Découverte et gestion distribuée de chemins alternatifs à contraintes de qualité de service dans l'internet

Thierry Rakotoarivelo

N° d'ordre : 2448

Thèse

préparé au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
et à l'**École Nationale Supérieur d'Ingénieurs de Constructions Aéronautique**

en co-tutelle avec
l'**University of New South Wales, Sydney, Australie**
et **National ICT Australia, Australie**

Pour obtenir le titre de
DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE
École doctorale : Informatique et Télécommunications
Spécialité : Réseaux et Télécommunications

Par
M. Thierry RAKOTOARIVELO

Découverte et Gestion Distribuée de Chemins Alternatifs à contraintes de Qualité de Service dans l'Internet

---

Soutenue le 31 janvier 2007 devant le jury composé de :

Président :	Maximilian	OTT
Directeur de thèse :	Michel	DIAZ
Co-Directeurs :	Aruna	SENEVIRATNE
	Patrick	SÉNAC
Rapporteurs :	Anne-Marie	KERMARREC
	Laurent	MATHY
Membre :	Marius	PORTMANN

Acknowledgements

First, I would like to thank my advisors Patrick Sénac, Aruna Seneviratne and Michel Diaz for their trust in me throughout the last four years. Aruna welcomed me into his research groups at the University of New South Wales and then later at the National ICT Australia. He gave me the opportunity to undertake this PhD study. Patrick offered me his guidance and advice on every part of this work. I am very grateful for his invaluable supervision. Michel provided important and constructive feedback at key points during the development of this research work.

I would also like to thank Anne-Marie Kermarrec, Laurent Mathy, and Marius Portmann for their interest in this work, their valuable comments, and for accepting to be part of my thesis review commitee.

I would also like thank all the former members of the "late" MobQoS group (Sebastien, Binh, Zhe Guang, Robert, Tim, Krit, Stephen, etc...), the people from the DMI group at ENSICA (Laurent, Tanguy, Jérôme, Yves, Ernesto, all the PhD students and staff), and the people from the NPC group at NICTA (Emmanuel, Guillaume, etc...). Thank you all for your help, your technical advice, the long fruitful discussions, and for making the last few years more enjoyable.

Finally, I would like to express my profound gratitude to my parents, my sister, and my partner Tiana. They encouraged and inspired me in many ways. I would have never made it through this journey without their love and their continuous support.

Abstract

The convergence of recent technology advances opens the way to new ubiquitous environments, where network-enabled devices collectively form invisible pervasive computing and networking environments around the users. These users increasingly require extensive applications and capabilities from these devices. Recent approaches propose that cooperating *service providers*, at the edge of the network, offer these required capabilities (i.e *services*), instead of having them directly provided by the devices. Thus, the network evolves from a plain communication medium into an endless source of services. Such a service, namely an *overlay application*, is composed of multiple distributed application elements, which cooperate via a dynamic communication mesh, namely an *overlay association*. The Quality of Service (QoS) perceived by the users of an *overlay application* greatly depends on the QoS on the communication paths of the corresponding *overlay association*.

This thesis asserts and shows that it is possible to provide QoS to an *overlay application* by using alternate Internet paths resulting from the compositions of independent consecutive paths. Moreover, this thesis also demonstrates that it is possible to discover, select and compose these independent paths in a distributed manner within an community comprising a limited large number of autonomous cooperating peers, such as the fore-mentioned *service providers*. Thus, the main contributions of this thesis are i) a comprehensive description and QoS characteristic analysis of these composite alternate paths, and ii) an original architecture, termed SPAD (Super-Peer based Alternate path Discovery), which allows the discovery and selection of these alternate paths in a distributed manner. SPAD is a fully distributed system with no single point of failure, which can be easily and incrementally deployed on the current Internet. It empowers the end-users at the edge of the network, allowing them to directly discover and utilize alternate paths.

Keywords:

Quality of Service, Overlay Networks, Peer-to-Peer Systems, Service-oriented Networks

Résumé

La convergence de récentes avancées technologiques permet l'émergence de nouveaux environnements informatiques pervasifs, dans lesquels des terminaux en réseaux coopèrent et communiquent de manière transparente pour les utilisateurs. Ces utilisateurs demandent des fonctionalités de plus en plus avancées de la part de ces terminaux. Etant données les limites intrinsèques des terminaux mobiles, ces fonctionalités, au lieu d'être directement implémentées dans les terminaux, sont appelées à être fournies par des *fournisseurs de services* situés à la périphérie du réseau. Ce derniers devient alors une source illimitée de services, et non plus seulement un medium de communication. Ces services, ou *applications d'overlays*, sont formés de plusieurs éléments applicatifs distribués qui coopèrent et communiquent entre eux via un réseau de recouvrement dynamique particulier, une *association d'overlay*. La Qualité de Service (QdS) perçue par les utilisateurs d'une *application d'overlay* dépend de la QdS existant au niveau des chemins de communications qui forment l'*association d'overlay* correspondante.

Cette thèse montre qu'il est possible de fournir de la QdS à une *application d'overlay* en utilisant des chemins Internet alternatifs, résultant de la composition de chemins distincts. De plus, cette thèse montre également qu'il est possible de découvrir, sélectionner, et composer d'une manière distribuée ces chemins élémentaires, au sein d'une communauté comprenant un nombre important d'entités paires (telles que les précédents *fournisseurs de services*). Les principales contributions de cette thèse sont : i) une description et une analyse des caractéristiques de QdS de ces chemins alternatifs composés, ii) une architecture originale appelée SPAD (Super-Peer based Alternate path Discovery), qui permet la découverte et la sélection de manière distribuée de ces chemins alternatifs. SPAD est un système complètement décentralisé, qui peut être facilement et incrémentalement déployé sur l'Internet actuel. Il permet aux utilisateurs situés à la périphérie du réseau de découvrir et d'utiliser directement des chemins alternatifs.

Mots-Clés :

Qualité de Service, Réseaux de Recouvrement, Système Pair-à-Pair, Réseaux de Services

Table of Contents

1 Résumé de la thèse en français **19**

1.1 Introduction (chapitre 2) . . . 19

1.1.1 Contexte et Problématique . . . 19

1.1.2 Contributions . . . 20

1.2 Le Réseau : une Source Illimitée de Services (Chapitre 3) . . . 22

1.3 La Qualité de Service dans le cadre d'Applications Distribuées (Chapitre 4) 22

1.4 Les QEAPS : Chemins Alternatifs proposant une QoS améliorée (Chapitre 5) 23

1.5 SPAD : un Système Distribué de Recherche et de Sélection de QEAPs (Chapitre 6) . . . 24

1.6 Evaluation du Système SPAD (Chapitre 7) . . . 25

1.7 Conclusion (chapitre 8) . . . 25

2 Introduction **27**

2.1 Computing & Networking, From static to mobile . . . 27

2.2 Contributions of this work . . . 28

2.3 Dissertation Overview . . . 30

3 The Network, an Endless Source of Services **31**

3.1 Introduction . . . 31

3.2 The Concept of Service Composition . . . 32

3.2.1 What is Service Composition ? . . . 32

3.2.2 Challenges and Related Work . . . 33

3.2.3 Summary . . . 37

3.3 The Concept of Overlay Networks . . . 37

3.3.1 What is an Overlay Network ? . . . 37

3.3.2 Related Work: Overlay Networks on the current Internet . . . 38

3.3.3 Overlay Network Challenges . . . 39
3.3.4 Summary . . . 40
3.4 Chapter Summary . . . 40

4 Quality of Service for Distributed Applications 43
4.1 Introduction . . . 43
4.2 The Notion of Quality of Service . . . 44
4.2.1 QoS from the user perspective . . . 44
4.2.2 QoS from the application perspective . . . 44
4.2.3 QoS from the network perspective . . . 45
4.2.4 Mapping QoS requirements across these different perspectives . . . 45
4.3 Classic Approaches to QoS provision . . . 45
4.3.1 Network Level . . . 45
4.3.2 End-to-End and Application Level . . . 50
4.4 Another Approach to end-to-end QoS provision . . . 51
4.4.1 Virtual Paths on Overlay Networks . . . 52
4.4.2 Related Work . . . 53
4.4.3 Discovering Virtual Paths . . . 54
4.5 Chapter Summary . . . 55

5 QoS Enhanced Alternate Paths 57
5.1 Introduction . . . 57
5.2 The origin of QEAPs . . . 58
5.3 Presentation of the Experimental Data Sets . . . 59
5.4 Characteristic Analysis of QEAPs . . . 60
5.4.1 Existence . . . 61
5.4.2 Gain . . . 62
5.4.3 Constancy . . . 64
5.4.4 The Case for Considering both Delay and Packet-Loss Parameters . . . 65
5.4.5 Comparison between 2-hop and 3-hop QEAPs . . . 66
5.5 Chapter Summary . . . 67

6 SPAD: Super-Peer based Alternate path Discovery architecture 69
6.1 Introduction . . . 69

6.2 A Simple QEAP Discovery Scheme . . . 70
6.2.1 Scheme Overview . . . 70
6.2.2 Peer Initialization . . . 70
6.2.3 Scheme Description . . . 71
6.2.4 Discussion . . . 73
6.3 SPAD: Architecture and Design Assumptions . . . 73
6.3.1 Architecture Overview . . . 73
6.3.2 Design Requirement and Assumptions . . . 75
6.4 SPAD: System Description . . . 78
6.4.1 SPAD Peer Initialization . . . 78
6.4.2 SPAD Reactive Information Exchange Scheme . . . 79
6.4.3 SPAD Proactive Information Exchange Scheme . . . 81
6.4.4 Integration of Loss-QEAP Search in SPAD . . . 89
6.4.5 SPAD QEAP Selection Schemes . . . 90
6.5 Towards a Complete SPAD System . . . 93
6.6 Chapter Summary . . . 94

7 SPAD Performance Evaluation 97
7.1 Introduction . . . 97
7.2 Evaluation of A Simple QEAP Discovery Scheme . . . 98
7.3 Evaluation of SPAD Reactive Scheme . . . 99
7.4 Evaluation of SPAD Proactive Scheme . . . 100
7.5 Evaluation of SPAD QEAP Selection Scheme . . . 102
7.6 Discussion . . . 105
7.7 Chapter Summary . . . 106

8 Conclusion 109
8.1 Problem Summary . . . 109
8.2 Contribution Summary . . . 110
8.3 Future Research Directions and Perspectives . . . 111
8.4 Final notes . . . 113

Appendix 115

List of Publications **119**

Bibliography **121**

Glossary **131**

List of Figures **131**

List of Tables **134**

To my beloved Mother and Father,

I am eternally grateful for your love and your
unconditional encouragement in all my endeavours

Chapter 1

Résumé de la thèse en français

1.1 INTRODUCTION (CHAPITRE 2)

Cette partie propose un résumé en français du chapitre 2 de cette thèse. Ce chapitre introduit le contexte général dans lesquel se placent les travaux de recherche effectués dans le cadre ce cette thèse, ainsi que la problématique sur laquelle se concentrent plus particulièrement ces travaux. Cette partie présente ensuite les diverses contributions apportées par cette thèse afin de proposer une solution à ce problème.

1.1.1 Contexte et Problématique

La technologie alors disponible a imposé des dimensions physiques considérables aux premiers ordinateurs qui, tel l'ENIAC [1], pouvaient typiquement occuper une superficie de $100m^2$ pour une consommation énergétique de l'ordre de 150kW. Bien que les avancées technologiques des années suivantes aient successivement réduit l'ordre de grandeur de ces dimensions physiques, les premiers systèmes destinés à relier des terminaux informatiques entre eux afin de former un réseau informatique étaient aussi basés sur une contrainte similaire et implicite d'immobilité. Les avancées technologiques successives ont ensuite permis à ces terminaux informatiques d'atteindre un niveau de miniaturisation leur permettant d'être aisément transportés par les utilisateurs. Les systèmes de réseaux informatiques ont alors évolué parallèlement afin d'accomoder cette nouvelle hypothèse de mobilité. Cette évolution simultanée des terminaux et des réseaux a permis l'émergence du concept d'ubiquité informatique [2]. Dans ce concept, une multitude de terminaux informatiques mobiles et immobiles coopèrent via un ou des réseaux pour former un environnement où la technologie de communication parait invisible aux utilisateurs. Ces derniers cessent d'être conscients de la présence d'appareils et de réseaux informatiques, et les utilisent de manière complètement transparente.

Dans ce contexte d'informatique ambiante, les utilisateurs requièrent des fonctionnalités de plus en plus complexes. Afin de satisfaire ces besoins, une première approche consisterait à augmenter les capacités de calcul, de stockage et de communication de ces terminaux. Cependant, des contraintes liées à la portabilité, à la comsommation d'énergie ou encore au coût de production limitent considérablement l'application d'une telle solution. Une approche alternative serait de faire appel à des *fournisseurs de services*, localisés à différents endroits d'un réseau tel l'internet, pour fournir d'une manière coopérative ces nouvelles fonctionalités aux utilisateurs [3]. Ces fonctionalités seraient alors composées de différents éléments applicatifs qui seraient hébergés par plusieurs machines hôtes distribuées à différents endroits dans le réseau. Dans le cadre de cette approche, le réseau ne se limite plus à un rôle de medium de communication, mais devient également une source illimitée d'éléments applicatifs que les terminaux peuvent organiser afin d'obtenir des applications composées fournissant des fonctionalités complexes.

Ces applications composées peuvent notamment créer, recevoir, ou transformer des données ou flux multimédias. Dans ces cas, la perception des utilisateurs de la qualité fournie par l'application ainsi composée dépend de manière significative de la qualité de service (QdS) existant sur les connections qui relient les divers élements applicatifs composant cette application. L'amélioration de la qualité de service de ces applications composées distribuées constitue encore actuellement un sujet de recherche ouvert. Cette thèse s'intéresse à cette problématique. Plus particulièrement, cette thèse propose une solution au problème de l'amélioration de la qualité de service entre les divers composants élémentaires constituant une application distribuée. Ces composants élémentaires sont répartis en plusieurs points de l'Internet, et sont donc reliés entre eux par des *chemins* dans l'Internet.

1.1.2 Contributions

Plusieurs travaux de recherches ont proposé des solutions au problème de l'amélioration de la QdS sur l'Internet. Le chapitre 4 de cette thèse (résumé en français dans la section 1.3 suivante) présente et analyse certains de ces travaux. Cette thèse adopte une approche originale et récursive par rapport à ces travaux précédents. Une application composée fournit un service complexe aux utilisateurs en assemblant et utilisant plusieurs applications élémentaires provenant de divers fournisseurs de services. Cette thèse applique ce même modèle de manière récursive, et propose une approche où la qualité de service est aussi considérée comme un service complexe qui est donc fournie aux utilisateurs par le biais d'une composition de services élémentaires d'amélioration de la QdS entre les différents composants applicatifs. Toujours selon ce même modèle, un service élémentaire d'amélioration de la QdS entre deux composants applicatifs est fournie par la composition de chemins Internet successifs et indépendants. Cette composition produit un *chemin alternatif* qui peut apporter une amélioration de la QdS par rapport au chemin par défaut donné par le service de routage de paquets de l'Internet. Cette approche repose sur la diversité des chemins entre deux noeuds dans l'Internet, mise en évidence dans [4]. Par ailleurs, les fournisseurs de services étant autonomes et situés à différents points du réseau, la découverte et la composition de ces *chemins alternatifs* doivent être effectuées de manière distribuée.

Cette thèse montre qu'il est possible de fournir de la QdS, et dans une certaine mesure de la contrôler, à une application distribuée en utilisant des chemins Internet alternatifs, résultant de la composition de chemins distincts. De plus, cette thèse montre également qu'il est possible de découvrir, sélectionner, et composer d'une manière distribuée ces chemins élémentaires, au sein d'une communauté comprenant un nombre important d'entités paires (telles que les précédents fournisseurs de services).

Les principales contributions de cette thèse sont :

– une description et une analyse des caractéristiques de QdS des chemins alternatifs composés mentionnés precédemment. Cette analyse complète et étend les précédents travaux de recherche sur le sujet [4, 5, 6]. Elle montre l'existence dans l'Internet de chemins alternatifs qui fournissent des avantages conséquents aux paquets de données les empruntant. Ces chemins peuvent par exemple fournir un délai et un taux de perte de paquets inférieurs à ceux fournis par les chemins par défaut de l'Internet. Dans la suite de cette thèse, le terme QEAP (QoS Enhanced Alternate Path) est utilisé pour définir de tels chemins alternatifs. Cette analyse initiale supporte l'approche proposant d'utiliser des QEAPs afin de fournir de la QdS entre deux noeuds distants de l'Internet.

– une architecture originale appelée SPAD (Super-Peer based Alternate path Discovery), qui permet la découverte et la sélection de manière distribuée de ces chemins alternatifs. SPAD est basé sur une communauté de noeuds coopérants qui forment un réseau non structuré de *super-pairs* [7]. Ces noeuds utilisent les mécanismes distributés de SPAD pour rechercher, sélectionner, construire et utiliser des QEAPs. SPAD est un système complètement décentralisé, sans point de vulnérabilité unique, qui peut être facilement et incrémentalement déployé sur l'Internet actuel. Il permet aux utilisateurs situés à la périphérie du réseau de découvrir et d'utiliser directement des chemins alternatifs. De plus, SPAD s'inscrit dans une architecture globale de type *middleware*, située sur les machines en périphérie de l'Internet. Par conséquent, le déployement de SPAD sur l'Internet actuel ne nécessite aucune modification des mécanismes et des équipements de routage existants.
  A travers cette approche originale de découverte, de sélection et d'utilisation distribuées de QEAPs, SPAD permet donc de fournir de la QdS entre les divers éléments applicatifs composant une application distribuée.

– une évaluation approfondie des performances de SPAD. Cette évaluation est basée sur des expériences utilisant des données de mesures relatives à différents sous-ensembles de l'Internet. De plus, elle montre que SPAD permet effectivement la découverte et la sélection de QEAPs au sein d'une communauté comprenant un grand nombre d'entités paires coopérantes.

Les parties suivantes de ce chapitre présentent les résumés en français des chapitres suivant le chapitre 2 de cette thèse.

1.2 LE RÉSEAU : UNE SOURCE ILLIMITÉE DE SERVICES (CHAPITRE 3)

Afin de fournir aux utilisateurs de terminaux mobiles les fonctionalités complexes qu'ils requièrent, le chapitre 2 de cette thèse propose une approche dans laquelle des fournisseurs de services distribués dans le reseau mettent à disposition des éléments applicatifs composables. La composition de ces éléments permet d'obtenir les fonctionalités complexes requises. Cette approche repose sur deux concepts informatiques récents : la composition de services, et les réseaux de recouvrement. Le chapitre 3 propose une description détaillée de ces 2 concepts, ainsi qu'un panorama des travaux de recherche correspondants, et souligne la problématique à l'origine de cette thèse.

La composition de services est le concept clé dans la réalisation du modèle proposant de considérer le réseau comme une source illimitée de services. Le chapitre 3 décrit les différentes étapes liées à la composition de service, les domaines de recherche associés, ainsi que les contributions existantes. Le modèle des réseaux de recouvrement fournit un support adéquat au déploiement d'un système de composition de services au sein de l'Internet actuel. Un réseau de recouvrement est un réseau virtuel construit sur une infrastructure réseau existante. La communauté des utilisateurs et des fournisseurs de services intervenant dans un système de composition de services forment un réseau de recouvrement. De même, une instance particulière d'application distribuée constitue aussi un réseau de recouvrement dynamique et éphémère.

Tout au long du chapitre 3, les différentes études des précédentes contributions montrent qu'il n'existe pas encore de solution efficace à la problématique de mise en place de QdS pour un service composé. Par ailleurs, étant donné qu'une application composée déploie un réseau de recouvrement, la QdS liée à cette application dépend de la QdS entre les noeuds formant ce réseau de recouvrement. A ce titre, cette thèse a pour objectif de fournir de la QdS aux services composés, et plus particulièrement, de fournir une amélioration de la QdS entre les noeuds d'un réseau de recouvrement.

1.3 LA QUALITÉ DE SERVICE DANS LE CADRE D'APPLICATIONS DISTRIBUÉES (CHAPITRE 4)

Les éléments applicatifs au sein d'une application distribuée peuvent créer, recevoir, ou échanger des données multimedias à contraintes temporelles, tel qu'un flux audio continue. La percéption par l'utilisateur de la qualité de service (QdS) d'une telle application distribuée dépend de la gestion et de la transmission de ces données entre les différents éléments applicatifs.

Dans la communauté des réseaux informatiques, le terme QdS fait référence à différentes notions. Le chapitre 4 présente ces différentes notions à travers trois perspectives : celle de l'utilisateur, celle de l'application et celle du réseau. Ce chapitre décrit également les approches classiques visant à fournir une amélioration de la QdS entre deux hôtes de l'Internet, et propose une brève étude des diverses solutions implémentant ces approches, soulignant leurs avantages et leurs limites.

Le chapitre 4 présente ensuite l'approche alternative prise par cette thèse par rapport à ce problème. Une application distribuée offre une fonctionalité complexe (c.a.d. un service composé) aux utilisateurs, en composant divers éléments applicatifs indépendants. L'approche proposée dans cette thèse suit un modèle similaire. Elle propose de considérer la QdS d'une application distribuée comme un service composé qui serait alors le résultat de la composition de divers chemins virtuels offrant une QdS améliorée entre les différents éléments applicatifs associés. Plus précisement, cette approche propose de construire chacun de ces chemins virtuels en assemblant des chemins successifs indépendants de l'Internet. Un tel chemin virtuel constitue donc un chemin alternatif, différent du chemin par défaut fourni par les mécanismes de routage de l'Internet.

Le chapitre 4 met ensuite en évidence le problème de la découverte de ces chemins alternatifs virtuels. Il présente un panorama des précédents travaux apportant une solution à ce problème. Ce chapitre souligne le fait qu'aucun de ces travaux ne propose de solution applicable efficacement à une large communauté d'hôtes distribués, comparable à la communauté des fournisseurs de services présentée au chapitre 2. Finalement, ce chapitre introduit l'architecture SPAD (Super-Peer Alternate paths Discovery) proposée par cette thèse afin de permettre la découverte et la gestion de manière distribuée de chemins alternatifs à QdS améliorée dans l'Internet.

1.4 LES QEAPS : CHEMINS ALTERNATIFS PROPOSANT UNE QOS AMÉLIORÉE (CHAPITRE 5)

Le chapitre 5 présente la première contribution de cette thèse : une description et une analyse détaillée des caractéristiques des chemins Internet alternatifs à qualité de service ameliorée (QEAPs, QoS Enhanced Alternate Paths). La première partie de ce chapitre décrit l'origine de ces QEAPs. Leur existence est principalement due au fonctionnement du protocole BGP (Border Gateway Protocol), qui est le protocole de routage standard de-facto entre différents domaines autonomes (AS, Autonomous domain) de l'Internet. Par conséquent, les QEAPs sont inhérents à l'Internet actuel.

La partie suivante du chapitre 5 présente les ensembles de mesures utilisés dans les expériences realisées dans le cadre de cette thèse. Ces ensembles fournissent des mesures de délai de bout-en-bout, aller-simple et aller-retour, entre plusieurs hôtes de l'Internet. De plus, l'étude de ces mesures permet également d'évaluer le taux de perte de paquets entre ces hôtes. Ces ensembles de mesures sont accessible publiquement, et sont mis à disposition par les projets All-Pair-Ping de PlanetLab [8], Active Measurement Project de NLANR (National Laboratory for Applied Network Research) [9], et Test Traffic Measurement de RIPE (Réseaux IP Européens) [10].

La dernière partie du chapitre 5 présente les expériences réalisées afin d'analyser certaines caracteristiques des QEAPs, ainsi que les résultats correspondants. Ces résultats permettent d'obtenir les conclusions suivantes :

– un nombre important de QEAPs existe au sein de l'Internet actuel,
– la plupart de ces QEAPs fournissent un gain substantiel en terme de délai de bout-en-bout et de taux de perte de paquets,

– ces gains sont quasi-constants sur une durée moyenne permettant à la majorité des flux de données actuels de l'Internet de bénéficier avantageusement des QEAPs,
– la corrélation existante entre le délai et le taux de perte de paquets sur une même route dans l'Internet n'implique aucune correlation évidente entre les gains en délai et les gains en taux de perte de paquets sur les QEAPs. Un système de découverte des QEAPs doit donc prendre en compte ces deux paramètres afin de fournir un service optimal à ses utilisateurs,
– les QEAPs constitués de 3 chemins successifs (c.a.d 3 *sauts*) fournissent des gains négligeables comparés aux QEAPs à 2 *sauts*. Par conséquent, les mécanismes de découverte et de gestion presentés au chapitre 6 ne s'intéresseront qu'aux QEAPs à 2 *sauts*. Une éventuelle extension de ces mécanismes aux QEAPs à plus de 2 *sauts* ne requiert que des modifications mineures aux mécanismes proposés dans la suite de cette thèse.

