctionner MPTCP et MOBIKE-X ensemble. Ensuite, il serait également souhaitable de voir comment MPTCP ou MOBIKE-X pourraient réduire leurs signalisations re-

spectives. En effet, lors d'un changement d'interface, par exemple, MOBIKE-X et MPTCP communique cette information, il est vraisemblable que l'établissement d'un canal de communication entre MOBIKE-X et MPTCP optimiserait les opérations de mobilité, multihoming et d'interfaces multiples. Ces tests doivent également faire l'objet de comparaison avec HIP. IPsec a ses avantages, mais il serait également avantageux de comparer les performances d'une communication sécurisé avec TLS / DTLS. Cela inclut une comparaison entre IPsec et TLS/ DTLS à la fois de manière statique et pour les opérations de mobilité multihoming, et interfaces multiples. Dans cette thèse, nous nous sommes efforcés, avec SCTP de gérer la mobilité avec SCTP, i.e. au niveau transport. Il est alors intéressant de positionner une mobilité au niveau transport par rapport à des mécanismes de type "session resumption" qui peuvent être gérés par l'application.

Accomplished Work

Publications

This thesis results in a few publications and we are currently working on publishing the code for the opensource community. Publications include:

- [MGL10] D. Migault, C. Girard, and M. Laurent. A performance view on dnssec migration. In *CNSM 2010*, pages 469–474, oct 2010 presents the DNS to DNSSEC migration problematic, as well as a DNS and DNSSEC performance comparison.
- [Mig10] D. Migault. Performance Measurements on BIND9/NSD/UNBOUND. In *IETF79/IEPG*. IEPG, november 2010 presents the DNS to DNSSEC during an open session at the IEPG 79, before the IETF. Results are provided from [MGL10].
- [XMSF11] Q. Xu, D. Migault, S. Sénécal, and S. Francfort. K-means and adaptive k-means algorithms for clustering dns traffic. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11, pages 281–290, ICST, Brussels, Belgium, Belgium, may 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) presents the use of K-mean for an analysis of the FQDNs.
- [FMS11, FMS12] S. Francfort, D. Migault, and S. Sénécal. A bi-objective mixed integer linear program for load balancing dns(sec) requests. In *Global Annual Symposium on DNS-SSR the DNS-EASY*, DNS-EASY 2011. ACM, october 2011 and S. Francfort, D. Migault, and S. Sénécal. A bi-objective mixed integer linear program for load balancing dns(sec) requests. In *International Journal of Critical Infrastructure Protection*. Elsevier, 2012 presents a description of the use of Mixed Integer Linear Program (MIP) for load balancing the DNS load.
- [MHS+a] D. Migault, E. Herbert, S. F. Stanislas, S. Sénécal, and M. Laurent. "analyzing traffic and building routing tables for increasing dns(sec) resolving platforms efficiency (under submission)" presents how to distribute properly the traffic between the nodes of the platform. A K-mean analyses of FQDNs is followed by different algorithms based on Mixed Integer Linear Program (MIP) and others.
- [ML11] D. Migault and M. Laurent. How dnssec resolution platforms benefit from load balancing traffic according to fully qualified domain name. In *ICSNA International Conference on Secure networking and Applications (ICSNA)*, october 2011 presents the DNSSEC migration costs for an ISP as well as the benefits provided by a FQDN load balancer strategies.
- [MHS+b] D. Migault, E. Herbert, S. F. Stanislas, S. Sénécal, and M. Laurent. "overcoming dnssec performance issues with fqdn load balancer and cache sharing (under submission)"

presents the various DHT architectures as an alternate way of FQDN load balancer. This paper also shows the benefits of the pro-active caching architecture.

- [JMDJ+07, JMDJ+09] C. Jean-Michel, M. Daniel, B. Julien, C. Hakima, and L.-M. Maryline. Sécurité des réseaux mobiles ip. In *Traité IC2 (Hermès) "Sécurité des réseaux sans fil et mobiles"*. Lavoisier, 2007 and
C. Jean-Michel, M. Daniel, B. Julien, C. Hakima, and L.-M. Maryline. Security of ip-based mobile networks. In *Wireless and Mobile Network Security*, ISTE (International Society for Technology in Education), London, UK, 2009. Wilet presents the Security of the Mobility Protocols.
- [MPH+12b] D. Migault, D. Palomares, E. Herbert, W. You, G. Ganne, G. Arfaoui, and M. Laurent. E2e: An optimized ipsec architecture for secure and fast offload. In *International Workshop on Security of Mobile Applications IWSMA'12 (co-located with the ARES'12)*, august 2012 presents the measurement performances of MOBIKE-X in conjunction with SCTP. It also compares MOBIKE-X and MOBIKE.
- [MPH+12a] D. Migault, D. Palomares, E. Herbert, W. You, G. G. G. Arfaoui, and M. Laurent. Isp offload infrastructure to minimize cost and time deployment. In *Proc. of IEEE Global Telecommunications Conference - Communication and Information System Security (GLOBECOM '12)*, december 2012 presents deployable offload solutions based on MOBIKE, MOBIKE-X and SCTP + MOBIKE-X. This paper is intended to minimize the cost of offload.
- [DMM+09] M. Daniel, L. Maryline, N. M.Tran, K. Brigitte, A. Achour, S. Nuyen, N. Abid, N. Boukatem, N. Tran, F. Mirani, and B. Eric. 3ming: Mobility multi-technologie multi-homing: Wp1 - d1.3. French National Research Association, TELECOM, ANR-07-TLCOM-01, march 2009 is a State Fund Research project where we first presented MOBIKE-X.
- [MPL] D. Migault, D. Palomares, and M. Laurent. MOBIKE-X to overcome Simultaneous Support of Security, Mobility, Multihoming and Multiple Interfaces presents MOBIKE-X as well as its performances.
- [Dan09b, Dan09a, DC12] M. Daniel. MOBIKE eXtension (MOBIKE-X) for Transport Mobility and Multihomed IKE_SA. draft. (Work in Progress) Internet Engineering Task Force, september 2009, M. Daniel. IPsec mobility and multihoming requirements : Problem statement. draft. (Work in Progress) Internet Engineering Task Force, september 2009 and M. Daniel and W. Carl. Multiple Interfaces IPsec Security Requirements. draft. (Work in Progress) Internet Engineering Task Force, july 2012 are contribution for MOBIKE-X at the IETF.

Pieces of Software

Source code includes:

- DHT simulator: Written in C, this simulator that evaluates the CPU consumption of a DNS(SEC) platform composed of multiple nodes. The Simulator also computes the Response Time perceived by the End User. This simulator takes as input distribution for the FQDN popularity. The distribution can be provided with a mathematical expression or with a list. From this distribution, the Simulator applies probabilistic model to simulate different architecture. The considered architectures are:
    - o IP: where traffic is split between the nodes according tothe IP addresses.
    - o FQDN: where traffic is split between the nodes according to the FQDN.
    - o Pastry: where the platform architecture is based on Pastry. Note that we use a configuration for Pastry that do not cache responses.
    - o Pastry-Straight-Forward: which is an enhancement of Pastry, that makes the Respon-

sible Node sends the response directly to the End User rather then to the Pastry Node requesting on behalf of the End User.

  - o Pastry-Passive-Caching: which evaluates how caching all queries and responses affects the node. Pastry-Pro-Active-Caching: which evaluates how pro-active caching affects the performances of the platform.

- DNS traffic simulator and traffic analyser. PCAP Analyzer and Simulator written in python. This script takes a pcap file as input, runs tshark to gather information on FQDN, IP addresses, or TTL and fill its dictionaries. Once dictionaries are build, Python routines plot statistics such as distribution of certain network parameters. Python routines also compute traffic repartition among the nodes of a platform according to different rules. This is the simulation part.
- MOBIKE-X implementation. This implementation is based on strongSwan 4.3.
- IPsec performance measurements tools. These tools are provided to proceed to performance tests over various configurations. The tools enables measurements and to figure the results.
- SCTP platform guide lines, with multiple how-to written on a wiki, as well as scripts used to perform mobility tests.

Bibliography

[3GP11] 3GPP-LTE: 3GPP system to Wireless Local Area Network (WLAN) interworking; System description, TS 23.234, Release 10. ETSI Standard, march 2011.

[AAL+05a] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. URL: http://www.ietf.org/rfc/rfc4033.txt, march 2005. RFC 4033 (Proposed Standard), Updated by RFC 6014.

[AAL+05b] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. URL: http://www.ietf.org/rfc/rfc4035.txt, march 2005. RFC 4035 (Proposed Standard), Updated by RFCs 4470, 6014.

[AAL+05c] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. URL: http://www.ietf.org/rfc/rfc4034.txt, march 2005. RFC 4034 (Proposed Standard), Updated by RFCs 4470, 6014.

[ABV+04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). URL: http://www.ietf.org/rfc/rfc3748.txt, june 2004. RFC 3748 (Proposed Standard), Updated by RFC 5247.

[ADF05] B. Ager, H. Dreger, and A. Feldmann. Exploring the Overhead of DNSSEC. URL: http://www.net.in.tum.de/~anja/feldmann/papers/dnssec05.pdf, 2005.

[Age05] B. Ager. Performance Evaluation of DNSSEC. URL: http://www.net.t-labs.tu-berlin.de/papers/A-PEDNSSEC-05.pdf, march 2005.

[AH06] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). URL: http://www.ietf.org/rfc/rfc4187.txt, january 2006. RFC 4187 (Informational), Updated by RFC 5448.

[Ait09] R. Aitchison. Choosing a DNSSEC Solution. URL: http://www.zytrax.com/books/dns/info/choosing-dnssec-solution.pdf, may 2009.

[AKB07] R. Arends, M. Kosters, and D. Blacka. DNS Security (DNSSEC) Opt-In. URL: http://www.ietf.org/rfc/rfc4956.txt, july 2007. RFC 4956 (Experimental).