Ces résultats montrent donc qu'il est possible d'obtenir une amélioration des paramètres de qualité de service entre deux hôtes de l'Internet en utilisant des chemins virtuels alternatifs tels que les QEAPs.

1.5 SPAD : UN SYSTÈME DISTRIBUÉ DE RECHERCHE ET DE SÉLECTION DE QEAPS (CHAPITRE 6)

Le chapitre 6 présente la seconde et principale contribution de cette thèse : l'architecture SPAD (Super-Peer Alternate path Discovery). SPAD est un système distribué visant à améliorer la QdS entre deux hôtes au sein d'une association d'overlay. Cette architecture est fondamentalement basée sur la découverte, la sélection et l'utilisation de chemins alternatifs dans l'Internet. Ces chemins sont composés de chemins successifs indépendants qui fournissent une QdS améliorée par rapport aux chemins par défaut de l'Internet. Le système SPAD repose sur une communauté d'hôtes coopérants, qui sont regroupés au sein d'un réseau non-structuré de *super-pairs* [7]. Dans cette communauté, certains hôtes agissent en tant que *simple-pair* et produisent des requêtes demandant des informations sur des chemins alternatifs vers d'autres hôtes de la communauté. Parallèlement, d'autres hôtes assument un rôle de *super-pair* et traitent ces requêtes afin de permettre la découverte de chemins alternatifs. Ces hôtes de type *super-pairs* relaient les requêtes entre eux, échangent des informations sur les connections existant au sein de la communauté, et retournent les informations pertinentes aux hôtes à l'origine des requêtes. Ces derniers utilisent ces informations reçues pour découvrir des QEAPs potentiels, et sélectionner celui qui correspond le mieux aux besoins applicatifs.

La première partie de ce chapitre présente une étude préliminaire d'un système élémentaire de découverte de QEAPs. Cette étude préliminaire montre qu'il est possible de découvrir des QEAPs de manière distribuée au sein d'un environnement coopératif. SPAD s'inspire de ce système élémentaire, et l'étend afin de fournir un système complet de découverte et de gestion des QEAPs, supportant plusieurs échelles de grandeur. La partie 6.3 décrit une vue d'ensemble de l'architecture SPAD, puis présente et discute les hypothèses retenues lors de la conception de ce système. La partie 6.4 propose une description detaillée des mécanismes de SPAD, suivants :

– deux mécanismes complèmentaires, l'un *re-actif* et l'autre *pro-actif*, permettant la découverte de chemins alternatifs à QdS améliorée,
– un mécanisme de sélection du QEAP optimal (par rapport aux besoins applicatifs) parmi les QEAPs découverts

L'évaluation des performances de ces différents mécanismes est presentée au chapitre 7.

1.6 EVALUATION DU SYSTÈME SPAD (CHAPITRE 7)

Ce chapitre présente et analyse les performances des mécanismes de SPAD permettant la découverte et la sélection de QEAPs. Les expériences realisées dans le cadre de ces évaluations de performances, sont basées sur les ensembles de mesures RIPE-TTM, NLANR-AMP, et PlanetLab introduits au chapitre 5.

La première partie de ce chapitre présente les résultats de l'évaluation du système élémentaire de découverte de QEAPs décrit au début du chapitre 6. Puis, les parties 7.3 et 7.4 analysent les performances des mécanismes complémentaires *re-actif* (c.f. la partie 6.4.2) et *pro-actif* (c.f. la partie 6.4.3) d'échange d'informations mis en oeuvre par SPAD. L'évaluation des mécanismes de sélection de QEAPs optimaux par rapport aux besoins applicatifs est proposée et discutée dans la partie 6.4.5. Finalement, la partie 7.6 décrit et commente certaines limites des expériences realisées.

Les différents résultats présentés dans ce chapitre montrent que le système SPAD proposé par cette thèse permet :

– la découverte de manière distribuée des QEAPs existants dans une communauté d'hôtes Internet coopérants,
– la sélection parmi les QEAPs découverts, de celui ayant les caractéristiques les plus optimales par rapport aux besoins applicatifs de QdS.

Par conséquent, ce chapitre valide l'approche et la solution proposées par cette thèse, au problème de la mise en œuvre de la QdS entre les hôtes faisant partie d'une *association d'overlay*.

1.7 CONCLUSION (CHAPITRE 8)

Ce chapitre conclut cette thèse. Il propose dans une première partie un resumé de la problématique abordée, en reprenant les principaux points et conclusions developpés dans les chapitres 2, 3, et 4. Ces chapitres montrent que cette thèse s'intéresse au problème de l'amélioration de la qualité de service entre les divers composants élémentaires constituant une application distribuée sur un réseau de recouvrement. Ces composants élémentaires sont répartis en plusieurs points de l'Internet, et sont donc reliés entre eux par des chemins sur l'Internet. A ce titre, cette thèse se concentre plus particulièrement sur le problème de

l'amélioration de la QdS sur les chemins reliant deux hôtes d'un réseau de recouvrement déployé sur de l'Internet.

La deuxième partie de ce chapitre présente un resumé de l'approche et de la solution originales proposées par cette thèse au précédent problème, et décrites de manière détaillée dans les chapitres 5, 6, et 7. Cette approche repose sur la diversité des chemins entre deux noeuds de l'Internet, et montrent qu'il est possible de fournir de la QdS à une application distribuée en utilisant des chemins Internet alternatifs, résultant de la composition de chemins élémentaires distincts. La solution adoptée propose un système complet incluant des mécanismes distribués permettant de découvrir, sélectionner, et composer ces chemins élémentaires, au sein d'une communauté comprenant un grand nombre d'entités paires. Ce systeme est complètement décentralisé, sans point de vulnérabilité unique, et peut être facilement et incrémentalement déployé sur l'Internet actuel, sans aucune modification des mécanismes et des équipements de routage existants. Il permet aux hôtes ou utilisateurs situés à la périphérie du réseau de découvrir et d'utiliser directement des chemins alternatifs afin d'améliorer la QdS de leurs communications.

Finalement, la dernière partie de ce chapitre présente plusieurs futures pistes de développement constituant autant de prolongements potentiels aux travaux de recherche menés dans le cadre de cette thèse. Ces perspectives incluent notamment : l'étude de système de contrôle d'admission des requêtes de chemins alternatifs, l'étude de mécanismes de decouverte de tels chemins dans un contexte pair-à-pair structuré, ou encore l'étude de l'utilisation de tels chemins en conjonction avec des mécanismes de codage réseau ou de code correcteur d'erreur afin d'optimiser l'utilisation d'un réseau de recouvrement donné.

Chapter 2

Introduction

2.1 COMPUTING & NETWORKING, FROM STATIC TO MOBILE

Academic and military research projects first designed, built and used productive electronic computers [11]. Examples of these computing machines include the British Colossus and the American ENIAC [1]. These machines required enormous space and power resources, on the order of $100m^2$ and 150kW for ENIAC. Thus, once such a computing machine was installed in its operating place, it was seldom moved to another location. The Internet has a similar origin, and its first embodiment, the ARPANET, had a similar implicit immobility assumption [12]. The ARPANET was a military funded project that aimed at connecting computers. Its protocols of communication, such as the earlier Network Control Protocol (NCP) [13] and the later Internet Protocol (IP) [14], were not designed to support either the mobility of a given computer within its surroundings or the network, nor the mobility of a given session between multiple computers.

Exponential advances in electronic technologies have led to the emergence of miniaturized, easily transportable computing devices that are affordable to the common consumer. Thus, computers have evolved from an earlier technology-bound static state to a mobile state that enables new application opportunities. Following this evolution, Internet networking technologies have developed new communication protocols to maintain and manage the connectivity of these mobile devices with each others and the Internet. Examples of such protocols are the Internet Protocol version 6 (IPv6) [15], and the Mobile IP protocol and its extensions [16]. The convergence of these computing and networking advances opens the way to new ubiquitous computing environments, as introduced by Weiser in [2].

The concept of ubiquitous computing refers to the cooperation of mobile and fixed network-enabled devices that collectively form an invisible pervasive computing and networking environment around the users, informing them, managing their communication needs, and performing various tasks on their behalf. In such an environment, the users cease to be aware of the presence of the devices and the networks, nonetheless they continue to require

increasing extensive applications and capabilities from these devices. Multiple constraints, such as mobility support, low manufacturing cost, and low energy consumption, make it unrealistic to assume that these devices will be capable of embedding local computing, networking, or storage resources to satisfy the increasing user needs.

In contrast, recent approaches [3] propose that autonomous cooperating service providers, at the edge of the network, offer these required resources to the devices. These resources are distributed among the service providers, and are accessible to the devices through elementary services, which are application elements executed on the computers of the service providers. As a consequence, the devices (also referred to as end-systems) view the network as both a communication medium and an endless source of usable distributed services, which they can compose to form complex composite services. Thus a composite service is a distributed application, i.e. an application that is composed of multiple scattered application elements. Such an approach reduces the complexity, the manufacturing cost and the energy consumption of the mobile devices, but raises multiple new challenges.

Examples of these challenges include the discovery of these elementary services by the devices, the planning of a composition of these elementary services, the implementation of this composition, and the assurance that the provided composite service delivers a sufficient quality for the user. Indeed in a particular instance of a distributed application, i.e. a composite service, the elementary constituent applications will create, consume, or process multimedia flows. In this case, the quality of the distributed application, as perceived by the user, depends on the management of these flows by the elementary components, and is sensitive to perturbations on the communication path between these components. Examples of such perturbations are the loss of a frame in a video presentation or the delay of an answer in a voice call.

This later challenge is the focus of this thesis. More specifically, this thesis focuses on the problem of providing quality of service (QoS) to the users of distributed applications or composite services.

2.2 CONTRIBUTIONS OF THIS WORK

As introduced previously, the QoS on the connections between the components of a distributed application determines to a great extent the QoS perceived by the users of this application. Therefore, this thesis concentrates on the provision of QoS on the paths between the elementary application components that form a distributed application. Since these components are scattered on the Internet, their paths are Internet paths.

Many contributions have been presented to address the issue of providing QoS on Internet paths. Chapter 4 summarizes and discusses some of these contributions. This work proposes an original recursive approach to address this challenge. As described in the previous section, a distributed application provides a complex composite service to the users by collating elementary service components from independent service providers. The proposed original approach also considers QoS management and control as an elementary service, which i) is deployed between two elementary application components, and ii) can be composed to provide global QoS to a given distributed application. In the proposed approach,

such an elementary QoS service is based on the selection, management and composition of successive independent Internet paths. This composition creates an alternate Internet path, which is different than the default path given by the Internet routing mechanisms. This approach takes advantage of the diversity of Internet paths between two Internet hosts [4]. Since the service providers are autonomous and on different locations on the network, the discovery and composition of these independent Internet paths have to be performed in a distributed manner.

This thesis asserts and shows that it is possible to provide QoS to a distributed application by using alternate Internet paths resulting from the compositions of independent consecutive paths. Moreover, this thesis postulates and demonstrates that it is possible to discover and compose these independent paths in a distributed manner within a community comprising a limited large number of autonomous cooperating peers, such as the fore-mentioned service providers.

To support this thesis, this dissertation makes the following contributions:

- first, this dissertation presents a description and a comprehensive characteristic analysis of the fore-mentioned composite alternate Internet paths. This analysis complements and extends the work presented in previous studies [4, 5, 6]. It indicates that within the Internet there exist many alternate paths that provide significant benefits to the data packets that use them, compared to the default Internet paths. Examples of such benefits are significant lower packet delays and packet loss rates. An alternate paths with such properties is termed a QEAP (QoS Enhanced Alternate Path). The results of this initial analysis motivate the use of QEAPs to provide enhanced QoS between two distant applications on the Internet.

- second, this work proposes an original architecture, termed SPAD (Super-Peer based Alternate path Discovery), which allows the discovery and selection of QEAPs in a distributed manner. SPAD is built on a community of cooperative end-points[1] grouped as an unstructured Super-Peer network [7]. These end-points use the distributed mechanisms provided by SPAD to search for, select, construct and utilize QEAPs. SPAD is a fully distributed system with no single point of failure, where each end-point cooperates as an equal peer. Thus, SPAD empowers the end-users at the edge of the network, allowing them to directly discover and utilize QEAPs, without having to rely on any central entity or third-parties at the Internet Service Provider (ISP) level. Moreover, SPAD is designed to be part of a middleware framework, which resides on the end-point machines. Therefore, it can be easily and incrementally deployed, as it does not require any change to the current Internet routing mechanisms, or core network components, such as routers inside Autonomous Systems (ASes).

  Through its novel approach to the discovery, the selection, and the use of QEAPs, SPAD enables the provision of QoS on the connections between the elementary application components that form a distributed application.

[1] These end-points are the service providers that host and offer elementary services in the form of application elements.

- finally, this dissertation present an extensive performance evaluation of SPAD. This evaluation is based on simulation experiments that use measured data from subsets of the Internet. It indicates that SPAD succeeds at discovering and selecting QEAPs within a limited large community of cooperating peers, on the order of 110 peers.

2.3 DISSERTATION OVERVIEW

The remainder of this thesis dissertation is organized as follows.

Chapter 3 presents the concept of *Service Composition*, and *Overlay Network*, which are the foundations of the approach "the network as an endless source of services", as introduced in section 2.1. It then discusses the challenges and some of the previous contributions that are related to these concepts. This chapter outlines the open issue of providing QoS to composite services, i.e. distributed applications, which motivates this thesis.

Chapter 4 introduces the notion of Quality of Service (QoS) from the perspective of the user, the application, and the network. It then presents the classic approaches for QoS provision on the Internet, and examines previous proposals that follow these approaches. Finally, it describes and discusses the original recursive approach of this thesis to address the issue of QoS provision for distributed applications, as introduced in section 2.1.

Chapter 5 presents the first contribution of this work. This chapter states the case for the use of QEAPs to provide QoS to distributed applications. To support this argument, it presents a comprehensive characteristic analysis of QEAPs, using measured data from subsets of the Internet. This analysis demonstrates that QEAPs do exist, and do provide significant durable benefits compared to default Internet paths.

Chapter 6 describes the second contribution of this work, namely SPAD, an original distributed QEAP discovery architecture. This chapter starts with a description of a preliminary proposal of a distributed QEAP discovery scheme. SPAD capitalizes on this preliminary scheme, using it as a base for its alternate path discovery functions. Chapter 6 proceeds with a design descriptions of the SPAD architecture. It then describes in details the SPAD functions and mechanisms that allow the distributed discovery and selection of QEAPs. This chapter ends with a discussion on the remaining SPAD functions that are not related to the discovery or selection of alternate paths.

Chapter 7 provides an extensive performance evaluation of the QEAP discovery and selection schemes from the previous chapter. It begins with the evaluation of the fore-mentioned preliminary scheme. It continues with the evaluation of the complete SPAD QEAP discovery and selection schemes, and finishes with a brief discussion on the performed evaluation experiments.

Last, chapter 8 concludes this dissertation, and outlines some future research directions.

Chapter 3

The Network, an Endless Source of Services

3.1 INTRODUCTION

As introduced in chapter 2.1, there is a dramatic increase in the use of mobile network-enabled devices, and their users constantly require more functionalities while expecting lower cost, longer battery life, and seamless mobility. To address this challenge, several contributions (such as [3, 17]) propose a model where a community of service providers offer elementary service components to the devices (and their users). These service providers are distributed over the network, i.e. the Internet, and the elementary service components are building blocks that the devices can select and compose into multiple new services to satisfy any potential user requirements. In such a model, the network evolves from its historical role as a communication medium to become an endless source of services.

Compared to a classic approach where the device directly provides the required functionality to the user, this model provides multiple benefits. For example, it removes the need for local intensive computation and large storage on the device, hence reducing the manufacturing cost associated with embedding high capability CPU (Central Processor Unit), and reducing the energy consumption required to power potential hard drives.

This model of the network as an endless source of services is founded on the notion of *Service Composition*, and its implementation within the current Internet is based on the notion of *Overlay Networks*. This chapter introduces these two concepts, provides an overview of the related contributions, discusses the associated challenges, and outlines the particular open issue that motivates this research work.

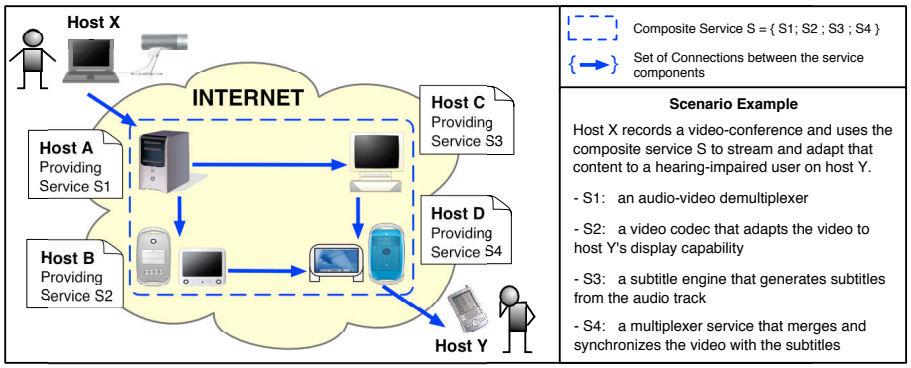


Figure 3.1 Deployment of a Composite Service S between hosts X and Y.

3.2 THE CONCEPT OF SERVICE COMPOSITION

3.2.1 What is Service Composition ?

A *service* is a function that is performed for a given customer. This function may accept or produce inputs or outputs, which could be data units or continuous streams. An example of an input data unit is a destination name in a service that provides plane ticket booking. An example of an output continuous stream is a AAC (Advanced Audio Coding) audio flow in a service that provides raw-to-AAC adaptation. The customer of a service is either a user/device or another service (i.e machine-to-machine interaction). A *service provider* is an entity that offers services to customers. It manages a set of machines that host software applications, which implement the offered services. The customers access these services via the network, i.e. the Internet. A service provider machine is usually located at the edge of the network, i.e. it is an end-point host, as opposed to a router inside an ISP, which is in the core of the network. However, it is possible that a service provider contracts an agreement with its ISP to have its machines located within the ISP's core network.

Service Composition refers to a computing model where new services are built by assembling a set of already existing elementary services, following a particular layout. Such a new service is termed a *composite service*, and the constituent elementary services are termed *service components*. This model allows the re-usability of already developed services, and reduces the development cost, time, and complexity that are associated with the design, the implementation, and troubleshooting of new services. Figure 3.1 illustrates the deployment of a composite service S between hosts X and Y. S is composed of the set of service components $\{S_i\}_{i=[1;4]}$, which are provided by the cooperating service providers A, B, C, and D respectively. Figure 3.1 also provides an example of a possible scenario that uses the concept of service composition.

There are two types of composition of services, namely static and dynamic. In a static service composition, the service providers or some third-parties (i.e. brokers) design and construct some composite services prior to receiving any user requests. These composite services are then accessible to the users in the same manner as a service component. Such an

approach leads to tailored optimized composite services. However, it is not able to process a user's need for a non existing particular service. Dynamic service composition allows the construction of new services *on-the-fly*. The service providers, some third-parties, or directly the devices plan and implement a new composite service for each user request. This type of composition is flexible to the user's need, and allows the provision of any type of requested services. The deployment of static or dynamic service composition on the current Internet raises multiple challenges, as described in the next subsection.

3.2.2 Challenges and Related Work

Service Description

A service needs to have a formal description, namely a service descriptor, to present to potential users that would access and use it as a standalone service or as a component of a composite service. A service descriptor contains two type of attributes, namely syntactic and semantic attributes, that provide information on the service. Examples of syntactic attributes are the data definition (e.g. type, range, rate) of any eventual inputs/outputs, and the access interface to the service (e.g. port number on the host machine, protocol to use). Examples of semantic attributes are the service performed functionality, its performance claims, its possible rating compared to other similar services. Semantic attributes are used during the specification and the planning of a composite service to design the required layout of service components and to discover/select the potential usable service components. While syntactic attributes are used during the planning and the implementation of a composite service to ensure the interoperability of the involved service components.

There already exist standard languages, such as XML (eXtensible Markup Language) [18] and SDP (Session Description Protocol) [19], capable of describing simple or complex syntactic attributes of a service. Whereas, there is not yet a standard language capable of describing complex machine-processable semantic attributes. But, several contributions in the *Semantic Web* research area have proposed and are working on such languages [20].

The Resource Description Framework (RDF) [21] language and its extension RDF Schema (RDFS) [22] from the World Wide Web Consortium (W3C) are the first standard attempts at providing a semantic-capable language to describe any Web resources. RDF and RDFS use the XML syntax. To semantically describe a resource, they provide a 3-tuple data model (object; attribute; value), some structured types (e.g. containers), and an object-oriented type system (with concepts such as class, and inheritance). However, RDF and RDFS lack several features, such as primitive data types, support of variables, negation statements, and thus they are not capable of describing complex semantic information [20].

The DARPA Agent Markup Language (DAML) [23] and the Ontology Inference Layer (OIL) [24] both builds on top of RDFS to provide a more expressive semantic-capable description language. DAML is a project funded by the government of the United States of America, while OIL is project funded by the European Union IST program (Information Society Technologies). Both languages extend the semantic capability of RDFS with features such as the support for complex inheritance relationships between classes (e.g. union, intersection), rich description of property constraints (e.g. domain, range, cardinality), and

complex relation properties (e.g. transitive, inverse). The latest versions of DAML, OWL (Web Ontology Language, formerly DAML+OIL) [25] aims at unifying both languages.

DAML-S [26] is an extension to OWL that defines a specific ontology to describe the properties and capabilities of Web-Services[1]. DAML-S only allows the semantic description of a Web-Service, and needs to be associated with another language, such as WSDL (Web Service Description Language) [28], which allows in turn the syntactic description of a service (i.e. description of the messages and the protocols used by the service). Currently, this association of DAML-S and WSDL represents the most elaborate solution to the problem of describing syntactic and semantic attributes of Web-Services. However, further investigations are required to assert their applicability to the description of generic service components, as described in section 3.2.1.

Service Discovery

In the service composition model, as illustrated in figure 3.1, service components are distributed over several service providers, which in turn are scattered on different location of the Internet. Therefore, a user needs a service discovery scheme to locate the particular service components that are required to build a given composite service. Conversely, service providers also requires a service discovery system to advertise their available service components. Thus, the service discovery scheme has a key role in the service composition model, and should have *good* properties such as: scalability, availability, accuracy in locating services matching any given query, and user privacy. Previous contributions have proposed two type of service discovery architectures, namely hierarchical and peer-to-peer based. This subsection reviews some of the proposed schemes within these two categories.

Service Location Protocol (SLP, and its wide-area extension WASRV) [29, 30], Service Discovery Service (SDS) [31], and the Universal Description Discovery and Integration protocol (UDDI) [32] are examples of hierarchical service discovery schemes. These three contributions propose an architecture of distributed hierarchically organized directory servers. These directory servers store service descriptions that are published by the service providers. They also accept, process, and forward user requests for particular services, and return back to the user either the descriptions or the locations of any matching services. SLP/WASRV and SDS use soft-state with periodic multicast announcements to store the service descriptions, thus providing tolerance to both server and service provider failures. SDS features secure authentication and communication schemes, which offer privacy and integrity guarantees to the users. UDDI is designed for Web-Services and has the support of major industrial companies (e.g. IBM, Microsoft, Oracle), which sponsor its deployment in the current Internet by hosting UDDI servers, and providing UDDI software implementations [33]. Although these schemes are designed for wide-area networks, their organization around centralized server (maintained by a limited set of administrative entities) still limits their potential scalability, i.e. their complexity remains linear with the number of servers.

[1] Web-Services are service components, as described in section 3.2.1, that are accessible only via Web protocols, such as SOAP (Simple Object Access Protocol) [27]. Thus they constitute a subset of the services that are considered in this thesis.

Open Distributed Service Directory (OpenDir) [34], SpiderNet [35], and the Universal Ring [36] are examples of service discovery schemes based on peer-to-peer systems[2]. These three proposals utilize a Distributed Hash Table (DHT), such as CHORD [37], CAN [38], and Pastry [39], to store the service descriptions. A DHT is a virtual structured key-space. Some given hash functions map the peer node IDs and the service descriptions into keys in this virtual key-space. A node stores the service descriptions that have their keys closest to its own key. The DHT nodes execute specific distributed algorithms to forward users requests for particular services to the nodes responsible for the corresponding keys. These search algorithms are more scalable than the search schemes of the previously described hierarchical architectures, e.g. search complexity is logarithmic for Chord and Pastry. However, these DHT based service discovery systems do not support complex queries [40], e.g. queries involving multiple keywords with boolean operators such as AND, NOT, OR. Furthermore, their forwarding schemes are vulnerable to misbehaving nodes that can redirect or drop a particular user query [41].

Each of these two categories of service discovery scheme has its advantages and drawbacks. In addition, they both share some common unresolved issues, such as the support of sudden short-lived popularity increase for a given service, i.e. *flash crowd* effect. Thus, designing a large-scale distributed service discovery scheme with the previously mentioned properties remains an open research topic.

Planning and Implementing a Composite Service

When a user requires a functionality that is not provided by any existing services, she/he initiates a request for a composite service with the required functionality. Designing this composite service is the first stage of the service composition process. The result of this task is a composite service specification, which provides the list of the required service components, and the specific layout for organizing them. The second stage of the composition process is the implementation of this specification, it includes subtasks such as the initialization and execution of the service components, the provision of the required resources, the monitoring of the service execution. There are several approaches to these two phases.

The ICEBERG architecture [42] uses XML to describe services, and introduces the notion of *Path*, i.e. a sequence of operators (i.e. service components) and connectors between them, to define a composite service specification. The ICEBERG *Path* provides only the sequential composition capability, thus it does not support composition scenarios with parallel service components, such as the example in figure 3.1. In the ICEBERG architecture, an Automatic Creation Path (APC) entity resides on each network domain, and handles *Path* creations and implementation/execution for the users on that domain. To provide high availability and failure resilience, APCs are executed on cluster platforms from the NINJA project [43]. Deploying such a cluster platform in every domain participating to a service composition system requires significant financial support. Furthermore, the APC in [42] is not fully automated, and only creates pre-specified composite services.

[2]Section 4.4.3 presents more details on peer-to-peer systems

SAHARA [3] proposes two alternatives to the service composition planning: a cooperative model and a brokered model. In the first model, the service providers collaborate to plan and implement a composite service. This model provides inherent resilience to single point of failure and load distribution, but requires effective trust mechanisms between service providers to guarantee fairness and composite service integrity. In the second model, broker entities interact with the service providers on behalf of the users, and provide them with the requested composite services. This model ensure that a unique entity is responsible for a given composite service, hence removing the need of service provider trust mechanisms. However a given broker does not have the complete knowledge of all the available services, and might select sub-optimal service components to create the requested composite services.

SpiderNet [35] proposes a decentralized service composition scheme. In this scheme, a user builds a graph of the required functions for a given composite service, and sends it to an initial set of service providers, using probe messages. The service providers analyze the function graph to determine if they can provide any service components which can perform the required functions. Then they forward the probes to other service providers, which recursively execute the same task until all the required functions have an assigned service component. The number of probes for a given composite service is bounded to control the forwarding overhead. This approach results in multiple possible layouts for a given composite service, SpiderNet provides a selection scheme to select the optimal layout to use based on some user requirements.

In the Web-Service context, [44] proposes a solution, where a reasoning capable broker assists the user in planning the composite service. This broker consists of a composer coupled with an inference engine. The user selects and organizes the desired service components using the composer, which provides a list of available services. After each user's action, the inference engine applies built-in axioms, matching and filtering rules on a service description Knowledge Base to update the list of possible components for the next action, thus assisting the user's decision process. The user selects the next component to add into the composite service from this updated list. In a similar approach, [45] proposes a task-planning capable broker. This broker is essentially an inference engine with a Knowledge Base of available service components. It takes a composite service specification in DAML-S and WSDL, and compiles it to produce a Structural Synthesis of Program (SSP) [45]. From the SSP, the inference engine builds a sequence of service components that implements the requested composite service. This SSP sequence is then compiled back to a DAML-S specification and returned to the user. This composition planning scheme is completely automated, but only support sequential composition of simple service components.

Quality of Service for a Composite Service

As discussed in chapter 2.1, some instance of composite services will generate, or process some continuous multimedia flows, such as a video stream. In such cases, the quality of the composite service, as perceived by the user, depends on the management of these flows by the service components, and on the provision of sufficient resources (i.e. QoS provision) on the communication paths between these components. Thus, the global QoS of the composite service not only depends on the QoS of its components, but also on the QoS on

the logical links between these components. This subsection reviews some of the proposed solutions to the problem of QoS provision and management in service composition.

In [46], the authors propose an architecture, where network domains (such as an ISP) are aggregated into logical domains (LD). A Clearing House (CH) entity is responsible for QoS provision and management on the links within each LD. When a CH receives the QoS specifications for a continuous stream between two service components on different service providers within its LD, it computes the optimal path to connect the involved service providers, and performs the necessary resource reservation. To increase scalability, the CH performs resource reservations on aggregates of streams. Furthermore, to allow QoS provision across different LDs, the LDs are recursively aggregated into larger LDs with their own CHs. This result in a hierarchical tree structure, where parent CHs are responsible for QoS provision and management between their child LDs.

SpiderNet [35] proposes to consider QoS management at the composite service planning phase. The probe message with the function graph for a composite service also includes the QoS requirements corresponding to each function. The service providers on the probe path use these requirements to pre-select the service components to include in the possible composite service layouts that are returned to the source of the probe message. The SpiderNet selection scheme further uses the specified QoS requirements to perform the final selection of the composite service layout to implement.

3.2.3 Summary

This section 3.2 presented the concept of Service Composition and the principal related challenges. As described in section 3.2.2 several proposals aim at addressing the issues of service description, service discovery, and composite service planning/implementation. However at the time of this service composition literature review, the two propositions [46] and [35] were the only noticeable contributions that aimed at providing and managing QoS within a composite service. Thus, few contributions focus on the issue of Quality of Service for a particular instance of composite service, and more specifically on the issue of QoS provision on the communication paths between the service components that constitute this composite service.

This observation motivated the research work performed within this thesis. The resulting contributions propose a complete architecture that aims at providing enhanced Quality of Service to the users of composite services.

3.3 THE CONCEPT OF OVERLAY NETWORKS

3.3.1 What is an Overlay Network ?

An overlay network is a virtual network that is built over an existing underlying network. A subset of selected nodes from the underlying infrastructure forms the nodes of the overlay network. A set of logical links connect the nodes on the overlay network nodes. A given logical link is mapped to a concrete path on the underlay network, i.e. it is a sequential

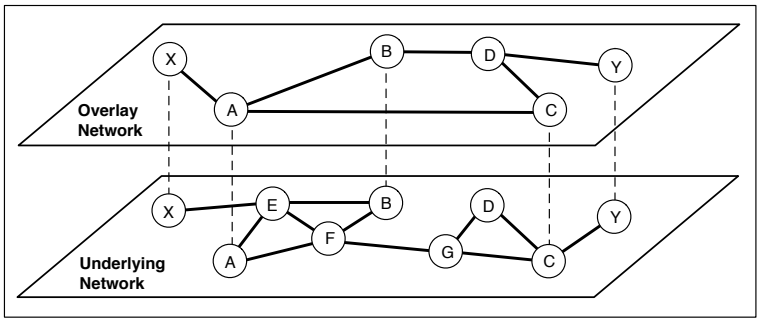


Figure 3.2 Example of an Overlay Network.

composition of links on the underlay network. Figure 3.2 presents an example of overlay network.

An overlay network adds a layer of abstraction between the underlying network and the end-users. This concept provides a substrate to deploy or experiment new architectures and functionalities over an existing network, without requiring any expensive or major infrastructural changes on this network. Examples of such functionalities include multicast communication [47], QoS provision [48], file-sharing [49], content distribution [50], anonymous information publication [51], and service composition, as described previously.

Within a service composition system, the users and the service providers are distributed over the Internet and interact with each other, thus implicitly forming an overlay network. Furthermore within a given composite service, the distributed service components cooperate and communicate with each other via logical links, and therefore also form a dynamic short-lived isolated overlay network. For example, the overlay network from figure 3.2 is a possible substrate for the composite service from figure 3.1.

Following these considerations, this section introduces the term *overlay application* to refer to a distributed application, i.e. an *overlay application* is a distributed application, which is composed of scattered independent application elements. The interconnection of these application elements forms an overlay network. Moreover, this section also introduces the term *overlay association* to refer to the set of connections within an overlay application, i.e. an *overlay association* is the set of paths between the application elements that form a distributed application. Therefore, a composite service is an instance of an *overlay application*, and the set of connections between its service components is an instance of an *overlay association*.

3.3.2 Related Work: Overlay Networks on the current Internet

Content Distribution and File-Sharing are notorious examples of distributed applications based on an overlay network. The Akamai [50] content distribution service deploys an overlay network with nodes in several thousand of geographic and network locations [52]. These nodes are servers that cache contents, such as images and audio streams, from Akamai's customers (e.g. Reuters, Google, Apple Computer). A user in Australia that

browses the website of an Akamai's customer in Europe, receives some of the website content from a nearby Akamai server, which is potentially located in Australia. This service reduces the content access latency, and increases its availability, thus enhancing the user's browsing experience. Gnutella [49] and BitTorrent [53] are examples of file-sharing distributed applications that deploy dynamic overlay networks of more than 5.10^4 nodes on the Internet [54]. These nodes execute particular distributed protocols to share files with each other.

Large scale information publication and dissemination systems, such as Scribe [55] and FreeNet [51], also use the concept of overlay network to deploy their services over the Internet. Scribe is a generic and scalable group communication and event notification system that is built over the Pastry [39] DHT architecture. It deploys an overlay network, where nodes can create, join, and send messages to communication groups. FreeNet is a distributed information storage system that deploys an overlay network where each node can anonymously publish and retrieve information, thus avoiding potential censorship and persecution.

Although some routers within the current Internet implement IP Multicast [56], several drawbacks, such as increasing router state complexity, have prevented multicast to be widely deployed at the IP layer [57]. In contrast, application-layer multicast presents significant advantages, such as no router support or global group identifier (e.g. IP multicast address) requirements. Overlay networks are the substrate of several contributions that provide such an application-layer multicast service. Examples of these contributions include End System Multicast [47], Overcast [58], and PeerCast [59].

Similar to the deployment of multicast service at the IP layer, classic approaches to QoS provision at the IP layer also suffer from several drawbacks, as described in chapter 4. Recent contributions (e.g. OverQoS [48], QRON [60]) use the concept of overlay network to provide QoS above the underlying IP network. Chapter 4 presents and discusses these contributions in more detail.

Last, the PlanetLab project [61] uses the overlay network abstraction to provide a virtual controlled wide-area testbed to experiment new technologies. PlanetLab currently deploys an overlay network over 336 different sites with a total of 698 nodes. Many research projects use this testbed to evaluate their system proposals in a wide-area environment [62]. The PlanetLab nodes are time-shared servers, which can: i) execute multiple experimental applications, and ii) be dynamically interconnected in multiple overlay topologies, each corresponding to a particular application.

3.3.3 Overlay Network Challenges

As described in the previous subsection, many contributions successfully use the overlay network concept as a mean to introduce or experiment new services and technologies on the current Internet. The increasing number of these deployed overlay networks raises some issues that require further research investigation.

As described in [63], current overlay networks implicitly assume that they are the sole user of the underlying infrastructure, i.e. they assume that no other overlay network concurrently uses the same underlying network services. With the increasing number of

overlay networks proposals over the IP infrastructure, this assumption no longer holds. Therefore, multiple independent overlay networks need to coexist over the IP network, and equally share the network, computation, and storage resources provided by the potential overlay nodes. Implementing such coexistence is an important challenge. Contributions, such as the X-Bone [64], propose some initial schemes to address this challenge.

The topology of the overlay network has an impact on the performance of the new service that it offers. For example, Li and Mohapatra [65] demonstrate that overlay forwarding strategies (e.g. link-state based and feedback based) have significantly different performance results depending on the considered overlay topology (e.g. full-mesh, k-spanning tree, adjacent connection). Fan and Ammar [66] further demonstrate that the optimal overlay topology depends on the nature of the communication pattern between the overlay nodes. In a dynamic environment with evolving communication patterns, they propose a set of reconfiguration policies that allows the adaptation of the overlay topology to the changing communication patterns. These works are some of the initial contributions that address the challenge of overlay network topology configuration, which requires further investigations.

3.3.4 Summary

This section 3.3 introduced the concept of Overlay Network. This concept allows the deployment or the evaluation of new architectures and functionalities over an existing network infrastructure. This section further gave some examples of such use on the current Internet, and briefly introduced some related challenges. A Service Composition system is another example of such a new architecture that use the concept of Overlay Network as a deployment substrate. A composite service, as described in section 3.2.1, is essentially a distributed application that builds an overlay network, which connects independent application components. Thus this section introduced the terms *overlay application* and *overlay association*, to refer to a composite service and its related overlay network respectively.

3.4 CHAPTER SUMMARY

This chapter introduced and discussed the two concepts of Service Composition and Overlay Networks. The notion of Service Composition enables the "Network as an endless source of services" computing model, as introduced in chapter 2.1. Following this model, a mobile network-enabled device requires minimal embedded resources or functionalities. It relies instead on the network to provide it with several elementary service components, which it can compose to potentially obtain any required resources or functionalities.

The notion of Overlay Network provides an appropriate substrate for the deployment of a Service Composition system on the current Internet infrastructure. An overlay network is a virtual network built over an existing network infrastructure. A Service Composition system with its community of users and service providers is an overlay network. A particular instance of a service composite also forms a dynamic short-lived overlay network.

In section 3.2.3, this chapter concludes that the provision of QoS for a composite service remains an open issue. Furthermore, since a composite service deploys an overlay network,

the QoS on this composite service depends on the QoS between the nodes of this overlay network. Therefore this thesis aims at providing QoS to the users of any composite services, and more specifically it aims at providing enhanced QoS between the nodes of any dynamic overlay networks.

Chapter 4

Quality of Service for Distributed Applications

4.1 INTRODUCTION

As presented previously, the service components within an overlay application may produce, consume and exchange time-constrained multimedia contents, such as a continuous audio stream. The user quality perception of the overlay application depends on the management and the transmission of these contents between the service components. This thesis aims at enhancing the quality of these transmissions, i.e. at providing QoS to the transmissions of multimedia contents between end-points within an overlay application.

Depending on the considered perspective, the term *Quality of Service* (QoS) refers to different notions. This chapter introduces the notions of QoS from the perspective of the user, the application, and the network. It then describes the classic approaches to the provision of enhanced QoS between two Internet end-points, and discusses previous proposals that follow these approaches, outlining their benefits and drawbacks.

This chapter then presents the alternate approach taken in this thesis. A given overlay application provides a composite service to the users by collating independent elementary service components. As introduced in section 2.2, this thesis recursively applies this pattern to provide enhanced QoS to the overlay application by composing QoS-enhanced virtual paths between the nodes of its overlay association. Applying this pattern further, this thesis proposes to construct each of these QoS-enhanced virtual paths by assembling successive independent Internet path, thus creating a composite alternate Internet path that is different than the default paths given by the Internet routing mechanisms [4].

Last, this chapter discusses the challenge in discovering these virtual alternate Internet paths in a distributed manner, reviews some related contributions, and introduces the virtual path discovery and selection architecture proposed in this thesis.

4.2 THE NOTION OF QUALITY OF SERVICE

The ITU-T Recommendation X.902 [67] defines Quality of Service as the "set of quality requirements on the collective behavior of one or more objects". Vogel et al. [68] further refine that definition as "the set of those technical and other parameters of a distributed multimedia system, which influence the presentation of multimedia data to the user, and in general the user's general satisfaction with the application". Finally, Ferguson and Huston [69] defines QoS as "a method to provide preferential treatment to some arbitrary amount of network traffic, as opposed to all traffic being treated as best-effort, and in providing such preferential treatment, attempting to increase the quality level of one or more of these basic metrics for this particular category of traffic", with the basic metrics being delay, jitter, bandwidth and reliability.

These definitions highlight the multiple meanings given to the term QoS in the literature [70]. Following these definitions, this section introduces the notion of QoS from three complementary perspectives.

4.2.1 QoS from the user perspective

The user's perception of the quality of a given service is subjective and depends on multiple factors [71, 72]. Examples of these factors include the importance and the purpose of the service to the user, the type of multimedia content generated by the service, the potential cost of the service, and the previous personal experience of the user with similar services. Thus a given user could be satisfied by the service delivered by a given instance of an overlay application, while another user could consider the same service as performing under his/her quality expectations. Given these considerations, the user-level description of QoS requirements for a particular service is based on qualitative parameters reflecting the user's preferences, such as "excellent", "poor", "urgent", or "slow".

4.2.2 QoS from the application perspective

From the application perspective, the quality of the service (which is provided to the user) is defined by a set of qualitative and quantitative constraints for each media contents that it delivers to the user. These constraints depend on the type of these media contents, and the capability of the device hosting the application. The ITU-T Recommendations F.700 [73] and G.1010 [74] provide a classification of such constraints. For example, these recommendations state that the minimum QoS constraints for a mobile video-conference service are: i) an audio stream with a 7kHz speech quality, less than 3% of information loss, ii) a video stream in a QCIF format (144x176 pixels) with less than 1% of information loss, and iii) both streams with a target one-way delay of 150ms, a synchronization within 80ms, and a jitter lower than 1ms. Other contributions propose schemes to describe some of these constraints, such as the RTP Profiles (Real-time Transport Protocol) [75], which allow the description of some qualitative constraints.

4.2.3 QoS from the network perspective

From the network perspective, the QoS provided to a given application refers to a set of characteristic guarantees for the network paths that convey each data flow of that application. These characteristics are qualitative, such as ordered and/or secured (i.e. data integrity) delivery, and quantitative, such as bandwidth, latency, and packet-loss. Since a network path is composed of multiple components (e.g. routers) and links, the QoS characteristics of a given flow on such a path is an aggregate of the characteristics of the corresponding network links. The Integrated Service (IntServ) architecture, as described in the next section 4.3.1, provides a scheme (the Traffic Specification TSPEC field) to describe some network-level QoS characteristics of a data flow [76].

4.2.4 Mapping QoS requirements across these different perspectives

As stated in [68], the human user is the starting point of the overall QoS consideration. The user requires a functionality from an application (e.g. video-conferencing) with some qualitative user-level QoS specifications. Given these specifications and the capabilities of its host device, the application then produces a corresponding application-level QoS specification, such as the example in the previous subsection 4.2.2. These application-level specifications are then used to derive the network-level QoS requirements for each data flow of the application. Such a mapping process of QoS requirements from the user-level down to the network-level is a complex task [77]. Within the Open Systems Interconnection (OSI) layered model [78] of the network stack, the Transport layer which is located between the Application and the Network layers, is an adequate point to perform this QoS mapping process.

Exposito et al. [79] propose an XML-based specification language (XQoS) and a methodology to perform such an optimal QoS mapping. In contrast, Ardon et al. [71] propose a reactive approach where a software agent assists the user in dynamically (re)defining the application-level and network-level QoS specifications according to its current experience of the service. Other contributions, such as [80], propose extensions to the Session Description Protocol (SDP) [19] and the Session Initiation Protocol [81], which allows the description and the mapping of application level media stream requirements during the setup of multimedia session/call between two parties.

4.3 CLASSIC APPROACHES TO QOS PROVISION

Several contributions have proposed schemes to provide QoS between end-points on the Internet (i.e. *end-to-end QoS*). This section classifies and reviews some of these contributions into two categories, namely the network-level and the end-to-end/application-level.

4.3.1 Network Level

Network-level end-to-end QoS provision systems aim at adapting the network to the QoS requirements of the applications. Several contributions have proposed QoS-oriented routing

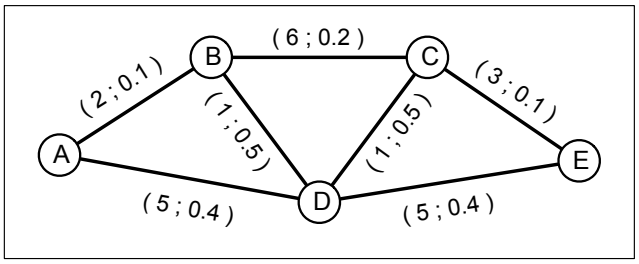


Figure 4.1 A network topology example to illustrate QoS routing.

schemes to perform such an adaptation. The Internet Engineering Task Force (IETF) has also proposed two different architecture models to provide end-to-end QoS on the Internet. This subsection reviews some of these network-level schemes.

QoS Routing Schemes

On the Internet, a data packet is transmitted from a host H_A to another host H_E via a sequential chain of routers. Each router regularly executes a common standard routing algorithm to build a routing table. Upon receiving an incoming data packet, a router consults its routing table to select the next router to forward the packet to. Currently within a network domain, this common routing algorithm aims at minimizing the number of hops between H_A and H_E. For example on figure 4.1, if H_A and H_E are connected to the routers A and E respectively, the current Internet routing algorithm selects the path $[A \rightarrow D \rightarrow E]$ (i.e 2 hops) for the transmission of a data packet from H_A to H_E.

However, one can design other types of routing algorithm that aim at optimizing one or a combination of QoS parameters, such as bandwidth, latency, jitter or packet-loss rate. Such an algorithm is a QoS-oriented routing algorithm. For example on figure 4.1, if the first weight parameter on each link represents the value of an additive QoS parameter (e.g. latency), then a QoS routing algorithm that aims at minimizing this parameter, will select the path $[A \rightarrow B \rightarrow D \rightarrow C \rightarrow E]$ to transmit a data packet from H_A to H_E. This path has a cost of 2+1+1+3=7, which is lower than the cost of $[A \rightarrow D \rightarrow E]$ (i.e. 5+5=10). Similarly, if the second weight parameter on each link represents a multiplicative QoS parameter (e.g. 1-loss probability), then another QoS routing algorithm might select the path $[A \rightarrow B \rightarrow C \rightarrow E]$ which has a cost of 0.1*0.2*0.1=0.002 (assuming no loops).

The optimization of only one QoS parameter has been extensively studied (e.g. Djikstra or Bellman-Ford algorithms). The current Internet routing is an example of such a single parameter QoS routing scheme, where the optimized additive parameter is the hop-count. Guerin and Orda [82] propose another example of QoS routing scheme, which optimizes either a single multiplicative or additive parameter.

Wang and Crowcroft [83] demonstrate that the problem of selecting multi-constraint paths with any combination of additive and/or multiplicative parameters is NP-complete. For example on figure 4.1 if the first and second weight parameters are additive and multiplica-

tive respectively, it is in general "difficult" to design an algorithm which manage to select the path that optimizes both weight parameters in a reasonable (i.e. polynomial) time. Wang and Crowcroft also demonstrate that this multi-constraint path selection problem can be solved only for the cases involving a combination of one additive/multiplicative parameter with a concave one (e.g. bandwidth). They propose an example of such a solution for the latency-bandwidth combination.

Several other contributions note that although the multi-constraint path selection is NP-complete, there exists many types of real-world network topology where this problem can be simplified. In these cases, these contributions propose simplification/approximation methods and/or heuristic algorithms to allow the selection of the path that best meets the given multiple constraints. Paul and Raghavan present a comprehensive survey of these contributions in [84].

By design, QoS routing schemes have to be implemented within the routers (i.e. *inside* the network) as a replacement or a complement to the current Internet routing scheme based on hop-count. However, two major factors prevent the deployment and use of these schemes. First, most ISPs are not inclined to integrate or activate new features in their routers. Indeed, such new features constitute potential sources of failure, and any network routing failure most probably leads to customer dissatisfaction and consequently to a loss of revenue. Second, the Internet now includes several thousands of ASes, which have in turns hundreds of routers, thus deploying a new routing scheme on such a large amount of machines involves several economic and technologic challenges. These two factors are parts of the causes that lead to the current Internet *ossification*, which has driven research in overlay-based routing schemes, as discussed in section 4.4.