[alt03] Nortel Networks Alteon OS 21.0, Alteon Application Switch, september 2003.

[And] Android 5 VPN doAndroids. URL: https://play.google.com/store/apps/details?id=com.doenter.android.vpn.fivevpn&hl=en.

[App] Apple iOS: Setting up VPN. URL: http://support.apple.com/kb/HT1424.

[APW] APWG Anti Phishing Working Group (APWG). URL: http://www.antiphishing.org.

[ATT] AT&T Corporation originally American Telephone and Telegraph Company. URL: http://www.att.com/.

[AW06] M. Andrews and S. Weiler. The DNSSEC Lookaside Validation (DLV) DNS Resource Record. URL: http://www.ietf.org/rfc/rfc4431.txt, february 2006. RFC 4431 (Informational).

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications . In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126 –134 vol.1, march 1999.

[BIKS03] S. Bellovin, J. Ioannidis, A. Keromytis, and R. Stewart. On the Use of Stream Control Transmission Protocol (SCTP) with IPsec. URL: http://www.ietf.org/rfc/rfc3554.txt, july 2003. RFC 3554 (Proposed Standard).

[BIN] BIND. URL: http://www.isc.org/bind10.

[Boi] Boingo: The Worldwide Leader in Wi-Fi Software and Services. URL: http://www.boingo.com/.

[BP08] R. Bellis and L. Phifer. DNSSEC Impact on Broadband Routers and Firewalls. Nominet, september 2008.

[cav] cavium. URL: http://www.cavium.com/Table.html.

[CB03] J. L. Cooke and D. Bryson. Strong Cryptography in the Linux Kernel, Discussion of the past, present, and future of strong cryptography in the Linux kernel, july 2003.

[CCC07] E.-C. Cha, H.-K. Choi, and S.-J. Cho. Evaluation of Security Protocols for the Session Initiation Protocol. In *ICCCN'07*, pages 611–616, 2007.

[Cis11] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015. URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, february 2011.

[Cle08] A. Clegg. DNSSEC in 6 minutes. URL: http://www.isc.org/files/DNSSEC_in_6_minutes.pdf, 2008.

[CLW+08] J. Cao, M. Li, C. Weng, Y. Xiang, X. Wang, H. Tang, F. Hong, H. Liu, and Y. Wang, editors. *IFIP International Conference on Network and Parallel Computing, NPC 2008, Shanghai, China, October 18-21, 2008, Workshop Proceedings*, 2008.

[CMM02] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS Using a Peer-to-Peer Lookup Service. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 155–165, London, UK, 2002. Springer-Verlag.

[col] Collect for Linux. URL: http://collectl.sourceforge.net.

[Com] DNSSEC Information Center. URL: http://www.dnssec.comcast.net.

[Con01] D. Conrad. Indicating Resolver Support of DNSSEC. URL: http://www.ietf.org/rfc/rfc3225.txt, december 2001. RFC 3225 (Proposed Standard), Updated by RFCs 4033, 4034, 4035.

[CR06] R. Chandramouli and S. Rose. Secure Domain Name System (DNS) Deployment Guide, may 2006.

[CS02] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 177–190, New York, NY, USA, 2002. ACM.

[Dan09a] M. Daniel. IPsec mobility and multihoming requirements : Problem statement. draft. (Work in Progress) Internet Engineering Task Force, september 2009.

[Dan09b] M. Daniel. MOBIKE eXtension (MOBIKE-X) for Transport Mobility and Multihomed IKE_SA. draft. (Work in Progress) Internet Engineering Task Force, september 2009.

[DBI03] S. M. Das, R. V. Belani, and M. Imhasly. CacheMakers : A Co-operative DNS Caching Service. In *WWW (Posters)*, 2003.

[DC12] M. Daniel and W. Carl. Multiple Interfaces IPsec Security Requirements. draft. (Work in Progress) Internet Engineering Task Force, july 2012.

[DKK+01] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 202–215, New York, NY, USA, 2001. ACM.

[DMM+09] M. Daniel, L. Maryline, N. M.Tran, K. Brigitte, A. Achour, S. Nuyen, N. Abid, N. Boukatem, N. Tran, F. Mirani, and B. Eric. 3ming: Mobility multi-technologie multi-homing: Wp1 - d1.3. French National Research Association, TELECOM, ANR-07-TLCOM-01, march 2009.

[dnsa] DNSPerf, ResPerf, DHCPerf. URL: http://www.nominum.com/services/measurement_tools.php.

[dnsb] DNSSEC Deployment. URL: http://www.dnssec-deployment.org/wiki/index.php/Tools_and_Resources.

[dnsc] DNSSEC.NET. URL: http://www.dnssec.net/.

[dns08] DNSSEC Walker. URL: https://www.dns-oarc.net/tools/dnssecwalker, january 2008.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. URL: http://www.ietf.org/rfc/rfc5246.txt, august 2008. RFC 5246 (Proposed Standard), Updated by RFCs 5746, 5878, 6176.

[Eas99] D. Eastlake 3rd. Domain Name System Security Extensions. URL: http://www.ietf.org/rfc/rfc2535.txt, march 1999. RFC 2535 (Proposed Standard), Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845.

[Eas00] D. Eastlake 3rd. DNS Request and Transaction Signatures (SIG(0)s). URL: http://www.ietf.org/rfc/rfc2931.txt, september 2000. RFC 2931 (Proposed Standard).

[EK97] D. Eastlake 3rd and C. Kaufman. Domain Name System Security Extensions. URL: http://www.ietf.org/rfc/rfc2065.txt, january 1997. RFC 2065 (Proposed Standard), Obsoleted by RFC 2535.

[end] endace. URL: http://www.endace.com.

[Ero06] P. Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). URL: http://www.ietf.org/rfc/rfc4555.txt, june 2006. RFC 4555 (Proposed Standard).

[ETS10] P. Eronen, H. Tschofenig, and Y. Sheffer. An Extension for EAP-Only Authentication in IKEv2. URL: http://www.ietf.org/rfc/rfc5998.txt, september 2010. RFC 5998 (Proposed Standard).

[Fal00] P. Faltstrom. E.164 number and DNS. URL: http://www.ietf.org/rfc/rfc2916.txt, september 2000. RFC 2916 (Proposed Standard), Obsoleted by RFC 3761.

[Fei73] H. Feistel. Cryptography and computer privacy. *Scientific american*, 228(5):15–23, 1973.

[Fit11] K. Fitchard. TIA 2011: Verizon to offload 3G/4G data through free Wi-Fi hotspots. URL: http://connectedplanetonline.com/3g4g/news/tia-2011-verizon-wireless-to-offload-3g-4g-data\-through-free-wi-fi-hotspots-0519, may 2011.

[FM04] P. Faltstrom and M. Mealling. The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM). URL: http://www.ietf.org/rfc/rfc3761.txt, april 2004. RFC 3761 (Proposed Standard), Obsoleted by RFCs 6116, 6117.

[FMS11] S. Francfort, D. Migault, and S. Sénécal. A bi-objective mixed integer linear program for load balancing dns(sec) requests. In *Global Annual Symposium on DNS-SSR the DNS-EASY*, DNS-EASY 2011. ACM, october 2011.

[FMS12] S. Francfort, D. Migault, and S. Sénécal. A bi-objective mixed integer linear program for load balancing dns(sec) requests. In *International Journal of Critical Infrastructure Protection*. Elsevier, 2012.

[FN10] F5-Networks. F5, Networks: BIG-IP Global Traffic Manager: Implementation, march 2010.

[Fre] Free Pastry. URL: http://www.freepastry.org/.

[FRH+11] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. URL: http://www.ietf.org/rfc/rfc6182.txt, march 2011. RFC 6182 (Informational).

[FS00] N. Ferguson and B. Schneier. A Cryptographic Evaluation of IPsec. Technical report, Counterpane Internet Security, Inc, 2000.

[Gal09] J. Galvin. .org and DNSSEC: Securing the DNS. URL: http://www.isoc.org/isoc/conferences/dnspanel, july 2009.

[Gar08] G. L. Garcia. IPSec performance analysis for large-scale Radio Access. In *Helsinki University of Technology, Master Thesis*, july 2008.

[GKD07] G. Giaretta, J. Kempf, and V. Devarapalli. Mobile IPv6 Bootstrapping in Split Scenario. URL: http://www.ietf.org/rfc/rfc5026.txt, october 2007. RFC 5026 (Proposed Standard).

[GKLN08] A. Gurtov, D. Korzun, A. Lukyanenko, and P. Nikander. Hi3: An efficient and secure networking architecture for mobile hosts. *Comput. Commun.*, 31(10):2457–2467, 2008.

[Gra10] M. Graff. DNSSEC is coming. Is your organization ready? URL: `http://www.isc.org/blogs/mgraff`, may 2010.

[Gri09] C. Griffiths. Comcast DNSSEC Trail Test Bed, january 2009.

[Gri10] C. Griffiths. Comcast Voices : DNSSEC. URL: `http://blog.comcast.com/2010/02/dnssec.html`, february 210.

[GTS+06] S. Grau, H. Tschofenig, M. Shanmugam, L. Coene, and S. Gros. IKEv2 Mobility and Multihoming using SCTP, june 2006.

[Gui06] A. Guillard. DNSSEC Operational Impact and Performance. In *Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on*, pages 63–63, august 2006.

[Gur08] A. Gurtov. *Host Identity Protocol (HIP): towards the secure mobile Internet*. Wiley series in communications networking & distributed systems. Wiley, 2008.

[Han09] A. Handa. Mobile Data Offload for 3G Networks. URL: `http://www.intellinet-tech.com`, october 2009.

[Han10] A. Handa. 3G/WiFi Seamless Offload. URL: `http://www.qualcomm.com/documents/files/3g-wifi-seamless-offload.pdf`, march 2010.