The Integrated Service model (IntServ)

To guarantee some given QoS requirements for a data flow between two end-points (i.e. a sender and a receiver), the Integrated Service (IntServ) model [85] proposes an approach where the necessary network resources are reserved on each network elements (i.e. routers) on the data path. This resource reservation is done prior to the transmission of any data on the path. Consequently, the sender, the receiver, and every router on the data path have to support the IntServ model. The routers on the data path of a given flow derive the type and quantity of resources to reserve from the QoS characteristics of that flow. These characteristics are specified within a Traffic Specification attribute (TSPEC), using a token bucket model [76].

IntServ currently defines two different class of services for a given data flow[1]:

- Guaranteed-Service [86]: this service guarantees a delay bound and a minimum bandwidth to the packets related to the data flow .
- Controlled-Load [87]: this service guarantees that the majority of the packets related to the data flow will approximately experience the same QoS than a similar data

[1]These services apply only to the data flow packets that comply with the flow QoS specifications, as described in its TSPEC

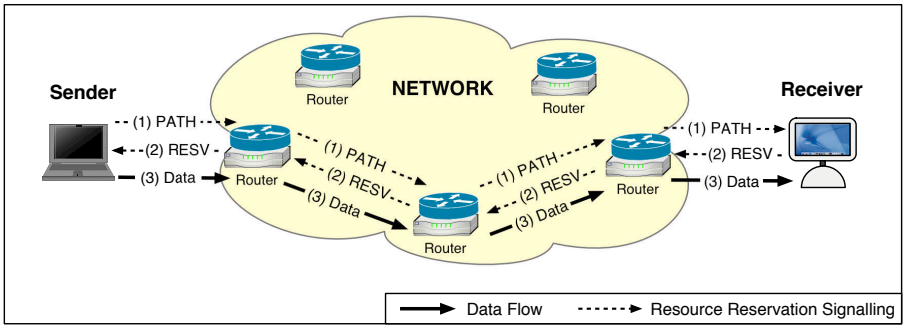


Figure 4.2 Resource Reservation within a Network implementing the IntServ model.

flow over a *best-effort* network under unloaded conditions. Thus, a minor acceptable number of packets may be lost or experience higher delay.

The IntServ resource reservation process uses the Resource ReSerVation Protocol (RSVP) [88], as illustrated in figure 4.2. The sender S initiates this process, and sends a RSVP PATH message (1) to the receiver R. The PATH message contains the QoS specification of S's data flow (TSPEC). The routers on the network path between S and R include their own resource capabilities within the PATH message (via the addition/modification of an ADSPEC attribute), and forward it. When R receives the PATH message, it builds a RESV reply message and sends it back (2) to S on the reverse path. This RESV message contains the QoS specifications RSPEC, which describes the resources that R would like to reserve for the reception of S's data flow. R constructs its RSPEC according to the TSPEC and ADSPEC from the incoming PATH message. On each router of the reverse path, an admission control function analyzes the RSPEC, and decides to accept or reject the reservation request, depending on the available resources. If the router accepts the reservation, it forwards the RESV message to the next router on the reverse path, and allocates the states and the necessary resources to forward S's data flow with the agreed QoS requirements. Otherwise, it discards the RESV message, thus aborting the resource reservation process. Once a reservation has been established, S starts the data flow transmission (3). IntServ routers associate timeout timers to the reservation states of accepted data flows (i.e. *soft states*), thus S and R periodically exchange PATH and RESV messages during the flow transmission to maintain the resource reservation.

Some studies demonstrate that the IntServ model manages to guarantee the QoS resources for accepted data flows on overloaded small scale networks [89]. However, the number of states to manage per flow, and the RSVP protocol complexity both limit the scalability of the IntServ model. Indeed, routers in the core Internet backbone have several thousands of flows to process at a given time, and would require significant processing and storage capabilities to implement this model. To mitigate this scalability issue, some contributions such as Baker et al. [90] propose to perform resource reservation per flow aggregates in the core routers.

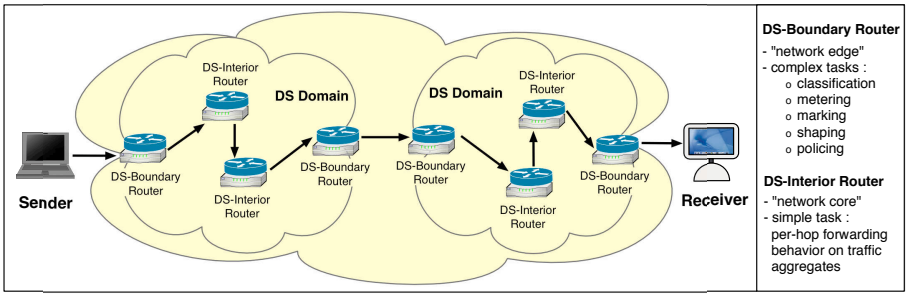


Figure 4.3 Transmission of a data flow across Networks implementing the DiffServ model.

The Differentiated Service model (DiffServ)

The Differentiated Service (DiffServ) model [91] addresses the scalability issue of IntServ by confining the complexity (i.e. packet classification functions, and traffic conditioning functions) to the nodes at the network boundaries, and leaving only a simple task (i.e. differentiated forwarding policies of traffic aggregate) to the nodes inside the network core.

In figure 4.3, a *DS Domain* represents an administrative network domain that implements the DiffServ model. A *DS Interior router* forwards data packets within a given domain, while a *DS Boundary router* receives/transmits data packets from/towards a customer or another domain. A sender S contracts a Service Level Agreement (SLA) with a DS domain D. This SLA defines both the specifications of the traffic that S agrees to send, and the QoS characteristics that D agrees to guarantee for that traffic. At the edge of D, a DS boundary router R receives S's incoming traffic, and applies a classification and a conditioning functions to the corresponding packets. Based on the contracted SLA, R assigns a Behavior Aggregate (BA) to these packets, and marks them accordingly, using the DS field in their IP header. Similarly, neighboring domains contracts inter-domain SLAs, and exchange traffics via their boundary routers. Within D, the DS interior routers use the DS field of a given packet to determine the type of forwarding behavior, i.e. Per-Hop Behavior (PHB), to apply to the packet depending on its assigned BA.

DiffServ currently defines two different classes of Per-Hop Behavior:

- Expedited Forwarding [92]: this service guarantees a low delay, a low loss rate and a low jitter to the packets that comply to the traffic specification as described in the contracted SLA.
- Assured Forwarding [93]: this service allows the prioritization of packets according to their compliance with the traffic specified in the contracted SLA. A non-compliant packet is marked with a lower priority than a compliant one. A DiffServ router forwards the packets according to their priority marks. When a congestion event occurs, the router first discards the packets with the lowest priority.

While the DiffServ model allows better scalability than the IntServ model, it still requires some modifications on the core network routers to allow them to process the DiffServ

marked packets. Furthermore, mapping the user end-to-end QoS requirements into the DiffServ per-hop forwarding services is a difficult task. Finally, the provision of such an end-to-end QoS requires that all the network domains involved in the end-to-end path implement the DiffServ model, and have contracted mutual inter-domain SLAs. These drawbacks and the lack of adequate economic incentives hamper the deployment of DiffServ on the current Internet [78]. Some contributions, such as [94], demonstrates that a DiffServ implementation with a correct set of traffic classification and conditioning functions manages to guarantee end-to-end QoS.

4.3.2 End-to-End and Application Level

The contributions within this category of end-to-end QoS provision systems aim at adapting the traffic and the behavior of the applications to the available network resources. This adaptation is performed either at the Application layer or at the Transport layer within the OSI layered model [78]. This subsection reviews some of the contributions within these two layers.

Adaptive Applications

An adaptive application monitors the available network resources and responds to eventual resource fluctuations by adapting its generated traffic in order to maintain an acceptable service quality to the user. For example, an audio-video conference application initially uses a video codec A for a call between two users. After determining a change in the available network resources, this application ensures that the call remains sufficiently interactive by switching to another video codec B, which requires less bandwidth at the price of a lower video quality.

The audio conferencing application Vat [95] is an example of adaptive application. Vat monitors the end-to-end delay between two users, and constantly estimate the related average and standard deviation. It uses these estimates to adapt the size of its audio playing buffer, thus reducing or suppressing any eventual jitter. The original INRIA Videoconferencing System (IVS) [96] follows a similar approach. The IVS sender-side regularly evaluates the available bandwidth and the video reception quality by polling the IVS receiver-side. When a significant bandwidth fluctuation occurs, the sender adapts the video compression ratio accordingly, possibly reducing the video quality to lower the amount of required bandwidth.

Other contributions propose complete application-level architectures to adapt multimedia contents to the available network and end-system resources. The Mobile Aware Server Architecture (MARCH) [97] is an example of such architecture. MARCH is a server-centric architecture which deploys proxies nodes at various point on the data path between a pair of communication end-systems. These proxies dynamically adapt the multimedia contents between the end-systems to correspond to the user preferences, the capability of the devices, and the available network resources.

Adaption at the Transport Layer

On the end-systems, the OSI model [78] locates the Transport layer between the Application and the Network layer. Thus the Transport layer is a vantage point for deploying mechanisms that mediate between the QoS requirements of an application and the available QoS resources from the network.

The Transport Control Protocol (TCP) [98] is one of the most used scheme at the Transport Layer. TCP is a connection-oriented scheme, which ensures some end-to-end QoS guarantees to the applications over the best-effort communication service provided by the Network layer. These QoS guarantees are reliable and ordered delivery of the application's data stream. To implement these guarantees, TCP uses a combination of acknowledgement, retransmission, timer management, and incremental sequence numbering mechanisms. While these two QoS guarantees were sufficient to earlier type of Internet traffic, they are not suited for the rapidly increasing multimedia type of traffic. Moreover, its Additive Increase Multiplicative Decrease (AIMD) congestion control mechanism prevents TCP from meeting the delay/jitter QoS constraints of time sensitive multimedia streams [99].

Other transport schemes, such as Stream Control Transmission Protocol (SCTP) [100] and Datagram Congestion Control Protocol (DCCP) [101], provide also some type of QoS guarantees to the application. SCTP is a connection-oriented scheme, which provides the following QoS guarantees to the application: reliable and ordered delivery, multi-stream delivery, and multi-homing delivery. DCCP is a datagram and connection-oriented transport protocol. It provides no reliability or ordered delivery, but allows transport entities to negotiate and select the given congestion control mechanism to use. TCP-Friendly Rate Control (TFRC) [102] is an example of such a congestion control mechanism. It is an equation-based mechanism for unicast flows. It guarantees that a given TFRC unicast flow fairly competes for bandwidth with coexisting TCP flows within a best-effort Internet environment.

These transport schemes (e.g. TCP, SCTP, DCCP) only provide a limited set of QoS guarantees to the applications, and do not support other QoS requirements, such bounded delay and jitter, or minimal throughput. Exposito et al. [103] propose a Fully Programmable Transport Protocol (FPTP) that supports the integration of these QoS requirements. FPTP is a modular transport system, which provides basic transport QoS-aware mechanisms, that can be configured and composed to implement different type of transport schemes depending on the QoS requirements of an application. For example, in the case of a MPEG video streaming application, FPTP can compose several transport mechanisms to provide a time-constrained and differentiated partially reliable service, which is not provided by any of the previous transport protocols.

4.4 ANOTHER APPROACH TO END-TO-END QOS PROVISION

The previous contributions (section 4.3) aim at addressing the issue of end-to-end QoS provision from the network, and the application/end-to-end perspectives. In addition to their individual limitations as described previously, they also share the following drawback:

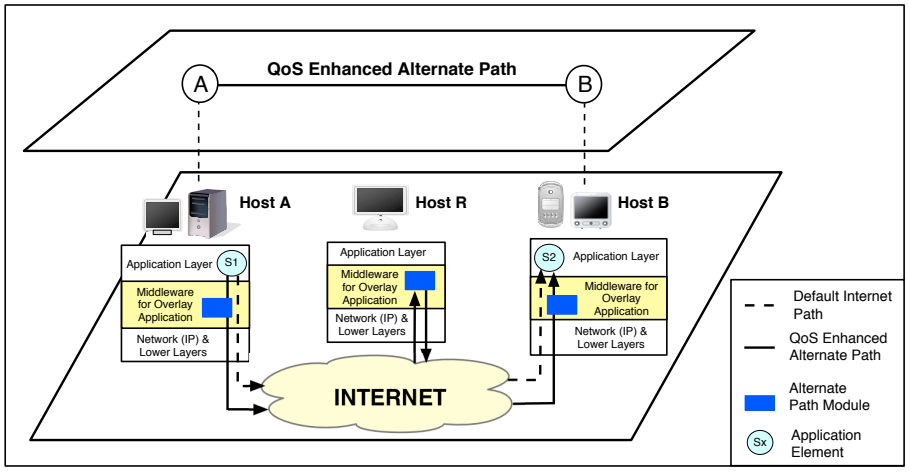


Figure 4.4 Comparison between a QoS enhanced alternate path (from host A to B, via a relay host R) and the default Internet path

they all rely on the end-to-end *default* path given by the network routing mechanisms to implement their QoS-oriented schemes. As described in chapter 5, in a significant number of cases this *default* path is not the optimal available path in terms of delay or reliability. This section present another approach to end-to-end QoS provision, which address this drawback. In most of the cases, this alternate approach is compatible with the other previously described contributions. For example, it can be included in the FPTP framework [103], and constitutes the basis of another transport module or micro-protocol, which can be part of a composite FPTP transport scheme.

4.4.1 Virtual Paths on Overlay Networks

Instead of adapting the network to the application requirements, or adapting the application behavior and its generated content to the network resources, this section propose to virtualize the network from the application's perspective.

As described in details in chapter 5, the Internet is composed of multiple interconnected independent network domains. The Internet Protocol (IP) and some standard routing protocols[2] are used to determine a particular route to connect any given hosts H_A and H_B across these domains. This particular route constitutes the default end-to-end Internet path between H_A and H_B. The proposed approach uses the overlay network model to create virtual paths over the Internet. Such a virtual path is composed of consecutive end-to-end paths over the IP network (i.e. the Internet). It forms a composite alternate Internet path that is different than the default Internet path. Figure 4.4 illustrates this approach. Providing end-to-end QoS with such a scheme is possible, only when these virtual paths offer better QoS characteristics (e.g. delay, loss rate, bandwidth) than the default path on the IP network. The next chapter provide thorough details on the characteristics and the origins of these virtual paths.

[2]More details in chapter 5.

From the network perspective, this approach does not require any modifications of network components such as routers. Therefore, it could be deployed on the current Internet without requiring administrative or financial support from the entities that manage these network components. Thus, it can potentially favor the emergence of a market of cooperating overlay-based QoS service providers. From the application perspective, this approach does not necessary require any software modification. It is potentially transparent for legacy applications, which are unaware of its deployment. They benefit from the advantages of any available virtual path, while their access to the network service remains unchanged. However, an applications can also be modified/created to be aware of the deployment of such an approach. In this case, the application can request virtual paths with specific QoS characteristics when available.

4.4.2 Related Work

Some former contributions have proposed systems that follow the virtual path approach, as described previously. This subsection presents and reviews some of these systems.

Savage, et al. [4] discussed the existence and benefits (i.e. enhanced or controlled QoS and robustness) of virtual paths between Internet hosts. Based on offline analysis of several end-to-end characteristic measurements for multiple hosts across North America, they found that for 80% of the node pairs there exists an alternate path that provides significantly superior QoS (such as lower delay or loss rate) than the default Internet path. They confirmed that this finding was independent of the considered time period, or QoS parameter (e.g. delay, loss). However, they do not provide any methods to dynamically discover and deploy these alternate paths on the current Internet.

The Resilient Overlay Network (RON) [104] is an application architecture that improves communication reliability between end-systems. This architecture uses virtual paths among RON nodes to recover from failures and QoS degradations on default Internet paths. It requires: i) the presence and management of dedicated RON nodes in different Internet routing domains (i.e. Autonomous System, AS), and ii) some agreements (similar to the SLAs from section 4.3.1) between the corresponding administrative entities. A given RON node uses a modified link state routing algorithm to discover any available virtual paths towards another RON node. Thus in a community of N RON nodes, each node has to maintain some information about N-1 other nodes. For these reasons, this architecture is not suitable for a large community of nodes, which are potentially spread across thousands of various Internet domains.

The QoS-aware Routing in Overlay Networks project (Q-RON) [60] proposes a unified framework to support the communications of emerging overlay applications, via the use of QoS enhanced virtual paths. The Q-RON architecture is based on hierarchically organized Overlay Brokers (OBs) located on different ASes. Similar to a RON node, each OB maintains some hierarchically aggregated information about the other OBs. Using this information, an OB has the choice between two modified link state algorithms to discover any available alternate paths. Third parties manage the OBs, they *buy* network resources from ASes, and *sell* them back to the users on end-systems, in the form of *added value* QoS enhanced alternate paths among the OBs. In contrast, this thesis proposes a different

architecture (chapter 6), which allows an end-system, e.g. a service provider as described in chapter 2.1, to directly discover QoS enhanced virtual alternate, and enjoys their benefits without paying any third-party broker.

OverQoS [48] uses the Controlled Loss Virtual Link (CLVL) abstraction to provide end-to-end QoS services. A CLVL is a virtual path between two OverQoS nodes, which reside on different participating ASes. An OverQoS node acts as virtual routers on the overlay network. It aggregates data traffic from various OverQoS users on its AS, and forwards the aggregated data on the CLVLs to the OverQoS nodes that reside on the destination ASes. This architecture has multiple resource management schemes that allows the provision of statistical loss and bandwidth guarantees to the OverQoS users. However, it does not address the issue of discovering and deploying the CLVLs, and assumes the existence of an external mechanism (e.g. RON) to perform these tasks.

Rahul, et al. [52] proposed a distributed scheme to discover and select alternate paths between an user at the edge of the network, and a server within a tier-one network[3]. Their scheme focuses on the discovery of alternate paths on an overlay network that is composed of servers within tier-one networks. An example of such an overlay network is the Akamai Content Delivery Network [50]. Their scheme requires that a single third-party (e.g. Akamai) controls and manages the overlay nodes, thus it is not suitable for a community of independent service providers, as described in chapter 2.

Fei, et al. [105] proposed a fully distributed scheme to select and utilize virtual paths on overlay networks. However, their approach is aimed at backup alternate paths and not QoS enhanced ones. They provided a heuristic to select a best backup alternate path among a set of potential ones, but they did not address the issue of discovering this initial set of candidate virtual paths. This heuristic was based on detecting virtual paths that diverge (in terms of sequence of ASes) as early as possible from the default paths. In contrast, this thesis proposes a system that allows the discovery of this initial set of candidate alternate paths, while trying to ensure that it contains QoS enhanced paths. Indeed, in some cases (such as an overlay with few participating nodes) there might not exist any QoS enhanced path. Finally, with minor modifications, this proposed system could be used in cooperation with the scheme from [105].

4.4.3 Discovering Virtual Paths

The previous related work subsection outlines that one of the important challenge of the QoS-enhanced virtual/alternate path approach is the discovery and selection of such paths. The previously described contributions propose either a distributed selection scheme that does not perform the discovery task, or a distributed discovery scheme that requires i) the cooperation of some ASes, and ii) the complete dissemination of all the overlay link states to all the participating nodes. This thesis aims at providing QoS to overlay applications with the use of QoS enhanced virtual paths. Overlay applications are composed of distributed application elements, which are hosted on various end-systems over the Internet. Thus, the schemes from the above contributions are not suitable for such an environment. Such

[3] A tier-one network is a network that connects to the entire Internet through solely free peering interconnection. Tier-one networks constitute the backbone of the Internet.

a context requires a fully distributed alternate path discovery and selection solution that i) is hosted and executed on end-systems, ii) does not require any deployment of systems or brokers within the Internet ASes.

This thesis proposes an architecture SPAD (Super-Peer Alternate path Discovery) with such a fully distributed alternate path discovery and selection scheme. This functionality provides a general service to the end-systems, and can potentially benefit any type of overlay applications. Therefore it should resides within a middleware framework, which would be located between the network and the application levels.

Within the overlay application context, nodes play both roles of clients and servers [4], hence the choice of a Peer-to-Peer (P2P) approach in the design of the proposed SPAD architecture. The P2P paradigm has become the focus of several research studies and the basis for many popular applications such as content search and distribution, as in Gnutella [54], or real-time communication, as in Skype [106]. Lua et al. [107] propose a classification of P2P systems into two models, namely structured and unstructured. Examples of unstructured P2P systems include Gnutella and Skype, while examples of structured P2P system include the DHT-based contributions such as Chord [37] or CAN [38], as introduced in chapter 3.2.2. The proposed SPAD architecture is based on the unstructured model. [108] presents a basic alternate path discovery scheme based on the structured models. The initial literature review for this research work suggests that the proposed SPAD is the only available scalable architecture that allows the distributed discovery and selection of QoS enhanced virtual paths.

4.5 CHAPTER SUMMARY

This chapter presented the notion of Quality of Service from different perspectives. It also introduced some methods to describe the QoS requirements and resources from these different perspectives, and outlines the mapping issue between these descriptions. At the network level, the IETF proposed two models to provide end-to-end QoS to applications. At the edge of the network (end-system level), various contributions proposed adaptive application mechanisms and transport protocols to achieve a similar goal. This chapter discussed the benefits and the drawbacks of these various classic proposals.

To provide enhanced end-to-end QoS between the nodes of any overlay applications, this thesis follows another approach that virtualizes the network from the application point of view. This approach proposes to use virtual/alternate paths, which are composed of consecutive end-to-end paths over the IP network, to provide enhanced QoS between two end-systems. This chapter reviewed some contributions that propose schemes to discover and select such alternate paths.

This chapter concludes that these schemes are not suitable to the overlay application context that involves multiple distributed application elements on various end-systems over the Internet. To address this challenge, this thesis proposes a middleware architecture SPAD

[4] For example, a node running a raw-to-AAC audio adaptation service acts i) as a client to the preceding node that provides the raw audio, and ii) as a server for the succeeding node that uses the AAC audio

(Super-Peer Alternate path Discovery) which features a fully distributed QoS enhanced alternate path discovery and selection scheme.

Chapter 5

QoS Enhanced Alternate Paths

5.1 INTRODUCTION

This chapter presents the first contribution of this thesis, namely a detailed description of the previously introduced QoS Enhanced Alternate Paths (QEAPs). The QEAP discovery architecture (SPAD), as described in chapter 6, uses these alternate paths to provide enhanced QoS between the end-systems that compose an overlay application.

First, section 5.2 discusses the origin of these alternate Internet paths. Their existence results from the operations of the Border Gateway Protocol (BGP), which is the de-facto routing protocol between Internet domains. Therefore, QEAPs are inherent to the current Internet. Section 5.3 presents the sets of measurements that are used in all the experiments in the remainder of this dissertation. These data are measured one-way or round trip delays between various end-systems on the Internet. The study of consecutive delay measurements between a given pair of end-systems allows the estimation of another QoS parameter between them, namely the packet-loss rate. Finally, section 5.4 presents the experiments that were performed to study a set of QEAP characteristics, and discusses the corresponding results.

The experiments and analysis from this chapter demonstrate that QEAPs do exist, and provide significant and durable QoS enhancements compared to default the Internet paths.

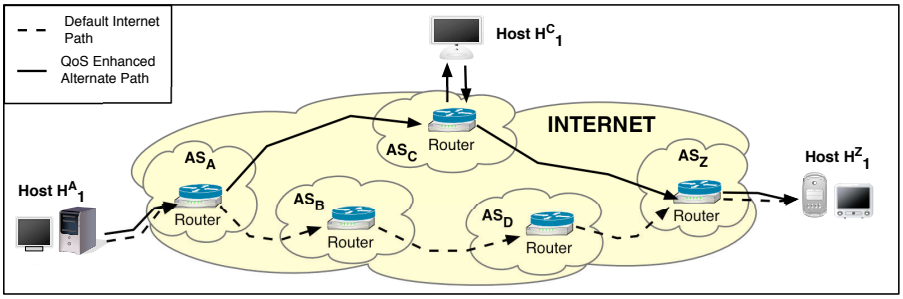


Figure 5.1 deployment of a QEAP between two hosts H_1^A and H_1^Z via a relay host H_1^C

5.2 THE ORIGIN OF QEAPS

The Internet is composed of multiple interconnected Autonomous Systems (ASes). An AS is a network domain that is under the management of a single administrative entity. Within an AS AS_A, data packets from a host H_1^A to another host H_2^A do not leave the AS boundaries, and are routed using an Interior Gateway Protocol (IGP), which is solely managed by the AS_A administrators. However, data packets from H_1^A to another host H_1^Z in AS_Z traverse the boundaries of AS_A, and possibly transit through various other ASes before arriving to AS_Z. To convey these packets, AS_A, AS_Z, and all the possible intermediate ASes have to cooperate, and execute a common routing scheme, i.e. an Exterior Gateway Protocol (EGP). Within the current Internet, the Border Gateway Protocol (BGP) [109] is the de-facto EGP routing scheme.