[HG12] T. Henderson and A. Gurtov. The Host Identity Protocol (HIP) Experiment Report. URL: `http://www.ietf.org/rfc/rfc6538.txt`, march 2012. RFC 6538 (Informational).

[HHK+10] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan. Cellular traffic offloading through opportunistic communications: a case study. In *Proceedings of the 5th ACM workshop on Challenged networks*, CHANTS '10, pages 31–38, New York, NY, USA, 2010. ACM.

[HHS10] B. Han, P. Hui, and A. Srinivasan. Mobile data offloading in metropolitan area networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 14:28–30, november 2010.

[Hig09] K. J. Higgins. Kaminsky Calls For DNSSEC Adoption. URL: `http://www.darkreading.com/security/vulnerabilities/showArticle.jhtml?articleID=214501924`, february 2009.

[HJH+10] T. Heer, T. Jansen, R. Hummen, S. Götz, H. Wirtz, E. Weingärtner, and K. Wehrle. PiSA-SA: Municipal Wi-Fi Based on Wi-Fi Sharing. In *ICCCN*, pages 1–8, 2010.

[Hob10] A. Hoban. Using Intel AES New Intsruction and PCLMULQDQ to Significantly Improve IPSec Performance on Linux. In *Intel Corporation*, august 2010.

[HRUT06] C. Hohendorf, E. P. Rathgeb, E. Unurkhaan, and M. Tüxen. Secure End-to-End Transport over SCTP. In G. Müller, editor, *Emerging Trends in Information and Communication Security, International Conference, ETRICS 2006, Freiburg, Germany, June 6-9, 2006, Proceedings*, volume 3995 of *Lecture Notes in Computer Science*, pages 381–395. Springer, 2006.

[HS06] H. Haverinen and J. Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). URL: http://www.ietf.org/rfc/rfc4186.txt, january 2006. RFC 4186 (Informational).

[ICA09] ICANN. Immediate security concerns addressed by DNSSEC. URL: http://www.icann.org/en/announcements/announcement-2-03jun09-en.htm, june 2009.

[iPa] iPass: Enterprise Mobility Services. URL: http://www3.ipass.com/.

[IRD02] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *12th ACM Symposium on Principles of Distributed Computing (PODC 2002)*, july 2002.

[Jac09] W. Jackson. 5 IT priorities for 2009. URL: http://gcn.com/articles/2009/01/12/5-it-priorities-for-2009.aspx, january 2009.

[JBB03] J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based Internet Caches. In *IEEE Infocom 2003*, San Francisco, CA, april 2003.

[JMDJ+07] C. Jean-Michel, M. Daniel, B. Julien, C. Hakima, and L.-M. Maryline. Sécurité des réseaux mobiles ip. In *Traité IC2 (Hermès) "Sécurité des réseaux sans fil et mobiles"*. Lavoisier, 2007.

[JMDJ+09] C. Jean-Michel, M. Daniel, B. Julien, C. Hakima, and L.-M. Maryline. Security of ip-based mobile networks. In *Wireless and Mobile Network Security*, ISTE (International Society for Technology in Education), London, UK, 2009. Wilet.

[JPA04] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. URL: http://www.ietf.org/rfc/rfc3775.txt, june 2004. RFC 3775 (Proposed Standard), Obsoleted by RFC 6275.

[JRa] JRat The Java Runtime Analysis Toolkit. http://jrat.sourceforge.net/.

[JSBM01] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS performance and the effectiveness of caching. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 153–167, New York, NY, USA, 2001. ACM.

[Kam08a] D. Kaminsky. Dan kaminsly blog. URL: http://www.doxpara.com/?p=1185, july 2008.

[Kam08b] D. Kaminsky. It's The End Of The Cache As We Know It, or 64K Should Be Good Enough For Anyone. URL: http://www.doxpara.com/DMK_BO2K8.ppt, july 2008.

[Ken05a] S. Kent. IP Authentication Header. URL: http://www.ietf.org/rfc/rfc4302.txt, december 2005. RFC 4302 (Proposed Standard).

[Ken05b] S. Kent. IP Encapsulating Security Payload (ESP). URL: http://www.ietf.org/rfc/rfc4303.txt, december 2005. RFC 4303 (Proposed Standard).

[KES06] T. Koponen, P. Eronen, and M. Särelä. Resilient connections for SSH and TLS. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 30–30, Berkeley, CA, USA, 2006. USENIX Association.

[KHNE10] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). URL: http://www.ietf.org/rfc/rfc5996.txt, september 2010. RFC 5996 (Proposed Standard), Updated by RFC 5998.

[Kol05] O. M. Kolkman. Measuring the resource requirements of DNSSEC. URL: http://ripe.net/docs/ripe-352.html, october 2005.

[Kol09] O. Kolkman. DNSSEC HOWTO, a tutorial in disguise. URL: http://www.nlnetlabs.nl/dnssec_howto/dnssec_howto.pdf, july 2009.