Within a given AS, a BGP gateway router maintains a routing table that includes the complete route, i.e. succession of ASes, to all the reachable destinations from this router. Gateway routers from neighbor ASes regularly exchange their routing tables, make some possible modifications to them, and propagate a selection of these updates, thus constructing an AS connectivity graph of the Internet. The process of converging towards an up-to-date and accurate AS connectivity graph may take a significant amount of time. As discussed in [104], this convergence time may be in the order of 10min, and in some 5% of all the detected BGP route failures the convergence towards a new available BGP route is in the order of 1 to 2 hours. During that period, the route from a BGP router towards some given Internet destinations might oscillate or experience a complete outage. This high convergence latency constitutes the primary weakness of BGP, and is inherent to its path vector routing protocol [104].

BGP allows the administrator of AS_A to specify a set of policies that determines the routes that data packets from AS_A will take to arrive to their destination ASes. Thus, routing decisions in the majority of the currently deployed BGP routers are primarily made based on administrative policies, and to a much lesser extent based on shortest AS hop count. Indeed, for various reasons, network administrators may not consider QoS optimization as a primary factor when specifying the routing policies of their ASes. For example, if AS_A and AS_C compete on some economic issues, thenAS_A might select a route that goes via AS_B to reach AS_Z, even if that route is longer in terms of AS hop count or latency than a

Table 5.1 Characteristics of the Experimental Data Sets

	Data Sets		
	RIPE-TTM	NLANR-AMP	PlanetLab All-Pair-Ping
Dates of Data Sets	25/05/04, 12/07/04	31/01/03, 17/06/04	11/04/05, 27/04/05
Number of Valid Hosts	52	107	177
Type of Available Measurement	One-way delay	Round Trip delay	Round Trip delay
Measurement Frequency	2 / min.	1 / min.	1 / 15min.

route via AS_C to AS_Z. Although BGP does not guarantee that consecutive packets from H_1^A will take the same route to reach H_1^Z, in practice Paxson [110] has demonstrated that two thirds of BGP routes persist for days or weeks.

For a particular pair of hosts on the Internet, one can create a virtual path by selecting and composing consecutive end-to-end paths. This virtual path might include a succession of ASes, which is different than the succession of ASes on the default path that is returned by the BGP routing mechanisms. Thus, this virtual path constitutes an alternate Internet path. In such a case, it is possible that the alternate path has better QoS characteristics than the default path, e.g. lower end-to-end delay or packet-loss rate [4]. This dissertation refers to these *better* alternate paths as QoS Enhanced Alternate Paths (QEAPs). Figure 5.1 illustrates the deployment of a 2-hop QEAP between two hosts H_1^A and H_1^Z via a relay host H_1^C.

Given these considerations, this thesis makes the fundamental assumption that the triangle inequality does not hold for QoS parameters (such as end-to-end delay and packet-loss rate) on Internet paths. For example, let $\mathcal{N}$ be the set of Internet hosts, $d(x, y)$ the one-way end-to-end delay between two hosts x and y, and $P(A)$ the probability that the event A is true, then this thesis assumes that:

$$\forall (x, y) \in \mathcal{N}, \quad P(\ \exists z \in \mathcal{N} \mid d(x, z) + d(z, y) < d(x, y)\ ) > 0 \tag{5.1}$$

The remainder of this chapter presents some experimental results that support this assumption, and provides further details on the characteristics of QEAPs.

5.3 PRESENTATION OF THE EXPERIMENTAL DATA SETS

The majority of the experiments in this dissertation uses three different sets of publicly available measurement data. These data sets are provided by the following projects:

- the Test Traffic Measurements project from the "Réseaux IP Européens" Network Coordination Centre (RIPE-TTM) [10],
- the Active Measurement Project from the National Laboratory for Applied Network Research (NLANR-AMP) [9],
- the PlanetLab All-Pair-Ping project [8].

Table 5.1 presents the different characteristics of these data sets. They provide a complete matrix of delay measurements at regular time stamps between pairs of hosts on the Internet. Most of these data sets contain more hosts than the displayed *Number of Valid Hosts* (e.g. 117 instead of 107 in the 17/06/04 NLANR-AMP set). However to refine the forthcoming analysis, the considered RIPE-TTM and NLANR-AMP data sets include only European and North-American hosts, respectively. While the considered PlanetLab data sets include hosts with no geographic restriction. Furthermore for a given day, some of the hosts are disconnected from the rest of the network after a particular time stamp, and remain in that state for the rest of that day. Thus, they presents incomplete measurement sets. These transient hosts were removed from the experimental data sets, as they cannot be used in experiments that involve constancy analysis and packet-loss estimation; hence the lower displayed *Number of Valid Hosts*.

In addition to providing end-to-end delay information between hosts, these data sets also mention the cases when that information is not available at a particular time stamp. This information can be used to compute an estimate of the packet-loss rate $L_{x,y}$ between two hosts x, y, at a given instant. For a given time stamp t_i, this thesis use the following equation to compute such an estimate $L_{x,y}(t_i)$:

$$L_{x,y}(t_i) = \frac{\sum_{j=i-k}^{i+k} \lambda_{x,y}(t_j)}{2k+1} \quad with \begin{cases} \lambda_{x,y}(t_j) = 1 & \text{if a loss event occurs at } t_j \\ \lambda_{x,y}(t_j) = 0 & \text{otherwise} \end{cases} \tag{5.2}$$

This equation (5.2) considers a window of size $(2k+1)$ around t_i to compute the estimate packet-loss rate $L_{x,y}(t_i)$. The experiments in this thesis use a value of $k = 2$. The relative accuracy of this estimation depends on the granularity of the measurements from the data sets. For example according to the measurement frequencies from table 5.1, the RIPE-TTM and NLANR-AMP data sets yield a more accurate estimation than the PlanetLab data sets.

Although the selected data sets include a relatively small number of hosts compared to the Internet, these hosts are real-world end-systems, which are located within various Internet ASes. Therefore, the provided measurements reflect a realistic view of the Internet connectivity characteristic in terms of delay. Moreover, these data sets provide successive "snapshots" of these connectivity characteristics for a given date. These attributes ensures that the experimental environments in this thesis accurately mimics the characteristics of real-world Internet environments.

5.4 CHARACTERISTIC ANALYSIS OF QEAPS

Given the available information from the previously introduced data sets, this thesis focuses on the study, discovery, and selection of QEAPs that enhance one or both of the following QoS parameters: latency and packet-loss rate. However, the schemes and mechanisms from chapter 6 can be adapted to QEAPs that enhance other type of QoS parameter, such as bandwidth. In the remainder of this dissertation, the term *Delay-QEAP* refers to a particular QEAP that enhances the latency QoS parameter. Similarly, the term *Loss-QEAP* refers to a particular QEAP that enhances the packet-loss rate QoS parameter. This

Table 5.2 Quantity and Type of Existing QEAPs

	Data Sets		
	RIPE-TTM (17/06/04)	NLANR-AMP (17/06/04)	PlanetLab All-Pair-Ping (11/04/05)
Percentage of pair of nodes with existing QEAPs (i.e. enhanced Delay and/or Loss)	40.4% (±1.6)	31.4% (±1.4)	61.6% (±1.5)
Average Number of QEAPs per pair of nodes	7.5 (±0.5)	10.7 (±1.0)	38.4 (±2.6)
Percentage of Delay-QEAPs among all existing QEAPs	94.2% (±2.6)	72.8% (±6.3)	96.7% (±1.4)
Percentage of Loss-QEAPs among all existing QEAPs	10.1% (±3.6)	46.8% (±6.1)	35.3% (±4.0)

section presents the experiments and the results related to the study of some characteristics of these types of QEAP.

5.4.1 Existence

The following experiment was performed to assess and quantify the existence of 2-hop QEAPs[1], as illustrated in figure 5.1. First, 1000 pairs of node $(A\ ;\ B)$ were randomly selected within each data set. Then for $(A_{ti}\ ;\ B_{ti})$, a given pair at a random point of time t_i, a *brute-force* search algorithm was performed on all the remaining nodes within the considered data set. This algorithm successively considers each remaining node N to assess if the path $[A_{ti} \rightarrow N_{ti} \rightarrow B_{ti}]$ is a QEAP compared to the default path $[A_{ti} \rightarrow B_{ti}]$. Table 5.2 presents the averaged results of this experiment and the related standard deviations over 100 trials. This table only presents the results for a given day for each data sets, as experiments on other days provide similar results. This experiment accounted for the processing delay on the potential relay nodes by adding an offset of 1ms to the delay of all discovered alternate paths before deciding if they were *Delay-QEAPs*. Earlier studies on transport layer processing overhead [111, 112] support this offset value[2].

Table 5.2 shows that there is a significant number of QEAPs between pairs of hosts in each of the data sets. For example on the PlanetLab data set, a QEAP exists in 61.6% (±1.5) of the cases between a random pair of nodes. Furthermore in these cases, there are on average multiple QEAPs available for a given pair of nodes, e.g. 10.7 (±1.0) available QEAPs for a pair of nodes from the NLANR-AMP data set. Finally, table 5.2 shows that among the existing QEAPs the number of *Delay-QEAPs* is more important than the number of *Loss-QEAPs*. For example in the RIPE-TTM data set, 94.2% (±2.6) of the existing QEAPs enhance the delay QoS parameter, while only 10.1% (±3.6) of them enhance the packet-loss parameter. Despite such a lower availability of *Loss-QEAPs*, subsection 5.4.4

[1] Subsection 5.4.5 will provide more details on QEAPs with more than 2 hops.

[2] The SPAD system from chapter 6 is part of a middleware that mediates application requirements and network resources, similar to the role of the transport layer. Thus, it can functionally be located at a similar level. TCP processing delay is about 1ms/packet when no kernel-to-user-space memory copy is performed, which would be the case within a relay node of a QEAP.

Table 5.3 Average Delay and Packet-Loss Gains on QEAPs

	Data Sets		
	RIPE-TTM (17/06/04)	NLANR-AMP (17/06/04)	PlanetLab All-Pair-Ping (11/04/05)
Average Delay Gain	40.6% (±3.3)	36.3% (±5.8)	40.6% (±3.3)
Average Packet-Loss Gain	96.5% (±1.4)	86.6% (±4.2)	96.5% (±1.4)

presents additional experimental results that support the case for the discovery of these *Loss-QEAPs* [3].

5.4.2 Gain

The same experiment as in the previous subsection 5.4.1 was performed to quantify and characterize the average gains in term of delay and packet-loss rate on a QEAP.

Delay Gain

For a given QEAP Q from hosts A to B via host R, the gain in delay G_Q^D at a given time t_i is defined as:

$$G_Q^D = \frac{D_{REF} - D_{QEAP}}{D_{REF}} \tag{5.3}$$

with D_{REF} and D_{QEAP} being the delay on the default path $[A_{ti} \rightarrow B_{ti}]$ and on the alternate path $[A_{ti} \rightarrow R_{ti} \rightarrow B_{ti}]$ respectively. Table 5.3 shows that the QEAPs from the three considered data sets provide on average a delay that is around 40% lower than the delay on the default Internet path. Furthermore, figures 5.2(a), 5.2(c), and 5.2(e) show that slightly more than 40% of the *Delay-QEAPs* provide a delay gain that is superior to 20%.

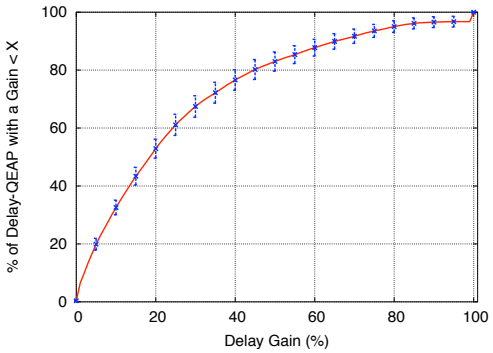
Packet-Loss Gain

For a given QEAP Q from hosts A to B via host R, the gain in packet-loss rate G_Q^L at a given time t_i is defined as:

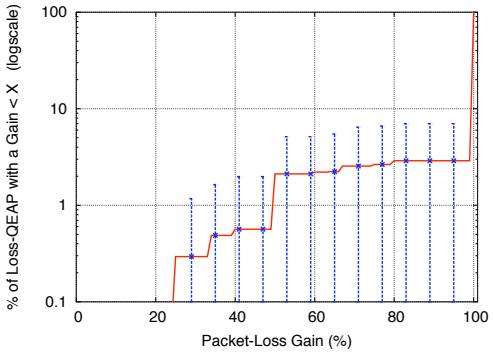
$$G_Q^L = \frac{L_{REF} - L_{QEAP}}{L_{REF}} \tag{5.4}$$

with L_{REF} and L_{QEAP} being the packet-loss rate on $[A_{ti} \rightarrow B_{ti}]$ and on $[A_{ti} \rightarrow R_{ti} \rightarrow B_{ti}]$ respectively. Table 5.3 shows that the QEAPs in the three considered data sets provide on average a packet-loss rate that is significantly lower than the default Internet path, e.g. more than 85% lower in all the data sets. This is due to pairs of node that have a high packet-loss rate on their default internet path at t_i. For these pairs, any QEAP with a packet-loss rate close to 0 will provide a gain close to 100%, thus increasing the average gain in the experiment. Figure 5.2(d) and 5.2(f) illustrate this, and show that for

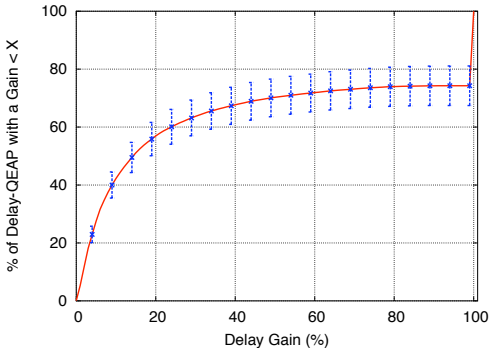
[3] Adding the percentage of *Delay-QEAPs* and *Loss-QEAPs* gives a number greater than 100% because there exist QEAPs that enhance both delay and packet-loss rate. These QEAPs are counted in both categories, hence the overlap.



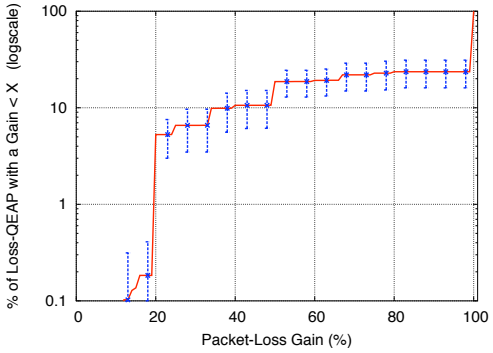
(a) Delay Gain (RIPE-TTM 17/06/04)



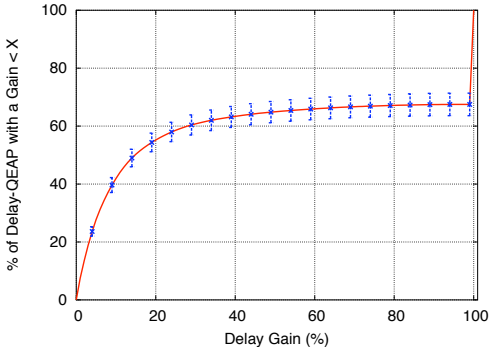
(b) Packet-Loss Gain (RIPE-TTM 17/06/04)



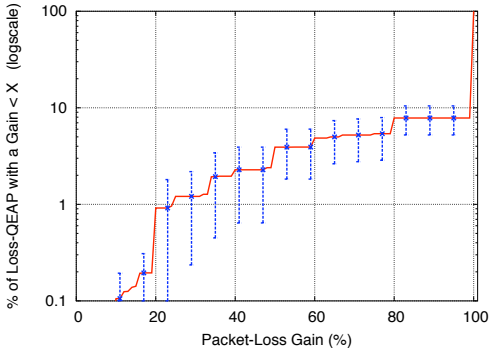
(c) Delay Gain (NLANR-AMP 17/06/04)



(d) Packet-Loss Gain (NLANR-AMP 17/06/04)



(e) Delay Gain (PlanetLab 11/04/05)



(f) Packet-Loss Gain (PlanetLab 11/04/05)

Figure 5.2 Distribution of delay and packet-loss gains for the different data sets.

the NLANR-AMP and the PlanetLab data sets, around 80% and 92% of the *Loss-QEAPs* have a gain close to 100%. Moreover, these figures also show that most of the *Loss-QEAPs* (e.g. 90% and 98%) provide a gain greater than 40%. The *y-axis* on figures 5.2(b), 5.2(d), and 5.2(f) is in logarithmic scale, hence the non-symmetrical appearance of the error bars.

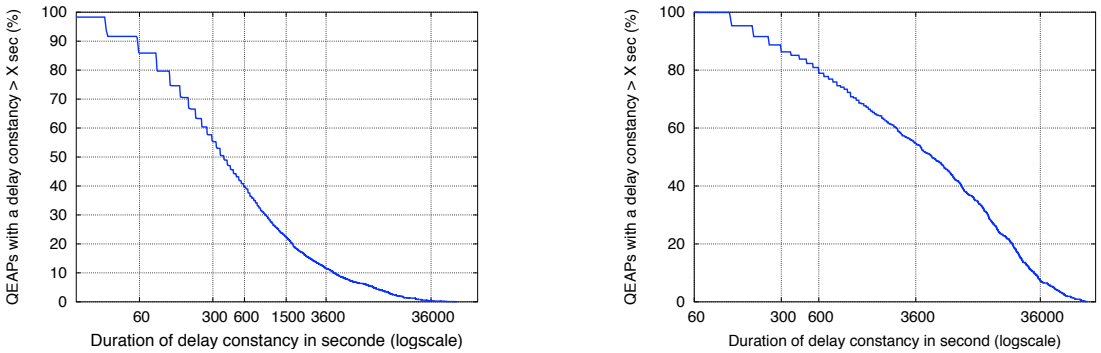
5.4.3 Constancy

Delay Constancy

The following experiment was performed to assess and quantify the constancy of the gain in delay provided by the QEAPs. First similarly to section 5.4.1, 1000 pairs of node (A_{ti} ; B_{ti}) were randomly selected at a random point of time t_i, and the same *brute-force* search algorithm is used to discover all the existing QEAPs between these pairs. Then for a given QEAP $[A_{ti} \rightarrow R_{ti} \rightarrow B_{ti}]$, the durations of the delay constancy $d_{A\rightarrow R}$ and $d_{R\rightarrow B}$ on each hop are estimated. Finally, the durations of the delay constancy $d_{A\rightarrow R\rightarrow B}$ on the entire QEAP is defined as: $d_{A\rightarrow R\rightarrow B} = min(d_{A\rightarrow R}; d_{R\rightarrow B})$. This is a conservative definition. Indeed, the overall delay on a QEAP might still be lower than the delay on the default path, even if the delay on one hop of this QEAP has slightly increase. In such a case, the path $[A_{ti} \rightarrow R_{ti} \rightarrow B_{ti}]$ remains a QEAP despite of the short delay constancy duration.

The following procedure is used to estimate the duration $d_{X\rightarrow Y}$ of the delay constancy between two hosts. First, the delay D_{REF} on the default Internet path $[X_{ti} \rightarrow Y_{ti}]$ at t_i is taken as a reference. Then a low-pass filter is applied to the consecutive delay values D_{tj} at t_j (with $j > i$) between X and Y. The procedure stops when one of these values D_{tk} is out of the interval $D_{REF} \pm 5\%$ at t_k. The delay constancy duration is then equal to $d_{X\rightarrow Y} = D_{tk} - D_{REF}$. The low-pass filter increases the significance of past delay values in the constancy estimation procedure, thus minimizing the effect of eventual outliers. This experiment uses a low-pass filter similar to the filter from the TCP's Round Trip Time (RTT) evaluation method [78]. The accuracy of this constancy estimation procedure depends on the granularity of the measurements in the considered data sets. The RIPE-TTM and the NLANR-AMP data sets have a measurement frequency less or equal than 1 per minute, thus allowing an acceptable coarse estimation of delay constancy durations. However the PlanetLab data sets have a measurement frequency of 1 per 15 minutes, and are not suitable for the proposed estimation procedure.

Figure 5.3(a) and 5.3(b) present the results of this experiment, i.e. the cumulative distribution of the QEAPs that have their delay constancy greater than a given value in seconds. The distribution of the delay constancy on the default Internet paths follows a similar trend than the graphs on these figures. Since i) the delay gain is the difference between the delays on the default and alternate path, and ii) both paths have similar constancy trends; then these figures also reflect the duration of the delay gain constancy. According to these figures, around 57% and 80% of the QEAPs have a delay constancy that lasts more than 5min, thus providing a positive delay gain during the same period. These results are consistent with the analysis from [113, 52], where Internet delays appear constant on a 10 to 30min timescale. Moreover in [114], Brownlee et al. show that 53% of current Internet data streams last between 2s and 15min. Given these results, one can infer that most QEAPs offer delay gains with sufficient durations to benefit a majority of the current Internet data streams.



(a) RIPE-TTM data set (25/05/05) (b) NLANR-AMP data set (17/06/04)

Figure 5.3 Cumulative distribution of the delay constancy duration on QEAPs.

Packet-Loss Rate Constancy

The granularity of the measurements from the data sets, coupled with the packet-loss rate estimation procedure from section 5.3 do not allow an accurate evaluation of the constancy duration of the packet-loss rate QoS parameter. For example, the RIPE-TTM data sets provide a measurement frequency of 1 per 30 sec (which is the highest among the three considered data sets), and the packet-loss estimation procedure requires a window of 5 measurements. Therefore, applying a similar constancy evaluation procedure as in the above delay case would result in a packet-loss rate constancy evaluation that would have a granularity of $5 * 30sec = 2.5min$. Such value is too high to allow an accurate constancy evaluation. To verify this assertion, a similar experiment than the one used in the delay constancy analysis was performed. The resulting constancy distribution graph (not presented in this dissertation) claims that around 97% of the QEAPs have their end-to-end packet-loss rates that are constant for more than 10 minutes. These values do not conform to the conclusions from previous contributions that analyze Internet end-to-end path characteristics. Indeed, Zhang et al. [113] show that Internet end-to-end packet-loss rates are quasi-constant on the scale of less than 10 minutes.

Although the considered data sets do not allow an accurate study of the constancy of the gain in packet-loss rate on QEAPs, based on the conclusions from previous contributions [113], this thesis make the assumptions that *Loss-QEAPs* offer gains in packet-loss rate that are constant for a duration of up to 10 minutes. As discussed previously, such a duration is within the life-span of the majority of Internet data streams.

5.4.4 The Case for Considering both Delay and Packet-Loss Parameters

Previous contributions concluded that there exists some correlation between end-to-end delay and packet-loss on a given Internet path. For example, Moon [115] demonstrates that a packet-loss event is often located within the same timeframe as a delay increase. Therefore, one could make the assumption that searching for a QEAP that would enhance any of the two QoS parameters (e.g. delay or packet-loss), would be sufficient to ensure

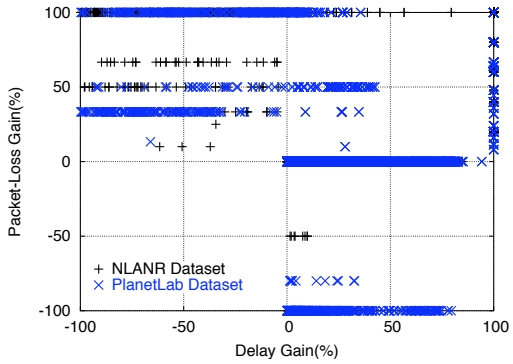


Figure 5.4 Scatter-plot of QEAPs according to their Delay and Packet-Loss gains (by definition a QEAP cannot have both gains < 0).

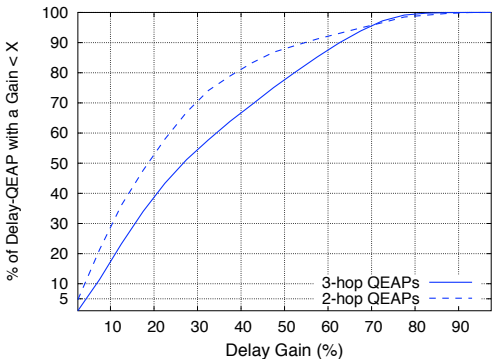


Figure 5.5 Delay gain comparison: 3-hop versus 2-hop QEAPs (RIPE-TTM 25/05/05).

the discovery of an existing QEAP between two hosts. Furthermore, given the result from table 5.2, one could infer that delay would be the most relevant QoS parameter to select for such a discovery scheme.

To investigate this assumption, figure 5.4 presents a scatter-plot of the delay gain versus the packet-loss gain on all the existing QEAPs of the experiment from section 5.4.1. This figure presents the results from the NLANR-AMP, and the PlanetLab data sets. The results from the RIPE-TTM data sets are similar and not presented here. Figure 5.4 shows that there exists a significant number of *Loss-QEAPs* that provide only a better packet-loss rate than the default Internet paths, and no improvement or worse, an increase, on the corresponding delay. A similar comment also stands for the *Delay-QEAPs*. These results complement the ones from the last 2 rows of table 5.2. Therefore, despite the existence of some correlation between delay and packet-loss rate on Internet paths, a QEAP discovery scheme that would be designed to search for alternate paths based only on a single QoS parameter (such as the delay), would not be efficient in discovering QEAPs that would enhance another correlated QoS parameter (such as packet-loss rate).

5.4.5 Comparison between 2-hop and 3-hop QEAPs

The following two experiments were performed to study 3-hop QEAPs (i.e. alternate paths that involve 2 relay nodes), both of them are modified version of the experiment from subsection 5.4.1. The first experiment performs a *brute-force* search algorithm to discover only 3-hop QEAPs. The second experiment initially performs a 2-hop QEAP discovery on the entire set of 1000 randomly selected pair of hosts, and then performs a 3-hop QEAP discovery on the remaining pair of hosts that do not have any existing 2-hop QEAP. Table 5.4 shows the results of these experiments.

This table shows that the numbers of existing 3-hop QEAPs are almost equal to the numbers of existing 2-hop QEAPs for the three considered data sets, e.g. 33.3% ($\pm$1.5) for 3-hop QEAPs compared to 31.4% ($\pm$1.4) for 2-hop ones in the NLANR data set. Therefore, searching for 3-hop QEAPs instead of 2-hop ones does not significantly increase the prob-

Table 5.4 Comparison between 2-hop and 3-hop QEAPs

Percentage of pair of nodes with existing QEAPs (type of QEAP from the below categories)	Data Sets		
	RIPE-TTM (17/06/04)	NLANR-AMP (17/06/04)	PlanetLab All-Pair-Ping (11/04/05)
only 2-hop QEAPs	40.4% (±1.6)	31.4% (±1.4)	61.6% (±1.5)
only 3-hop QEAPs	40.2% (±1.5)	33.3% (±1.5)	66.2% (±1.6)
initial search for 2-hop QEAPs, then search for 3-hop QEAPs on pair of nodes without 2-hop ones	42.7% (±1.5)	34.7% (±1.5)	67.4% (±1.5)

ability of discovering an alternate path with enhanced QoS characteristics. Furthermore, table 5.4 also shows that the total number of discovered QEAPs when initially searching for 2-hop ones then for 3-hop ones, is only slightly higher than the number of discovered 2-hop QEAPs from subsection 5.4.1, e.g. 34.7% (±1.5) compared to 31.4% (±1.4) in the NLANR data set. This result implies that the majority of pairs of hosts with 3-hop QEAPs have also existing 2-hop QEAPs. Finally, figure 5.5 presents a comparison between the distributions of gain in delay for 2-hop and 3-hop QEAPs. This figure shows that in 90% of the cases, the delay gain via a 3-hop QEAP is less than the delay gain via a 2-hop QEAP.

According to these various results, the benefit from searching and deploying alternate paths with 3 hops instead of 2 is relatively negligible. Although an implementation of the SPAD QEAP discovery system from chapter 6 would still gain from implementing such a functionality, the remainder of this dissertation will only consider 2-hop QEAPs. With minor modifications, the schemes and mechanisms from chapter 6 can be adapted to discover, select and use QEAPs with more than 2 hops. Moreover, such modified schemes and mechanisms will yield similar performances as the results from the evaluations in chapter 7.

5.5 CHAPTER SUMMARY

This chapter presented the first contribution of this thesis, namely a detailed analysis of the QoS Enhanced Alternate Paths (QEAPs), which are used to provide enhanced QoS between the end-systems that compose an overlay application.

This chapter first discussed the origin of QEAPs. Section 5.2 presented an overview of the Border Gateway Protocol (BGP), which is the de-facto routing protocol between Internet domains. This overview outlined the prevalent use of non-QoS optimal decisions by network administrators, who specify and implement routing policies into BGP routers. Thus, QEAPs result from these non-QoS optimal BGP routing policies, and are inherent to the current Internet.

Then this chapter presented the measurement data sets that are used in all the experiments in this dissertation. These data sets provide latency and packet-loss information between various end-systems on the Internet. While, they include a relatively small set of hosts compared to the Internet, these hosts are real-world end-systems from various ASes. Therefore, these data sets reflect a realistic view of the Internet connectivity char-

acteristic, and provide the basis for experimental environments that accurately mimic the characteristics of a real-world Internet environments.

Finally, this chapter presented a detailed analysis of some particular QEAP characteristics. This analysis concluded that:

- a significant number of QEAPs exists within the current Internet,
- most of these QEAPs provide substantial gains in term of end-to-end delay and packet-loss rate,
- these gains are on average constant for a sufficient duration to benefit the majority of the current Internet data streams,
- the existing correlation between Internet delay and packet-loss rate does not imply any evident correlation between gains in delay and gains in packet-loss rate on QEAPs. Thus a QEAP discovery scheme should consider both QoS parameter to provide optimal services to the end-systems,
- searching for 3-hop QEAPs provide negligible benefit increase compared to searching for 2-hop QEAPs. Therefore, the remainder of this thesis focuses only on 2-hop QEAPs. With minor modifications, the QEAP discovery schemes from chapter 6 could be adapted to QEAPs with more than 2 hops.

The next chapter presents the second and main contribution of this thesis, namely the SPAD architecture. SPAD proposes a complete set of schemes to discover, select and use QEAPs to provide enhanced QoS between the end-systems.

Chapter 6

SPAD: a Super-Peer based Alternate path Discovery architecture

6.1 INTRODUCTION

The Super-Peer based Alternate path Discovery (SPAD) architecture is the second and main contribution of this thesis. It is a distributed system that aims at enhancing the QoS between two peers of an *overlay association*. As introduced earlier, the fundamental idea behind SPAD is to discover, select, and utilize alternate composite Internet paths with enhanced QoS characteristics between two communication entities. SPAD is based on a community of cooperative end-points, which are grouped as an unstructured Super-Peer network [7]. Within this community, some end-points act as normal-peers and originate requests for information on desired alternate paths towards some destinations. Other end-points act as super-peers and process these requests to determine alternate paths. They forward the requests to other super-peer neighbors, exchange connectivity information on the SPAD community, and return the relevant information back to the requesting peers. With this information, the requesting peers discover some potential QEAPs, and select the ones that *best* meet their requirements.

First, section 6.2 presents a preliminary work on a simple QEAP discovery scheme. This work demonstrates the feasibility of discovering QEAPs in a distributed manner within a cooperative environment. SPAD extends this basic scheme to provide a complete system for QEAP discovery and selection. Section 6.3 presents an overview of the SPAD architecture, and discusses the related design assumptions. Then section 6.4 describes in details: i) the two complementary SPAD schemes, namely a reactive and a proactive schemes, which allow the discovery of alternate paths, and ii) the QEAP selection scheme, which allows the selection of the optimal QEAP among a set of potential ones, according to some application requirements.

The performance evaluation of these schemes are presented in chapter 7.

6.2 A SIMPLE QEAP DISCOVERY SCHEME

6.2.1 Scheme Overview

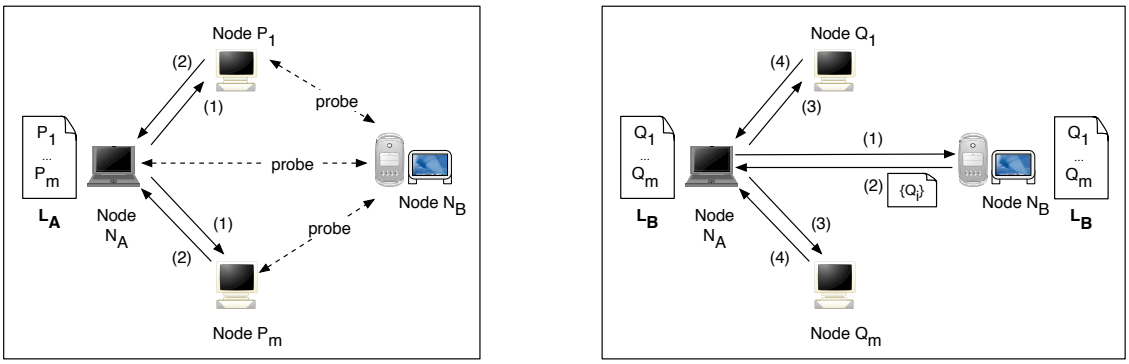
Overlay applications are potentially composed of large sets of application elements, which are scattered across several network domains, and are organized in complex *overlay association* structures. As introduced in section 4.4.3 in such distributed structures, hosts perform both roles of a client and a server. Thus, it is realistic to use a distributed peer-to-peer approach to establish and manage an *overlay association* and its related QoS. The users of the *overlay application* then benefit from enhanced QoS without relying on a central administrative and/or management entity, which is a potential single point a failure. These considerations led to the design of a preliminary simple distributed peer-to-peer scheme, where a host involved in an overlay application collaborates with other peer hosts. From this collaboration, it gains partial knowledge of the network connectivity parameters (e.g. delay), and executes a distributed algorithm in order to discover QEAPs towards other hosts within the same overlay application. This design assumes that all participating peers are willing to share a fixed amount of their resources with the community, to assist in discovering alternate paths and relaying data traffics.

The operations of the proposed simple scheme consist of 2 main phases: the initialization (i.e. bootstrap) of a new host in the community, and the discovery of QEAPs[1]. The following subsections 6.2.2 and 6.2.3 describe these 2 phases in the context of *Delay-QEAPs* discovery. The SPAD system (as described in the next sections 6.3 and 6.4) supports the discovery of both *Delay-QEAPs* and *Loss-QEAPs*.

6.2.2 Peer Initialization

During its initialization phase, a host N_A builds its initial partial knowledge of the network, namely a fixed-size list L_A of other peer hosts. N_A uses the peers in that list as candidate relay nodes in its future alternate paths. In this simple scheme, the list L_A contains a selection of the currently known peers N_Xs that have the *best* QoS values on the default Internet path $[N_A \rightarrow N_X]$, e.g. the lowest delay. To build its initial list, N_A contacts a small set of bootstrap peers, requests their own lists (i.e. bootstrap lists), evaluates the QoS parameter on the default Internet paths towards the nodes from these bootstrap lists, and selects the nodes corresponding to the paths with the *best* QoS values. To discover these bootstrap peers, N_A can use a distributed directory service based on one of the various robust scalable peer-to-peer frameworks, such as the DHT-based Chord [37]. For example in the case of the delay QoS parameter, N_A joins a Chord ring with other nodes deploying the same simple QEAP discovery scheme. It asks a small fixed number of its Chord successors for their own lists. Then it evaluates the delays on its default Internet paths towards the nodes within these bootstrap lists, and includes in L_A the ones having the lowest values, i.e. the ones being *closest* in term of delay. At the end of this phase, N_A has a list of candidate relay nodes based on its current knowledge of the network. As N_A discovers other nodes during its lifetime (via application QEAP requests, and its interaction with other hosts within overlay applications), it updates its list accordingly.

[1] Section 6.4.5 presents more details on optimal QEAP selection among a set of discovered ones.



(a) First Search-Level (b) Second Search-Level

Figure 6.1 A preliminary simple distributed QEAP discovery scheme.

With this updating strategy, the more a host participates in *overlay applications* by hosting an application element, the more it can potentially be used as relay to enhance the QoS of other *overlay applications*. The size of L_A and the number of bootstrap nodes are important scalability parameters. In this simple scheme, the list size is fixed to $log\mathcal{N}$ ($\mathcal{N}$ being the number of peers deploying this scheme) and the bootstrap node number varies between 2 and 4. $\mathcal{N}$ is not known a-priori and increases with new participating nodes. However, when initially deploying this scheme, peer participants could agree on an upper bound value for $\mathcal{N}$, and fix their list size (and those of subsequent participants) accordingly.

6.2.3 Scheme Description

At this stage, N_A is ready to accept an application element request for the search of a QEAP towards another application element on a host N_B. Upon receiving such request, it executes the following cooperative distributed algorithm, which is based on a conditionally controlled flooding technique. This algorithm makes the assumption that there are accurate low cost techniques to evaluate the current value of a certain QoS parameter on a directed Internet path between two end-systems. Sections 6.3.1 and 6.4.4 discuss further this assumption in the case of the delay and packet-loss rate QoS parameters, respectively. The proposed simple QEAP discovery algorithm is composed of two search levels. The second level is executed only if the first one is not successful, i.e. if it has failed to discover any QEAPs.

Figure 6.1(a) describes the first search-level. An application element on node N_A would like to set up a communication path with a peer application element on node N_B. First N_A probes N_B to get the delay value (*delayDef*) on the default Internet path $[N_A \rightarrow N_B]$. Then N_A simultaneously sends a request (1) to all the nodes in its candidate relay list $\{P_i\}_{1 \leq i \leq m}$, asking them to evaluate the delay on their default Internet paths to N_B. A node P_X (with $X \in \{1, ..., m\}$) checks its available resources to assess its capacity to participate in an alternate path from N_A. This task requires an admission control function, as briefly discussed in section 6.5. If P_X accepts to be a relay, it replies to N_A with its delay value towards N_B, and keeps a temporary resource reservation, waiting for N_A's path selection. A non-willing P_X returns an infinite delay value to N_A. When N_A receives back these

delay values (2), it computes the overall delay on each candidate alternate paths. The alternate paths with a delay value smaller than *delayDef* are QEAPs. N_A compares them to select the one with the minimum delay value. If such an alternate path exists and if it meets some optional user QoS preferences (e.g. delay $<$ 2 *DelayDef*), N_A uses it as the QEAP to reach N_B. In this case, the first search-level is successful, and there is no need to execute the second search-level. Since the size of the candidate relay list is $log\mathcal{N}$, the message cost to discover the QEAP is equal to $2log\mathcal{N}$. As this scheme assumes the existence of techniques to evaluate delays on Internet paths at low cost, the message cost does not include the delay probes. Furthermore based on the delay constancy results from section 5.4.3, it is possible to design a simple delay estimate cache in each participating node, hence removing the need to re-evaluate paths leading to recently visited nodes. The admission control function on each candidate relay node, together with N_A's selection of the QEAP with the minimum delay ensure load balancing among the candidate relay nodes.

Figure 6.1(b) describes the second search-level. N_A executes this search-level only when the previous one is not successful. N_A sends a request to N_B (1) asking for its candidate relay list (2). Similarly to the previous search level and in cooperation with the nodes from N_B's list $\{Q_i\}_{1\leq i\leq m}$, N_A evaluates the delay values on the candidate alternate paths through the Q_X (with $X \in \{1, ..., m\}$) nodes, (3) and (4). A QEAP is then discovered and selected in the same manner as in the previous search-level. The message cost for this second search-level is equal to $2(1 + log\mathcal{N})$. The total message cost when performing the complete search algorithm is either equal to $2log\mathcal{N}$ or to $2(1 + 2log\mathcal{N})$. Section 7.2 provides the average distribution between these two cost values.

The first search-level evaluates possible alternate paths by trying known first hops with lowest delay value on the directed path $[N_A \rightarrow N_B]$, hence the use of L_A. The second search-level evaluates possible alternate paths by trying known last hops with lowest delay value on the same directed path $[N_A \rightarrow N_B]$. This requires that N_A knows the existence of these possible last hops, which it does not. Moreover, N_B only knows the hops with the smallest delay value on the directed paths $[N_B \rightarrow N_X]$ (i.e. the nodes in the list L_B). The IP routing mechanism does not guarantee the symmetry of the routing paths and their related delays. However if delays on both ways of a given path were highly correlated, then N_A could use L_B to perform the second search-level. Section 6.3.2 presents experimental results that support this assumption.

Once N_A has discovered and selected a QEAP via a given node N_X, it notifies the involved nodes N_X and N_B. N_X changes its temporary resource reservation to a permanent one that will last for the entire communication duration. It creates the necessary states to relay the traffic from node N_A to N_B. N_A uses the alternate path and monitors the experienced QoS value. Upon eventual QoS degradation, N_A has the option to discover and use another QEAP. It also keeps a list of all the discovered alternate paths from the last QEAP discovery phase and use them as backup paths. At the end of the communication, N_A notifies N_X, and all the states and resources associated to the QEAP are released. N_A might cache the details of the successfully used QEAP for further use.

6.2.4 Discussion

Section 7.2 presents the performance evaluation of this simple scheme in terms of successful QEAP discovery. This evaluation is based on experiments on the RIPE-TTM data sets.

The main limitation of this simple scheme is the fact that N_A knows only delay information on paths of the type $[N_A \rightarrow N_X]$ and $[N_B \rightarrow N_Y]$. Thus, it has only a minor partial knowledge of the connectivity parameters within the community. The following simple analysis illustrates this limitation. Assuming that the size of a candidate relay list is $log\mathcal{N}$, and that each entry of this list contains information about a given potential relay node, then the percentage of nodes which are known to N_A at the end of the second search-level is given by:

$$KnownNode = \frac{2log\mathcal{N}}{\mathcal{N}-2} \qquad and \qquad \mathcal{N} \rightarrow +\infty \;\Rightarrow\; KnownNode \rightarrow 0 \tag{6.1}$$

Expression (6.1) shows that as the community grows, the relative percentage of nodes known to N_A decreases, thus eventually decreasing the QEAP discovery performances of the presented simple scheme.

Since measurement techniques can be asymmetric (such as King introduced in section 6.3.1), there might exist nodes N_Z in the community that have delay information of the type $[N_Z \rightarrow N_A]$ or $[N_Z \rightarrow N_B]$. If N_A had access to this information it could send probe message to these nodes and improve its chance of discovering a QEAP, assuming quasi-symmetry of the considered QoS parameter (section 6.3.2). The motivation behind the SPAD schemes (described in the following section 6.4) is to have this information stored in a distributed database managed by super-peer nodes. Each node N_Z in the community associates with a super-node that keeps a copy of its list L_Z, which contains information on paths of the type $[N_Z \rightarrow N_I]$. When a node requests a QEAP, these super-peer would cooperate to provide the requesting node with relevant information concerning potential relay nodes for that QEAP.

6.3 SPAD: ARCHITECTURE AND DESIGN ASSUMPTIONS

6.3.1 Architecture Overview

The SPAD architecture is based on a community of cooperating end-points that is organized as an unstructured Super-Peer network [7]. When joining this network, an end-point acts by default as a normal-peer. Any peer may evolve to become a super-peer. Each super-peer is associated with a maximum number m of normal-peers, as illustrated in figure 6.2 where this maximum number is m=3. In the SPAD community, super-peers are altruistic peers that in general have more resources at their disposal and therefore carry out more functions than normal-peers. Thus, in addition to performing the same functions than a normal-peer, a super-peer performs some extra functions for the community. The SPAD architecture uses similar super-peer selection/election techniques as in [116] to incrementally construct scalable balanced super-peer topologies.

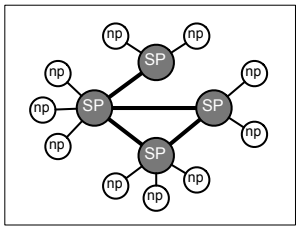


Figure 6.2 A Super-Peer network (SP: super-peers; NP: normal-peers).

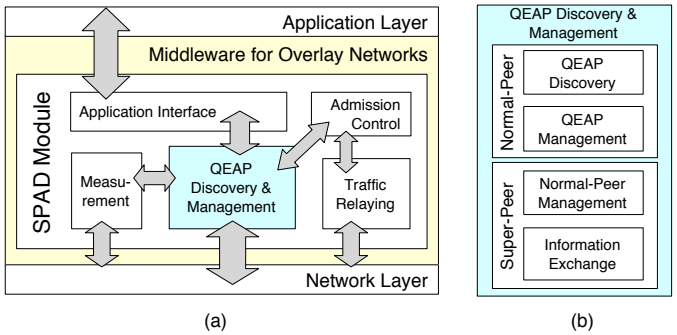


Figure 6.3 (a) Structure of a SPAD module within a middleware framework on each SPAD peer, (b) Details of the "Discovery & Management module"

The SPAD QEAP discovery method is based on the following basic procedure. Each peer collects a small amount of QoS information (e.g. delay) about its connectivity to other peers, and passes this information to its associated super-peer. Thus, the information about QoS parameters between the nodes on the overlay is distributed among the super-peers. When receiving a QEAP request from one of its associated normal-peers, a given super-peer will access this distributed knowledge and will return any information relevant to that request to the normal-peer. With this information, a normal-peer can discover potential QEAPs, and send queries to the corresponding candidate relay peers, requesting their participation in a QEAP. SPAD super-peers use two complementary information exchange schemes (a reactive one and a proactive one) to access the distributed QoS information. These schemes will be described in detail in section 6.4.

Figure 6.3 presents the structure of the SPAD module located within the middleware framework for overlay network running on each SPAD peer, as introduced in section 2.2. Within this framework, the SPAD module could be associated with other modules providing services to the nodes of the overlay network. For example, the SPAD module could be associated with a transport module providing services similar or compatible with TCP. Such associations are not within the scope of this thesis, and will not be discussed further.

The *Measurement* module is responsible for collecting QoS information such as delay or packet-loss rate towards other peers. Measuring or estimating QoS parameters in an accurate, unobtrusive, and efficient manner is an open research issue. However recent applications like King [117] indicate that measurement/estimation tools with these properties are becoming available. King uses small numbers of cautiously crafted recursive DNS queries in a novel way to accurately estimate end-to-end delays. It is assumed that the *Measurement* module will be based on such tools.

The *QEAP Discovery & Management* module is the primary focus of this chapter. This module includes two sub-modules. All peers execute the *Normal-Peer* sub-module, in addition super-peers also execute the *Super-Peer* sub-module. The first sub-module handles the functions related to QEAP discovery, such as initiating QEAP requests, selecting the *best* QEAP among discovered ones, or processing queries to be a relay in other QEAPs. It also handles management functions for QEAPs currently being used by this peer, such

as monitoring QoS on current QEAPs, recovering from QEAP failures. The second sub-module handles the super-peer related functions. It is responsible for the management of the associations between this super-peer and its set of normal-peers, namely setting or terminating associations, storing received QoS information, and processing QEAP requests. It is also the module that executes the two complementary schemes used to access the QoS information distributed among the SPAD community, in order to reply to QEAP requests.

The *Application Interface* module is responsible for making SPAD's functionalities available to the above application element. It provides 2 types of Application Program Interface (API): i) a *transparent* one, with primitives similar to the ones given by the current operating system network stacks (such as the Socket API in Unix based systems); and ii) a *specialized* one, with primitives and QoS mapping functions that allows applications aware of SPAD to request QEAPs with specific QoS requirements[2].

The *Traffic Relaying* module handles the relaying of data traffic from a peer towards another peer. This relaying function is activated when the peer running this SPAD module accepts to be part of QEAP between two other peers (similar to host R in figure 4.4). This module also monitors the available network resources for this peer, and provides this information to the *Admission Control* module.