[KS05] S. Kent and K. Seo. Security Architecture for the Internet Protocol. URL: http://www.ietf.org/rfc/rfc4301.txt, december 2005. RFC 4301 (Proposed Standard), Updated by RFC 6040.

[LB08] S. Lindskog and A. Brunstrom. A Comparison of End-to-End Security Solutions for SCTP. Technical report, Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW 2008). Karlskrona, Sweden, april 2008.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA, 2002. ACM.

[LCGW09] J. Livingood, T. Creighton, C. Griffiths, and R. Webe. Recommended Configuration and Use of DNS Redirect by Service Providers. URL: http://tools.ietf.org/html/draft-livingood-dns-redirect-00, july 2009.

[LKS] LKSCTP Linux Kernel SCTP. URL: http://sourceforge.net/projects/lksctp/.

[LM03] W. Lehr and L. W. Mcknight. Wireless Internet access: 3G vs. WiFi? *Telecommunications Policy*, 27(5-6):351–370, 2003.

[LRL+10] K. Lee, I. Rhee, J. Lee, Y. Yi, and S. Chong. Mobile data offloading: how much can WiFi deliver? *SIGCOMM Comput. Commun. Rev.*, 40:425–426, august 2010.

[LSAB08] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. URL: http://www.ietf.org/rfc/rfc5155.txt, march 2008. RFC 5155 (Proposed Standard).

[Mas06] D. Massey. A Comparative Study of the DNS Design with DHT-Based Alternatives. In *In the Proceedings of IEEE INFOCOM'06*, 2006.

[MD00] M. Mealling and R. Daniel. The Naming Authority Pointer (NAPTR) DNS Resource Record. URL: http://www.ietf.org/rfc/rfc2915.txt, september 2000. RFC 2915 (Proposed Standard), Obsoleted by RFCs 3401, 3402, 3403, 3404.

[MGL10] D. Migault, C. Girard, and M. Laurent. A performance view on dnssec migration. In *CNSM 2010*, pages 469–474, oct 2010.

[MHS+a] D. Migault, E. Herbert, S. F. Stanislas, S. Sénécal, and M. Laurent. "analyzing traffic and building routing tables for increasing dns(sec) resolving platforms efficiency (under submission)".

[MHS+b] D. Migault, E. Herbert, S. F. Stanislas, S. Sénécal, and M. Laurent. "overcoming dnssec performance issues with fqdn load balancer and cache sharing (under submission)".

[Mic] Microsoft Mobile VPN. URL: http://msdn.microsoft.com/en-us/library/cc440255.aspx.

[Mic09] Microsoft. Domain Name System Security Extensions. URL: http://www.microsoft.com/downloads/details.aspx?FamilyID=7a005a14-f740-4689-8c43-9952b5c3d36f&DisplayLang=en, february 2009.

[Mig10] D. Migault. Performance Measurements on BIND9/NSD/UNBOUND. In *IETF79/IEPG*. IEPG, november 2010.

[ML11] D. Migault and M. Laurent. How dnssec resolution platforms benefit from load balancing traffic according to fully qualified domain name. In *ICSNA International Conference on Secure networking and Applications (ICSNA)*, october 2011.

[MM06] D. Migault and B. Marinoiu. Evaluation du coût de la sécurisation du système dns. In *École Supérieure et d'Application des Transmissions*, SSTIC'06, pages 340–360, june 2006.

[MN06] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. URL: `http://www.ietf.org/rfc/rfc4423.txt`, may 2006. RFC 4423 (Informational).

[MNJH08] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. URL: `http://www.ietf.org/rfc/rfc5201.txt`, april 2008. RFC 5201 (Experimental), Updated by RFC 6253.

[Moc87a] P. Mockapetris. Domain names - concepts and facilities. URL: `http://www.ietf.org/rfc/rfc1034.txt`, november 1987. RFC 1034 (Standard), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[Moc87b] P. Mockapetris. Domain names - implementation and specification. URL: `http://www.ietf.org/rfc/rfc1035.txt`, november 1987. RFC 1035 (Standard), Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966.

[MPH+12a] D. Migault, D. Palomares, E. Herbert, W. You, G. G. G. Arfaoui, and M. Laurent. Isp offload infrastructure to minimize cost and time deployment. In *Proc. of IEEE Global Telecommunications Conference - Communication and Information System Security (GLOBECOM '12)*, december 2012.

[MPH+12b] D. Migault, D. Palomares, E. Herbert, W. You, G. Ganne, G. Arfaoui, and M. Laurent. E2e: An optimized ipsec architecture for secure and fast offload. In *International Workshop on Security of Mobile Applications IWSMA'12 (co-located with the ARES'12)*, august 2012.

[MPL] D. Migault, D. Palomares, and M. Laurent. MOBIKE-X to overcome Simultaneous Support of Security, Mobility, Multihoming and Multiple Interfaces.

[MS10] R. Malik and R. Syal. Performance Analysis of IP Security VPN. In *International Journal of Computer Applications*, volume 8, pages 0975–8887, october 2010.