The *Admission Control* module collects information from the other modules, and builds a view of the available computing and networking resources of this peer. Based on this view, it decides if this peer is able to participate as a relay in a given QEAP, it passes this decision on to the QEAP Discovery module that will forward it to the requesting peer. Section 6.5 briefly introduces the potential admission control schemes that can be used in this module.

6.3.2 Design Requirement and Assumptions

The performance of the proposed SPAD architecture depends on its ability to discover existing QEAPs, which in turn relies essentially on the performance of the reactive and proactive information dissemination schemes described in section 6.4. These schemes should allow the retrieval of all existing information relevant to a given QEAP request, while minimizing the associated bandwidth usage and the latency. In that regard, the design of these schemes makes two assumptions, namely the quasi-symmetry and constancy of QoS parameters, and the similarity of external connectivity from within an Internet Autonomous System (AS). As described in section 5.4, this dissertation presents a SPAD architecture that focuses on the discovery of *Delay-QEAPs* and/or *Loss-QEAPs*, i.e. alternate paths with better delay and/or packet-loss rate than default Internet paths. Without loss of generality, the remainder of this subsection shows the validity of the above assumptions in the context of *Delay-QEAPs*.

Quasi-symmetry and constancy of Internet end-to-end delays

As explained in previous section 6.3.1, each SPAD peer measures the delay from itself to a number of other peers and passes this information to its super-peer. The super-peers ex-

[2]Section 6.4.5 presents more details on these two types of API.

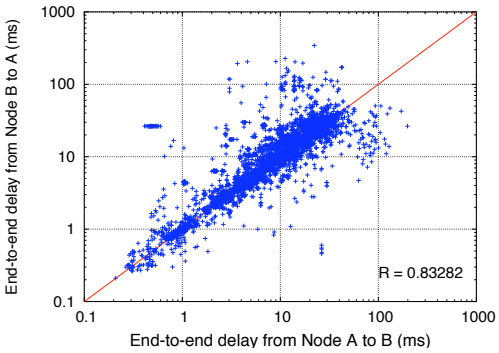


Figure 6.4 Comparison between *delayAtoB* and *delayBtoA* (RIPE-TTM 25/05/04).

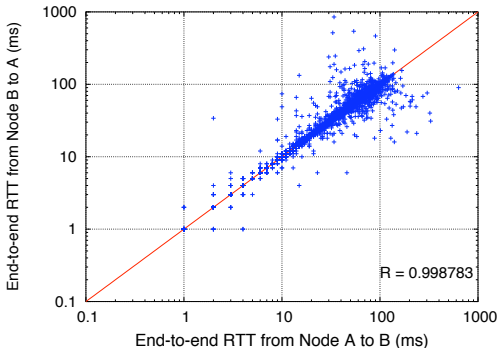


Figure 6.5 Comparison between *delayAtoB* and *delayBtoA* (NLANR-AMP 30/01/03).

change this information, thus building an overall view of the community's delay characteristic. The accuracy of this overall view and the frequency of the measurement/information updates depend on the constancy of Internet delays. Based on the studies from [113, 52] and from section 5.4.3, the SPAD architecture assumes that Internet delays are steady on time scales of 10-30 minutes.

Some measurement softwares, such as King [117], can be asymmetric: when running on a node A, they measure *delayAtoB*, but they do not provide *delayBtoA*. Furthermore, there is no guarantee that node B will measure *delayBtoA*. Thus this information might not exist within the community at a given time. However, it is possible to overcome this limitation and significantly minimize measurement overheads by assuming that *delayAtoB* is similar to *delayBtoA*. This approximation supposes that Internet end-to-end delays are quasi-symmetric despite IP routing mechanisms not guaranteeing symmetry of routing paths and related delays. Figure 6.4 and 6.5 present the comparisons of *delayAtoB* versus *delayBtoA* for 10_3 pairs of nodes from the RIPE-TTM and the NLANR-AMP data sets. These figures show that the average ratios R between both delays are approximately of 0.8328 and 0.9988, thus validating the above assumption.

Similar external connectivity of end-points within a same AS

The exchange of information messages within a large community of distributed peers consumes bandwidth. Minimizing this bandwidth consumption is a primary issue when designing an information exchange scheme. SPAD uses the following aggregation method to minimize the number and the size of these information messages. A simple information of delay between a host X_1 and another host Y_1 could be of the form: $[X_1 ; Y_1 ; delayX_1toY_1]$, similarly $[X_2 ; Y_2 ; delayX_2toY_2]$ for X_2 and Y_2. Assuming that two hosts within a same Autonomous System (e.g. hosts X_1 and X_2 within a same AS_U) experience the same connectivity characteristics towards other hosts within a different AS (e.g. host Y_1 and Y_2 within AS_V), the above simple delay information can be aggregated into a complex entry of the form: $[AS_U ; AS_V ; delayUtoV ; hostInU ; hostInV]$, where $delayUtoV$ is the mean of $delayX_1toY_1$ and $delayX_2toY_2$; and where $hostInU$ contains either X_1, or X_2. To allow some redundancy against host failure, $hostInU$ can contain a small list of hosts

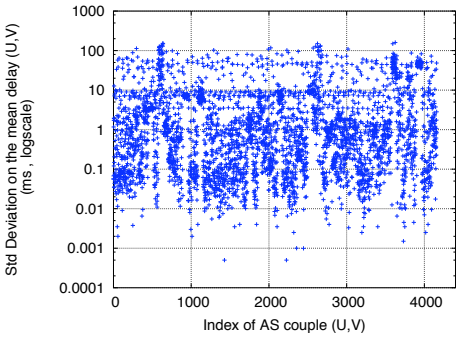


Figure 6.6 Distribution of the standard deviation of the delay (U,V), for all AS pairs (PlaneLab 27/04/05).

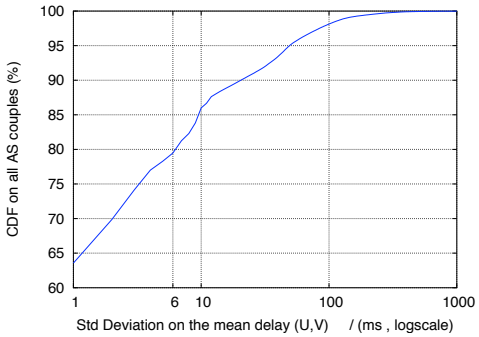


Figure 6.7 Cumulative distribution on all AS pairs of the standard deviation of the delay (U,V), (PlaneLab 27/04/05).

within AS_U. This complex entry provides aggregated delay information between AS_U and AS_V. Each SPAD super-peer performs this aggregation method on the delay information provided by their associated normal-peers. These aggregated entries constitute the information messages that are exchanged between super-peers, thus reducing bandwidth and storage utilization. Super-peers continuously updates these entries based on the information from other peers. Each entry is marked with a time stamp to reflect its accuracy: the older an entry is, the less accurate it might be.

The proposed aggregation scheme provides accurate results only under the assumption that hosts within a given AS experience the same connectivity characteristics towards hosts in other ASes. To support this assumption, the following experiment was performed on the PlanetLab measurement data sets. The IP-to-AS mappings of the 177 hosts from these data sets provide an average of 62 ($\pm$4.0) different ASes. Thus, on average 2.85 ($\pm$0.1) hosts are within the same AS. For each pair of AS (U ; V), the average delay from U to V and the corresponding standard deviation were computed. Figure 6.6 shows that most AS pairs have their standard deviation of the delay between 0.01 and 10 ms. Figure 6.7 indicates that for about 80% of the AS pairs, this standard deviation is equal or less than 6ms. Although the studied data sets have a small number of ASes, these results still support the above assumption, and confirm the use of the described aggregation method.

Finally, the implementation of this aggregation scheme requires an accurate IP-to-AS mapping mechanism. Recent research studies have proposed various mechanisms to perform such a mapping [118, 119]. The RIPE-RIS (Routing Information Service) project [119] retrieves and compiles BGP routing tables 3-times a day from over 300 worldwide routers. Specialized RIS whois servers provide a mapping service of IP prefixes to AS numbers. Any host can query these servers: sending its IP address, and receiving back its AS number. In SPAD, peers joining the community will be able to use a similar mapping service to retrieve their AS number. They will add that information to their network address to compose their SPAD node ID. Since each peer makes a unique query, the additional load should not significantly impact the operations of servers designed to provide services to the Internet community.

6.4 SPAD: SYSTEM DESCRIPTION

This section describes in details the SPAD QEAP discovery and selection schemes. Without loss of generality, the following subsection 6.4.1, 6.4.2, and 6.4.3 describe the discovery schemes in the context of *Delay-QEAPs*. Then subsection 6.4.4 presents the minor modifications that are required to allow these schemes to discover both *Delay-QEAPs* and *Loss-QEAPs*. Finally, subsection 6.4.5 describes the scheme, which is used to select the optimal QEAP among a set of discovered ones, according to the user QoS requirements.

6.4.1 SPAD Peer Initialization

When a new host N_A joins the SPAD community as a normal-peer, it performs the following initialization procedures. First, N_A establishes its partial knowledge of the community, by building a list L_A of other peers. This list is named a relay list (Rlist), and has a fixed size. It contains a subset of the peers known to N_A, that have the *best* QoS values, for example the lowest delay, on the default Internet path from N_A to them. To build L_A, N_A contacts a small random set of super-peers $\{SP_i\}$, and requests their list of associated normal-peers. It measures/estimates the QoS parameter (e.g. delay) on the default Internet paths towards nodes within these lists, and includes in L_A the nodes corresponding to the paths with the *best* values (e.g. the lowest delay). To discover the $SP_i s$, N_A can use a scalable distributed directory service such as Chord [37]. For example, peers in the community could form a Chord-ring, N_A would join that ring, ask a small fixed number of its Chord successors for the identity of their super-peers, contact these super-peers and retrieve their list of associated normal-peers. Then it would evaluate the delay towards the nodes in these lists, and retain in L_A the nodes having the lowest delay. Thus L_A contains entries of the form [N_A ; N_X ; $delay N_A to N_X$]. If N_A and N_X are in the same AS (e.g. $myAS$), they would probably share the same gateway towards another node N_B not in $myAS$. The paths $[N_A \rightarrow N_B]$ and $[N_X \rightarrow N_B]$ would then be similar, and the chances that a QEAP exists from N_A to N_B via N_X would be low. Therefore to ensure node diversity in L_A, potential $N_X s$ send their AS number to N_A during its initial delay evaluations. A node N_X gets its AS number via the IP-toAS mapping mechanism described in section 6.3.2. N_A uses that information to filter out $N_X s$ in the same AS as itself, and $N_X s$ in redundant ASes. For example, if N_X, N_Y, N_Z have the same AS number, they are "redundant", and N_A will consider only one of them for inclusion in L_A.

N_A periodically re-evaluates the delays in L_A and updates it. Existing results on Internet delay constancy [52, 113] can be used to select appropriate re-evaluation intervals. Furthermore, as N_A discovers other peers via user connection requests, it updates its list accordingly. With this updating strategy, the more a host participates in *overlay applications* by hosting a software element, the more it can potentially be used as relay in QEAPs for other *overlay applications*. The Rlist size and the number of bootstrap super-peers are important scalability parameters. In SPAD, the list size is fixed to $log\mathcal{N}$ ($\mathcal{N}$ being the number of peers in the community) and the number of bootstrap super-peers varies between 2 and 4. $\mathcal{N}$ is not known a-priori and increases with new participating nodes. However during initial deployment, the first participating peers could agree on an upper bound value of $\mathcal{N}$.

After building L_A, N_A randomly selects among the initial $SP_i s$ one super-peer S to associate with, and uploads L_A to it. To maximize node diversity in the super-peers' stored lists, N_A randomly selects a super-peer in a different AS than itself. Using the aggregation procedure based on AS number described in section 6.3.2, S processes the host-related delay information it receives from its set of associated normal-peers. As a result, S obtains aggregated AS-related delay information. It maintains this AS-related delay information in a Local Entry Table (LET). To keep a manageable load, super-peers have a maximum number of associated normal-peers.

6.4.2 SPAD Reactive Information Exchange Scheme

Scheme Description

At the end of the above initialization phase, the SPAD module on N_A is ready to receive from the application layer, a QEAP discovery request towards another peer N_B. When N_A's SPAD module receives such request, it triggers the following Reactive Information Exchange Scheme. First, N_A evaluates the delay $delayDef$ on the default Internet path to N_B, and constructs a QEAP request of the form [$requestID$; src ; dst ; $delayDef$; TTL_Q], with $src = AS_{NA}$, $dst = AS_{NB}$, and TTL_Q the request's time-to-live. N_A then passes that request to its super-peer S, which executes the query-processing algorithm described in figure 6.8. Using this algorithm, a super-peer selects all entries within its LET that match the request, in order to generate its part of the response (line 6-11). Then if the request's $TTL_Q > 0$, it recursively forwards the request to all of its connected super-peers, and wait for their responses (line 12-17). Upon receiving the responses, it aggregates them with its own responses (line 15). It then executes a filtering process (line 18), before sending it back to the requesting peer (line 19). As described on figure 6.9, the filtering process removes redundant (line 3, 5-8) or irrelevant entries from the aggregate response (line 10, 12-15, 16-19). Thus, responses to a QEAP request travel back along the forwarding path towards N_A. Super-peers on this path successively filter these responses, decreasing their size, and returning only relevant entries.

When the search process is complete, N_A receives a list of entries of the form [AS_X ; AS_Y ; $delayAS_XtoAS_Y$; $hostInX$; $hostInY$], with $\{AS_X \text{ or } AS_Y\} = \{AS_{NA} \text{ or } AS_{NB}\}$. N_A first analyzes this list to search for "trivial" QEAPs, namely cases where the list contains information such as $delayAS_{NA}toAS_Z$ and $delayAS_ZtoAS_{NB}$, with their sum smaller than $delayDef$. If such cases exist, N_A sends query messages to these potential relay nodes first (the host in the $hostInZ$ field in the above example). Otherwise, it sends query messages in parallel to all the nodes $\{N_i\}$ in the $hostInX$ or $hostInY$ fields. These queries ask a given node N_i to evaluate/measure the delay on the remaining hop N_AtoN_i or N_itoN_B. Each N_i checks its resource availability to participate in a QEAP from N_A. If sufficient resources are available[3], N_i responds to N_A with the requested delay evaluation/measurement, and keeps a temporary resource reservation, waiting for N_A's QEAP selection. Non-willing $N_i s$ return an infinite delay value to N_A.

[3]This task involves an admission control function on the potential relay nodes, as illustrated in figure 6.3 and briefly discussed in section 6.5.

```
NOTE:
a_request: <ID, from, to , delay, TTL>
an_entry: <from, to, delay, hostFrom, hostTo>
a_responseList: {entry1, …, entryN}


1 sp_search(a_request req) returns a_responseList
2 {
3  if (req.ID is in {alreadyProcessed_RequestID})
4     return null
5  add req.ID to {alreadyProcessed_RequestID}
6  for all entry E in this_superpeer.LET
8    if (E.from == req.from) or (E.from == req.to)
9        or (E.to == req.from) or (E.to == req.to)
10          add E to responseList
11 end // for all entry E ...
12 if ( req.TTL > 0)
13    req.TTL = req.TTL – 1
14    for all superpeer S in {connected_superpeer}
15      responseList=responseList+S.sp_search(req)
16    end // for all super-peer
17 end // if request.TTL
18 filteredResponse = filter(responseList, req)
19 return filteredResponse    }
```

Figure 6.8 Pseudocode for the query-processing algorithm on SPAD super-peers.

```
1  filter(a_responseList list, a_request req)
                              returns a_responseList  {
2  for all entry E in list
3    if (E has a duplicate E' in list) remove E' from list
4    for all remaining entry G in list
5      if (E is of the form SRC/DST_to_X or X_to_SRC/DST)
6      and (G is of the form SRC/DST_to_Y or Y_to_SRC/DST)
7      and (X and Y are same AS number)
8          remove E from list
9    end // for all remaining entry G ...
10   if (E.delay > req.delay) remove E from list
11   for all entry F in this_superpeer.LET
12     if ( E is of the form SRC_to_X or X_to_SRC )
13         if (F is of the form X_to_DST or DST_to_X)
14             if ( E.delay + F.delay > req.delay )
15                 remove E from list
16     if ( E is of the form DST_to_X or X_to_DST )
17         if (F is of the form X_to_SRC or SRC_to_X)
18             if ( E.delay + F.delay > req.delay )
19                 remove E from list
20   end // for all entry F ...
21 end // for all entry E ...
22 return list    }
```

Figure 6.9 Pseudocode for the filter algorithm on SPAD super-peers.

When N_A receives back these delay values, it computes the overall delay on each alternate path. The paths with delays smaller than $delayDef$, are tagged as QEAPs. If such QEAPs exist, N_A uses the selection scheme from section 6.4.5 to choose the *best* one, according to some application QoS requirements. Then N_A notifies the corresponding relay node $N_{selected}$, and N_B. $N_{selected}$ commits the required resources and creates the necessary states to relay the traffic from N_A to N_B. N_A uses the QEAP and monitors the received QoS. Upon eventual QoS degradation, N_A can discover and use another QEAP. It is possible to optimize this operation by storing details of all the previously discovered QEAPs, using them as backup paths. At the end of the session, N_A notifies $N_{selected}$, and all the states and resources associated to the QEAP are released.

Discussion on the Reactive Information Exchange Scheme

The above scheme is based on query flooding over an unstructured P2P system. Theoretical lack of scalability is the major limit of such technique. However, as discussed in [107], the use of a super-peer scheme significantly extends further this limit. Moreover, several studies [116] present design methods, query-processing algorithms, and topology construction protocols aiming at improving scalability of super-peer systems. These techniques could be readily included in SPAD. Finally, other measurement studies suggest that unstructured P2P systems tend to self organize into power-law topologies [54]. In such topologies, node degrees follow a power-law distribution $P(k) \sim Ck^{-\alpha}$; with $P(k)$ the probability of a node to have k edges, α the power-law exponent, and C a normalizing constant. In [54], the authors show that networks with this topology characteristic scale more optimistically in regards to query flooding. Based on these studies, the experimentations on SPAD performances in chapter 7 only use power-law topologies to interconnect super-peers. Indeed, simulations based on other topologies such as pure-random or regular graphs, might not provide realistic results.

Assuming that the size of the candidate relay list is $log\mathcal{N}$, and that the number of super-peers $\mathcal{N}_S = \gamma\mathcal{N}$ (with $0 < \gamma < 1$), then the upper bound of the percentage of nodes which are potentially known to N_A at the end of the above reactive scheme is given by:

$$KnownNode = \frac{m(\gamma\mathcal{N})log\mathcal{N}}{\mathcal{N}-2} \qquad and \qquad \mathcal{N} \to +\infty \;\Rightarrow\; KnownNode \to +\infty \tag{6.2}$$

with m being the maximum number of normal-peers associated with any given super-peers. Expression (6.2) shows that as the community grows, the upper bound of the relative percentage of nodes known to N_A also increases. Thus comparing expression (6.2) with expression (6.1) shows that the proposed SPAD reactive scheme does not suffer from the same limitation as the simple scheme from section 6.2.

Section 7.3 presents a detailed performance evaluation of the proposed reactive information exchange scheme, using experiments based on the NLANR-AMP data sets.

6.4.3 SPAD Proactive Information Exchange Scheme

Scheme Overview

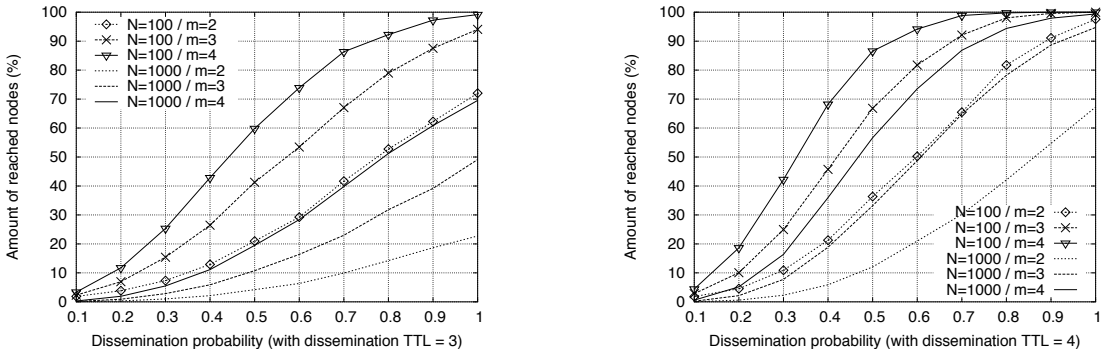
The reactive scheme described in section 6.4.2 uses a flooding-based technique, and is triggered by the reception of a QEAP request at a super-peer. Its performance, in terms of number of discovered QEAPs, is a function of the reach of the QEAP request, which is described in the evaluation section 7.3. Assuming a fixed number of super-peers with a fix minimum node degree, the reach of a request is a function of its TTL parameter (TTL_Q) [7]. Therefore, to increase the chances to discover QEAPs, a source node should set the TTL_Q of its QEAP requests to a high value. This implies more message forwarding between super-peers, thus increasing the message cost and the overall search latency for a given QEAP request. To address this limitation, SPAD features a proactive scheme, which is complementary to the previously described reactive scheme, i.e. SPAD super-peers execute both schemes within their *QEAP Discovery* module. The proposed reactive scheme allows super-peers to perform an information dissemination algorithm based on probabilistic flooding during their idle time. This algorithm allows the exchange of connectivity information, namely LET entries, between super-peers. As a result, a given super-peer potentially stores local copies of all the existing connectivity information relevant to its associated normal-peers. A QEAP request from a normal-peer then needs to be forwarded to only a few hops among super-peers (ideally one or two) in order to generate the same amount of relevant responses as in a SPAD system featuring only the previous reactive scheme. This allows the use of lower TTL_Q values in QEAP requests, thus resulting in a lower per-request message cost and search latency. The exact number of forwarding hops depends on the achieved dissemination state of the proposed algorithm, and is discussed further in the next subsections.

Probabilistic Flooding and Power-Law network

The simplest information dissemination strategy is the flooding technique. Unfortunately, this strategy requires significant bandwidth for large dynamic networks [120], which limits the scalability of the system. In contrast, it has been shown that percolation-based techniques, such as probabilistic flooding [121], provide efficient information dissemination with lower bandwidth usage than the classic flooding technique. Furthermore, these techniques do not require any membership management unlike the Gossip-based techniques [122]. In [123] the authors demonstrated that probabilistic flooding in power-law unstructured P2P networks is effective in terms of bandwidth consumption. Percolation-based random-walk also has low bandwidth requirements, but it achieves full information dissemination at a higher latency than probabilistic flooding, thus making it unsuitable for SPAD. Given these considerations, the probabilistic flooding strategy is an adequate candidate for SPAD's information dissemination scheme.

Probabilistic flooding derives from the bond-percolation theory [121]. In this scheme, a node forwards new generated or received information to its direct neighbors with a probability p, and drops that information with the complementary probability $(1 - p)$. It also drops already processed or forwarded information (i.e. *infect-and-die* policy per new information). Results from the bond-percolation theory demonstrate that for a given connected network graph, there exists a threshold dissemination probability $P_C < 1$ at which the information reaches all nodes at a minimum message cost. When this threshold is reached, the network is reduced to a minimum connectivity state that provides complete reachability without any redundant paths (i.e. the information does not reach a same node twice). Banaei-Kashani et al. [123] studied the application of that scheme to search (i.e. request forwarding) in power-law networks. They provided an analytic expression of P_C based on the power-law exponent α of the network being considered. However their expression assumed infinite request/information TTL. Extending their study, this section analyzes the dissemination probability in regard to the information propagation TTL and the amount of reached nodes. The following information propagation experiment was performed. First the topology generator BRITE [124] was used to generate 10^3 power-law topologies based on the Barábasi-Albert model (with α between 2.8 and 3). Then for each topology, a node was randomly selected as the source of an information propagation simulation.