[NB09] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. URL: `http://www.ietf.org/rfc/rfc5533.txt`, june 2009. RFC 5533 (Proposed Standard).

[NGH10] P. Nikander, A. Gurtov, and T. R. Henderson. Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks. *IEEE Communications Surveys and Tutorials*, 12(2):186–204, 2010.

[NL11] T. Norman and R. Linton. The case for Wi-Fi offload: the costs and benefits of Wi-Fi as a capacity overlay in mobile networks. Technical report, Analysys Masson, december 2011.

[NM08] P. Nikander and J. Melen. A Bound End-to-End Tunnel (BEET) mode for ESP. (Work in Progress), IETF, august 2008.

[NVCGRGL11] P. Noriega-Vivas, C. Campo, C. Garcia-Rubio, and E. Garcia-Lozano. Supporting L3 Femtocell Mobility Using the MOBIKE Protocol. Technical report, ACCESS 2011 : The Second International Conference on Access Networks, april 2011.

[Obe09] R. K. Oberman. DNSSEC Implementation. North American Network Operator' Group (NANOG45), january 2009.

[Oll05] G. Ollmann. The Pharming Guide : Understanding and Preventing DNS-related Attacks by Phishers. URL: `http://www.ngssoftware.com/papers/ThePharmingGuide.pdf`, august 2005.

[Oll09] G. Ollmann. Measures to Protect Domain Registration Services Against Exploitation or Misuse. URL: `http://www.icann.org/en/committees/security/sac040.pdf`, august 2009.

[ORMZ08] E. Osterweil, M. Ryan, D. Massey, and L. Zhang. Quantifying the Operational Status of the DNSSEC Deployment. Internet Measurement Conference, october 2008.

[OY02] L. Ong and J. Yoakum. An Introduction to the Stream Control Transmission Protocol (SCTP). URL: `http://www.ietf.org/rfc/rfc3286.txt`, may 2002. RFC 3286 (Informational).

[Per02] C. Perkins. IP Mobility Support for IPv4. URL: `http://www.ietf.org/rfc/rfc3220.txt`, january 2002. RFC 3220 (Proposed Standard), Obsoleted by RFC 3344.

[Per10] C. Perkins. IP Mobility Support for IPv4, Revised. URL: `http://www.ietf.org/rfc/rfc5944.txt`, november 2010. RFC 5944 (Proposed Standard).

[Phe08] T. Phelan. Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP). URL: `http://www.ietf.org/rfc/rfc5238.txt`, may 2008. RFC 5238 (Proposed Standard).

[PPPW04a] K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS: improving DNS performance and reliability via cooperative lookups. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pages 14–14, Berkeley, CA, USA, 2004. USENIX Association.

[PPPW04b] K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS: Masking DNS delays via Cooperative Lookups. In *Technical Report TR-690-04*. Princeton University Computer Science, 2004.

[RA10] S. Risto and L. Antti. Operator's Dilemma : How to take advantage of the growing mobile Internet. URL: `http://www.notava.com/notava/uploads/Whitepapers/Internet_growth_V10.pdf`, may 2010.

[Rad10] Radaware Alteon Application Switch 5412 Case study. URL: `http://www.radwarealteon.com/wp-content/uploads/Documents/CaseStudies/NAS/Alteon_Case_Study.pdf`, 2010.

[Rat04] M. Ratola. Which Layer for Mobility? - Comparing Mobile IPv6, HIP and SCTP. *HUT T-110.551 Seminar on Internetworking*, 2004.

[RD01a] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Computer Science*, 2218:329, 2001.

[RD01b] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *18th ACM Symposium on Operating Systems Principles (SOSP'01)*, pages 188–201, october 2001.

[RDM08] A. Roy, A. Datta, and J. C. Mitchell. Formal proofs of cryptographic security of Diffie-Hellman-based protocols. In *Proceedings of the 3rd conference on Trustworthy global computing*, TGC'07, pages 312–329, Berlin, Heidelberg, 2008. Springer-Verlag.

[rep09] DNSSEC Reply Size Test Server. URL: `https://www.dns-oarc.net/oarc/services/replysizetest`, july 2009.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[RHM09] J. Risson, A. Harwood, and T. Moors. Topology dissemination for reliable one-hop distributed hash tables. *IEEE Transactions on Parallel and Distributed Systems*, 20:680–694, 2009.

[Ric09] W. Rickard. The Long Road to DNSSEC Deployment. URL: `http://www.isoc.org/tools/blogs/ietfjournal/?p=1367`, september 2009.

[Roy09] Roy. FORMAL PROOFS OF CRYPTOGRAPHIC SECURITY OF NETWORK PROTOCOLS. URL: `seclab.stanford.edu/pcl/papers/Roy-Thesis-Final.pdf`, december 2009.

[RRDO10] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension. URL: `http://www.ietf.org/rfc/rfc5746.txt`, february 2010. RFC 5746 (Proposed Standard).

[RS04] V. Ramasubramanian and E. G. Sirer. Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 8–8, Berkeley, CA, USA, 2004. USENIX Association.

[RS07] V. Ramasubramanian and E. G. Sirer. Proactive Caching for Better than Single-Hop Lookup Performance. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.2.8918`, 2007.

[RT07] M. Riegel and M. Tuexen. Mobile SCTP. Internet-Draft (Work in progress), `http://tools.ietf.org/html/draft-riegel-tuexen-mobile-sctp-09`, IETF, november 2007.

[Sar10] P. Saragiotis. Good practices guide for deploying DNSSEC, january 2010.

[SCAC09] J. Schönwälder, G. Chulkov, E. Asgarov, and M. Cretu. Session resumption for the secure shell protocol. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pages 157–163, Piscataway, NJ, USA, 2009. IEEE Press.

[sct] SCTPlib: The SCTP library. URL: `http://www.sctp.de/sctp-download.html`.

[Ses08] S. Seshadri. DNSSEC on Windows 7 DNS client. URL: http://blogs.technet.com/sseshad/archive/2008/11/11/dnssec-on-windows-7-dns-client.aspx, november 2008.

[SFA04] Space, S. Fu, and M. Atiquzzaman. SCTP: State of the Art in Research, Products, and Technical Challenges, april 2004.

[SGM07] C. A. Shue, M. Gupta, and S. A. Myers. IPSec: Performance Analysis and Enhancements. In *IEEE International Conference on Communications (ICC)*, june 2007.

[SL10] S. Seshadri and G. Lindsay. DNSSEC Deployment Guide, march 2010.

[SMK+01] I. Stoica, R. Morris, D. Karger, F. M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 31, pages 149–160, New York, NY, USA, october 2001. ACM Press.

[Spr] Sprint: Global provider of voice, data and Internet services. URL: http://www.sprint.com/index_p.html?context=CP.

[Ste07] R. Stewart. Stream Control Transmission Protocol. URL: http://www.ietf.org/rfc/rfc4960.txt, september 2007. RFC 4960 (Proposed Standard), Updated by RFCs 6096, 6335.

[str] StrongSwan the OpenSource IPsec-based VPN Solution. URL: http://www.strongswan.org.

[SXT+07] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. URL: http://www.ietf.org/rfc/rfc5061.txt, september 2007. RFC 5061 (Proposed Standard).

[TDVK98] R. Tewari, M. Dahlin, H. Vin, and J. Kay. Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet, 1998. citeseer.ist.psu.edu/tewari98beyond.html.

[Tel] Telenor. URL: http://www.telenor.no/privat/.

[The] TheCloud. URL: http://www.thecloud.net/.

[Tru] Trustive | International mobile internet access (3G + WiFi access) in 150 countries worlwide, including at 300.000 WiFi hotspots from a leading international WiFI access provider & No1 in Europe. URL: http://www.trustive.com/.

[UNB] UNBOUND. URL: http://unbound.net/.

[URJ04] E. Unurkhaan, E. P. Rathgeb, and A. Jungmaier. Secure SCTP - A Versatile Secure Transport Protocol. *Telecommunication Systems*, 27(2-4):273–296, 2004.

[VGEW00] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). URL: http://www.ietf.org/rfc/rfc2845.txt, may 2000. RFC 2845 (Proposed Standard), Updated by RFC 3645.

[Wan99] J. Wang. A survey of Web caching schemes for the Internet. *ACM Computer Communication Review*, 25(9):36–46, 1999. citeseer.ist.psu.edu/wang99survey.html.

[WBS04] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the web from DNS. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 17–17, Berkeley, CA, USA, 2004. USENIX Association.

[Wei07] S. Weiler. DNSSEC Lookaside Validation (DLV). URL: `http://www.ietf.org/rfc/rfc5074.txt`, november 2007. RFC 5074 (Informational).

[Wel] B. Wellington. dnsjava. URL: `http://www.dnsjava.org/`.

[WeR] WeRoam: Comfone's Global Public WiFi Access Solution. URL: `http://www.comfone.com/index.php/weroam`.

[WG03] B. Wellington and O. Gudmundsson. Redefinition of DNS Authenticated Data (AD) bit. URL: `http://www.ietf.org/rfc/rfc3655.txt`, november 2003. RFC 3655 (Proposed Standard), Obsoleted by RFCs 4033, 4034, 4035.

[Wik] Wikipedia. Quartile. URL: `http://en.wikipedia.org/wiki/Quartile`.

[Wir] Wireshark. URL: `http://www.wireshark.org`.

[Wou10] P. Wouters. World Wide DNSSEC Deployment. URL: `http://www.xelerance.com/dnssec/`, 2010.

[WVS+99] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. R. Karlin, and H. M. Levy. On the scale and performance of cooperative Web proxy caching. In *Symposium on Operating Systems Principles*, pages 16–31, 1999.

[XMSF11] Q. Xu, D. Migault, S. Sénécal, and S. Francfort. K-means and adaptive k-means algorithms for clustering dns traffic. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11, pages 281–290, ICST, Brussels, Belgium, Belgium, may 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[ZHS+04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), january 2004.