Figure 6.10(a) and 6.10(b) show the amount of nodes reached by the information for different values of dissemination probability (P_D), dissemination TTL (TTL_D), number $\mathcal{N}$ of super-peers, and minimum node degree m. The comparison of these figures confirms two points: i) for a given network, propagation reach depends on both P_D and dissemination TTL_D; ii) higher minimum node degree allows faster information propagation. Figure 6.10(b) shows that the network diameter is 4 for power-law topologies with 100 nodes and $m = \{3; 4\}$. When $TTL_D = 4$ in a classic flooding strategy (i.e. $P_D = 1$), the information reaches 100% of the nodes. However, one can notice that the information has already reached "almost" all the nodes with a dissemination probability of $P_D = 0.6$ for $m = 4$, and $P_D = 0.8$ for $m = 3$. The same behavior stands also for the other topologies with $m = 2$ (not presented in this dissertation). These results demonstrate that in power-law networks, an information dissemination strategy using probabilistic flooding can achieve



(a) TTL=3 (b) TTL=4

Figure 6.10 Amount of reached nodes for different dissemination probabilities.

near full propagation at a lower message cost (20 to 40% less in the previous example) and a similar dissemination TTL than a classic flooding strategy. These results do not allow any conclusions on the scalability of probabilistic flooding. This is the subject of ongoing research studies [123].

Selection of P_D and TTL_D

The choice of P_D and TTL_D determines the information reach, and is a function of the characteristics of the super-peer network. This subsection presents an analytical study on the selection of these parameters. In [125], the authors provided a brief study of the average distance between any two nodes in random graphs. The following analysis is based on their problem formulation.

For simplicity, this analysis considers a graph of $\mathcal{N}$ nodes, with a fixed node degree $M+1$. The following equations remain valid for graphs with arbitrary node degree distribution, such as power-law distribution. In such case, $M+1$ would then represent the average node degree. Figure 6.11 shows such a graph (with $M+1=3$), where each node represents a SPAD super-peer. A node A is selected as the dissemination source of a new piece of information. The graph can then be represented as a tree structure (figure 6.11); where A is the root (rank 0), and where B, D, E (A's direct neighbors) are one level below (rank 1). Following the graph's topology, the direct neighbors of each node are added to the tree until all nodes are visited, with the only constraint that a tree node at rank k cannot have its parent (rank *k-1*) as one of its direct children (rank *k+1*). One can note that a given node from the graph can appear several times within the tree. For example, node C appears for the first time at rank 2, then many times at rank 3. This is due to the fact that a node has several neighbors in the graph; hence it could be the child of several parent nodes in the tree. Furthermore, one can also note that A has $M+1$ children nodes, and any node at rank k, with $k>0$, has M children nodes.

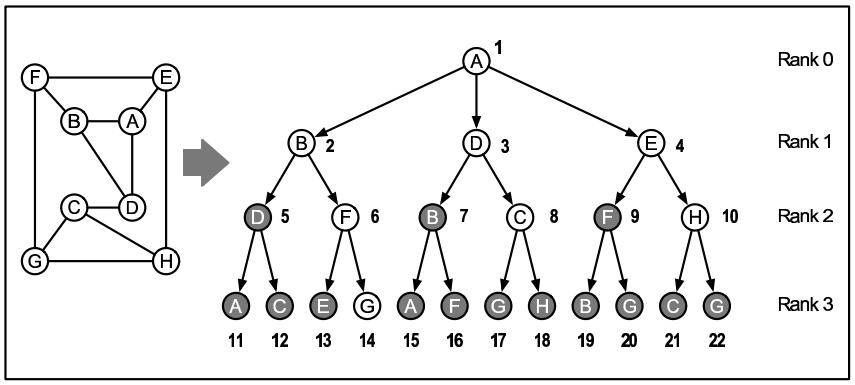


Figure 6.11 Graph and tree examples for the theoretical analysis of SPAD's information dissemination strategy.

In the next step of this analysis, the nodes are numbered according to their sequence of inclusion in the tree. For example, node A has the number 1, node C has the numbers 8, 12, 21. If t_k is the last node number at rank k, a classic flooding strategy yields:

$$t_1 = 1 + (M+1) \;\; ; \; ... \; ; \;\; and \;\; t_k = 1 + \sum_{i=0}^{k-1} (M+1)M^i \tag{6.3}$$

However, a probabilistic flooding strategy with a probability p gives:

$$t_1 = 1 + (M+1)p \;\; ; \; ... \; ; \;\; and \;\; t_k = 1 + \sum_{i=0}^{k-1} (M+1)M^i p^{i+1} \tag{6.4}$$

using iterative simplification on equation (6.4) and fixing $Mp \neq 1$ result in:

$$t_k = \frac{(M+1)M^k p^{k+1} - (p+1)}{(Mp-1)} \tag{6.5}$$

Let $f(t)$ be the total number of unique nodes in the tree after the t*th* node (N_t) has been added. For example, in figure 6.11, $f(4) = 4$, and $f(10) = 7$. Then $f(t_k) - f(t_{k-1})$ represents the number of unique nodes in the tree at rank k. Let δ_t be defined as:

$$\delta_t = f(t) - f(t-1) \tag{6.6}$$

with $\delta_t = 1$ if N_t is a new node, and $\delta_t = 0$ otherwise. The probability that N_t is a new node is then $P(\delta_t = 1)$, and:

$$P(\delta_t = 1) = \frac{\mathcal{N} - f(t-1)}{\mathcal{N}} \tag{6.7}$$

Let $E(t)$ be the expected value of $f(t)$, then taking the expectation on both side of equation (6.6) with the result from equation (6.7) gives:

$$E(t) - E(t-1) = \frac{\mathcal{N} - E(t-1)}{\mathcal{N}} \Rightarrow E(t) = 1 + \frac{\mathcal{N}-1}{\mathcal{N}} E(t-1) \tag{6.8}$$

solving (6.8) iteratively with $f(0) = 0$ produce the following result:

$$E(t) = \mathcal{N}\left(1 - \left(\frac{\mathcal{N}-1}{\mathcal{N}}\right)^t\right) \tag{6.9}$$

which represents the expected value of the total number of unique nodes in the tree after the inclusion of the t*th* node. From this result, one can estimate the number of unique nodes reached by a piece of information that was issued by node A and was disseminated through the graph using a probabilistic flooding strategy. Indeed for a given pair of dissemination probability P_D and time-to-live TTL_D (respectively corresponding to p and k), expression (6.5) gives a node number $t_{k=TTLD}$, that can be reported in expression (6.9) to obtain an estimation of the mean number of different reached nodes. Therefore in the proposed SPAD information dissemination scheme, with the assumptions that a given super-peer S knows $\mathcal{N}$ the maximum number of super-peers[4], and that it can estimate the average number of direct neighbors $M+1$; then using equations (6.5) and (6.9) it can compute estimates of P_D and TTL_D that would allow its information to reach, at a minimum message cost, a fraction x (as close to 1 as possible) of all the super-peers.

For instance, using expressions (6.9) and resolving $E(t) = \mathcal{N}$, a super-peer S would obtain a node number t. It would then report this number t into equation (6.5) to obtain a couple (P_D ; TTL_D). Equation (6.5) can provide multiple pairs of (P_D ; TTL_D) that achieve similar dissemination reach, but with different incurred message cost. For example in figures 6.10(a) and 6.10(b), the pairs (P_D=0.5 ; TTL_D=3) and (P_D=0.36 ; TTL_D=4) both provide a reach of 60% for a network with N=100 and m=4. In such cases, S uses the equation (6.12) from the next *discussion* subsection to compute the message cost incurred by each pair of solution, and selects the pair with the lowest message cost.

The following experiment was performed to verify the model provided by equations (6.5) and (6.9). First, 10^3 BRITE power-law topologies were generated, with $\mathcal{N} = 10^3$ nodes and a minimum node degree $m = 3$. Then several probabilistic flooding simulations were performed on these topologies with different P_D and TTL_D. For these topologies, the measured average network diameter is 5, and the average $M + 1 \approx 5.92$ (± 6.77). For TTL_D=5, figure 6.12 presents the comparison of the experimental propagation on the generated topologies and the theoretical propagation implied by expressions (6.5) and (6.9), using the measured value of $M+1$. According to these results, the estimation given by equations (6.5) and (6.9) of the average numbers of nodes in the network at the k*th* rank (e.g. $k = 5 = TTL_D$) is slightly smaller than the experimental values. Therefore, when using expressions (6.5) and (6.9) to determine optimal values of P_D and TTL_D, a SPAD super-peer obtains conservative values. As presented on figure 6.12, using P_D=0.6

[4] $\mathcal{N}$ could be a system-wide parameter, set during initial SPAD deployment and passed-on to new super-peers during their bootstrapping phase.

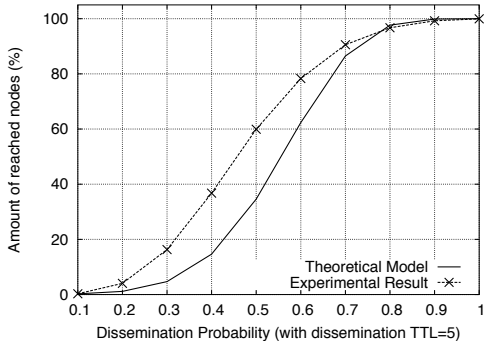


Figure 6.12 Comparison of information dissemination on experimental topologies versus theoretical model implied by equations (6.5) and (6.9).

and TTL_D=5 theoretically guarantees a reach of at least 60%; which is confirmed by experimentations showing an average reach of slightly less than 80%.

Integration of Probabilistic Flooding in SPAD

The previous subsection provides an analytical model (equations (6.5) and (6.9)) to dynamically compute P_D and TTL_D. This model is used in the following super-peer bootstrap mechanism to include the probabilistic flooding scheme into SPAD. The accuracy of this model depends on the value of $M+1$. In a stationary topology, where the number of super-peers remains constant after a given time, $M+1$ can be analytically computed using the topology's power-law distribution. This computation is described in the appendix A. However, in a dynamic P2P network, where super-peers join and leaves randomly, each super-peer needs to dynamically estimate the value of $M+1$. The following bootstrap mechanism, based on the strong law of large numbers, allows such an estimation. A new SPAD super-peer S_X retrieves the maximum number $\mathcal{N}_\mathcal{S}$ of allowed super-peers in its SPAD community (e.g. from an already existing super-peer such as one of its neighbors). It also measures m_{SX}, the number of its direct neighbor super-peers. Then it communicates its measured value m_{SX} to all of its direct neighbors S_I's, and receives back their measured values m_{SI}'s. Finally, S_X takes the average of its m_{SX} and all the received m_{SI}'s as its estimation of $M+1$. S_X uses the values of $\mathcal{N}_\mathcal{S}$ and $M+1$ as input parameters into the expressions (6.5) and (6.9). This bootstrap mechanism allows a new SPAD super-peer S_X to dynamically compute a couple (P_D ; TTL_D) that theoretically ensures a complete dissemination of its information among the other super-peers. Furthermore upon receiving the value m_{SX} from S_X, each S_I updates its estimation of $M+1$, and recalculates its parameters P_D and TTL_D. This allows dynamic updates of these parameters as the topology of the super-peer network evolves.

After accessing the status of super-peer and completing the related initialization tasks (e.g. association with normal-peers and LET construction, as described previously), a node S uses its idle time to perform the functions associated with the dissemination scheme. S is idle whenever it is not processing a QEAP request. As mentioned earlier, the entries

```
NOTE:
    an_entry: <ID, ASFrom, ASTo, delay, hostinFrom, hostinTo>

1   process_incoming_entry (an_entry E)
2   {
3    if (E.ID is in {already_processed_entry_ID})
4       discard E and exit
5    if ({already_processed_entry_ID}.size > threshold)
6       discard {already_processed_entry_ID}.older_element
7    add E.ID to {already_processed_entry_ID}
8    add E to {entries_to_forward_via_dissemination_scheme}
9    for all entry F in this_superpeer.LET
10      if ((E.ASFrom == F.ASFrom) || (E.ASFrom == F.ASTo)
11          || (E.ASTo == F.ASFrom) || (E.ASTo == F.ASTo))
12          add E to this_superpeer.RET
13  end // for all entry F
14  for all entry G in this_superpeer.RET
15      if ((E.ASFrom == G.ASFrom) && (E.ASTo == G.ASTo))
16          add E.hostinFrom to the list G.hostinFrom
17          add E.hostinTo to the list G.hostinTo
18          update G.delay and exit
19  end // for all entry G
```

Figure 6.13 Pseudocode of the Incoming Entry Processing algorithm executed by SPAD super-peers.

in the LETs of super-peers are the information to propagate. The first step for S is to communicate its LET to its directly connected neighbors. S initially sends its entire LET. When its LET subsequently changes due to updates from associated normal-peers, S will send only the modified entries to its neighbors. LET propagation is done using the previously introduced probabilistic dissemination technique. The second step is the processing of incoming LET entries, as described in figure 6.13. After receiving an incoming entry E, S verifies that it has not already processed it earlier (line 3-7), and adds it to the list of entries to forward via the dissemination scheme (line 8). Then S compares E with the entries from its own LET (line 9-13). If E matches one of these entries, then it contains information potentially relevant to future QEAP requests from S's associated normal peers. Therefore, S retains E and stores it in a Remote Entry Table (RET). S also compares E to its existing RET entries, to determine if E can potentially provide more information to one of them (line 14-19). The whole matching procedure (line 9-19) aims at selecting only incoming entries that contain information related to S's associated normal-peers. S does not propagate its RET entries to its neighbors. Indeed, via LET propagation, they may already have received the raw information on which they would probably have made a different selection than S. As dissemination progresses, S stores locally more and more entries (from further super-peers) relevant to its associated normal-peers. Each super-peer fixes the size of its own RET based on its available resources.

When S receives a QEAP request from N_A (one of its normal-peers) towards another node N_B, it processes it as described in section 6.4.2 with the following two modifications. First, S includes its RET entries in the selection process that builds its part of the response R_S. Therefore, its part of the response R_S contains a set of matching entries from both its LET and RET, these entries are of the form $[AS_U\ ;\ AS_V\ ;\ delayUtoV\ ;\ hostInU\ ;\ hostInV]$ where $\{U$ or $V\}$ is the AS of either node N_A or N_B. The second modification concerns the request forwarding to other super-peers. Depending on the state of the dissemination process, S might already have in its RET all the existing information relevant to the request. In such case, S would not need to forward the request to its neighbor super-peers. However, due to the dynamic character of the peer-to-peer SPAD community, and the probabilistic nature of the dissemination scheme, this ideal case might not be reached.

Therefore, S might still need to forward the request, but with a lower request TTL (TTL_Q) than the forwarding process of section 6.4.2.

The following simple method presents an example of such a decision process. S constructs $ListAS_S$, the list of distinct ASes in its LET and RET. Then $|ListAS_S|$ is a coarse lower bound of the total number of ASes involved in the SPAD community. Based on $ListAS_S$, a coarse lower bound of the number of potential relevant response entries in the community is given by:

$$NumResp_S = \begin{cases} 2 * (|ListAS_S| - 2) & \text{if (AS of } N_B) \in ListAS_S \\ 2 * (|ListAS_S| - 1) & \text{otherwise} \end{cases} \tag{6.10}$$

Then:

- if $|R_S| \geq \beta * NumResp_S \Rightarrow$ S decides that its local part of the response already contains enough entries, and does not forward the request
- else S forwards it with a $TTL_Q = \begin{cases} \gamma * \lceil NumResp_S/|R_S| \rceil & \text{if } |R_S| \neq 0 \\ \gamma & \text{if } |R_S| = 0 \end{cases}$

with $0 < \beta \leq 1$ and $\gamma \in \mathbb{N}^*$, a couple of system-wide parameters (e.g. $\beta = 0.7$ and $\gamma = 3$). Other decision techniques can be used to evaluate with more precision the state reached by the information propagation (e.g. techniques based on Markov decision processes). The remaining of the request processing, the response processing, and the QEAP set-up and utilization are unchanged compared to section 6.4.2.

Discussion on the proactive information exchange scheme

This thesis claims that the proposed proactive scheme provides gains in search message cost and latency. One can argue that these gains are balanced by the dissemination message cost and latency, thus implying no real improvement to SPAD. This subsection proposes a simple analysis to address this concern. A normal-peer N_A issues a QEAP request and passes it on to its associated super-peer S_1. This analysis assumes that there exists a piece of information J that is relevant to this QEAP request. Then J is necessarily stored on S_X, one of the existing super-peers, and the probability of finding J is $P = 1/\mathcal{N}_\mathcal{S}$, with $\mathcal{N}_\mathcal{S}$ being the number of super-peers. Without the proposed information dissemination scheme in SPAD, N_A would successfully access J only if its QEAP request reaches S_X. Using the QEAP processing reactive scheme from section 6.4.2, this would incur an average message cost C_1 among super-peers of:

$$C_1 = (M + 1) \sum_{k=2}^{\mathcal{D}} M^k \tag{6.11}$$

with $M + 1$ being the average super-peer degree, and $\mathcal{D}$ being the network diameter of the SPAD overlay network. In a SPAD community with the proposed information dissemination scheme, J would be disseminated from S_X to all the remaining super-peers. It would eventually reach S_1, with a message cost C_2 between super-peers of:

$$C_2 = p(M + 1) \sum_{k=2}^{TTL_D} p^k M^k \tag{6.12}$$

The previous analysis and results on P_D and TTL_D concluded that $p < 1$ and $TTL_D \leq \mathcal{D}$. Therefore, the proposed proactive scheme provides a gain in term of message cost between super-peers.

Let $C_3 = K\mathcal{L}$ be the average latency cost to access J, with $\mathcal{L}$ being the average latency on a hop between two super-peers, and K being the number of hops required to reach S_X from S_1. Without the dissemination scheme, K is equal to $\mathcal{D}$, whereas with the dissemination scheme, J would gradually reach super-peers closer to S_1 (and eventually S_1 itself), resulting in $K \leq \mathcal{D}$. Therefore to access J, the proposed proactive scheme may provide a gain in latency, depending on the advancement of the information dissemination process. Since super-peers perform the proactive information exchange scheme as a background task, this analysis does not consider the information dissemination latency.

Furthermore, the information gathered by a super-peer via the proposed dissemination algorithm can potentially be used to reply to multiple QEAP requests from its associated normal-peers. For example in the above scenario, once J arrives at S_1, it can not only be part of the response to N_A's QEAP request, but it can also be part of the responses to subsequent QEAP requests issued by S_1's associated normal-peers, hence a further gain in message cost and latency. Finally, as stated earlier due to the dynamic behavior of SPAD's peer-to-peer network and the probabilistic nature of the dissemination algorithm, a state of complete information dissemination might not be achievable among SPAD super-peers. However even with partial dissemination, the evaluations in the next chapter show that the proposed scheme still provides significant gains.

Section 7.4 presents a detailed performance evaluation of the proposed proactive information exchange scheme, using experiments based on the PlanetLab data sets.

6.4.4 Integration of Loss-QEAP Search in SPAD

The previous SPAD schemes were designed to support the integration of new QoS parameters in their QEAP discovery mechanisms with minimal modifications to the overall system. To illustrate such integration, this subsection presents the set of required modifications, which add *Loss-QEAPs* discovery capabilities to the SPAD system.

First, this integration requires the addition of a packet-loss measurement mechanism to the SPAD component within each peer. As introduced in section 6.3.1, each SPAD peer collects some QoS information about its connectivity to a small number of other peers. In the previously described SPAD system, the *Measurement* module (figure 6.3) had only a delay measurement mechanism. Measuring or estimating end-to-end packet-loss rate in an accurate, unobtrusive, and efficient manner is an open research issue. However recent contributions [126] indicate that measurement/estimation tools with these properties are becoming available. In [126], the authors proposed a mechanism based on a refined TCP retransmission count to accurately estimate end-to-end packet-loss rates in a novel way. It is assumed that the *Loss-QEAP* capable SPAD system will be able to use such tools within its *Measurement* module.

Second, as mentioned in section 5.4.3, end-to-end packet-loss rates are constant on a scale of less than 10 minutes, as opposed to 10-30 minutes for end-to-end delays [113]. Therefore,

the mechanism responsible for triggering measurement updates and their propagation to the SPAD super-peers needs to account for this difference in constancy durations.

Finally a major difference between *Delay-QEAPs* and *Loss-QEAPs* is the computation of the value of the QoS parameter over an entire QEAP, which is composed of consecutive end-to-end paths. For a *Delay-QEAPs*, the overall delay is the sum of the single delays on the component paths, i.e. delay is an additive metric. However for a *Loss-QEAP*, the computation of the overall packet-loss rate is performed using the complement of the packet-loss rate, which is a multiplicative metric [83]. For example, if Q_{AXB} is a *Loss-QEAP* from hosts A to B via a relay host X, and P_{AX} and P_{XB} are the packet-loss rates on the respective paths $[A \rightarrow X]$, and $[X \rightarrow B]$, then P_{AXB} the packet-loss rate on Q_{AXB} is defined as follows:

$$(1 - P_{AXB}) = (1 - P_{AX}) * (1 - P_{XB})$$

$$\Rightarrow P_{AXB} = 1 - (1 - P_{AX}) * (1 - P_{XB}) \tag{6.13}$$

When searching for a QEAP between two peers, the modified SPAD discovery schemes will use this expression (6.13) to compute the overall packet-loss rates on the potential alternate paths. It will then compare the resulting values with the packet-loss rate on the default path to determine if a given alternate path is a QEAP.

6.4.5 SPAD QEAP Selection Schemes

Given the added capability of discovering alternate paths that provide enhanced delay and/or packet-loss rate, a SPAD peer N_A will have a larger number of discovered QEAPs that it can potentially use to forward the traffic of one of its application $\mathcal{A}$. Table 5.2 shows that when N_A has some existing QEAPs towards another peer N_B, it can potentially discover on average 7.5 ($\pm$0.5), 10.7 ($\pm$1.0) or 38.4 ($\pm$2.6) of them, depending on the considered data set. The selection of the *best* QEAP depending on the provided delay gain, packet-loss gain, and the QoS requirements of the application $\mathcal{A}$, is important, as it greatly determines the user's QoS perception. This section proposes different selection schemes to address this issue. Section 7.5 presents a detailed performance evaluation of these different selection schemes, using experiments based on the NLANR-AMP and PlanetLab data sets.

There are 2 types of Application Program Interface (API) available to the applications that run above the SPAD framework. The first one is a *transparent* API. It is used when an application $\mathcal{A}$ is not aware of the existence of a SPAD module on its host machine. $\mathcal{A}$ does not know that its traffic towards a distant application $\mathcal{B}$ might potentially be forwarded over a QEAP discovered and managed by SPAD, when such QEAP exists and is available. In this case, SPAD does not receive any specific QoS requirement from the application layer, it will search for all QEAP towards the requested destination, and it will select the one to use according to one of the schemes from this section. For this type of API, the choice of the selection scheme to use is either arbitrary or user driven. In fact, although the application is not aware of SPAD, the user might be. In such cases, he/she can configure the SPAD module to use a given type of QEAPs for a given type of traffic (e.g. based on port number, or RTP profiles). The method-calls of this *transparent* API are similar to the

ones of *well-known* API, such as the Unix "Socket" API, and are accessible via dynamic shared libraries. This *transparent* API allows legacy applications to use SPAD's QEAP discovery service with no or minimal changes.

The second type of API is used by applications that are aware of the availability of a SPAD module on their host machines. These applications know that they can potentially benefit from using QEAPs to forward their traffic. In this case, when $\mathcal{A}$ running on node N_A wants to send some traffic to $\mathcal{B}$ running on node N_B, it builds a QEAP request for the SPAD module on N_A. This request is defined as follows:

$$Req_{\mathcal{A}} = [ID, src, dst, D_{REF}, L_{REF}, QoS_{SPEC}] \tag{6.14}$$

with ID = a unique identification for this request, $src = N_A$, $dst = N_B$, D_{REF} and L_{REF} the delay and packet-loss rate on the default Internet path between N_A and N_B, and QoS_{SPEC} defined as follows:

$$QoS_{SPEC} = [D_{REQ}, L_{REQ}, QoS_{PREF}] \tag{6.15}$$

with D_{REQ} and L_{REQ} = the maximum required delay and packet-loss rate required on a QEAP, i.e. for a discovered QEAP Q_x with delay D_{Qx} and packet-loss L_{Qx} we have:

$$\begin{cases} D_{Qx} \leq D_{REQ} \leq D_{REF} \\ and/or \\ L_{Qx} \leq L_{REQ} \leq L_{REF} \end{cases} \tag{6.16}$$

The parameter QoS_{PREF} can take one of the following values {D, L, T, DT, LT}. The application $\mathcal{A}$ selects one of these values according to the type of QEAP that it requires. The following subsections discuss the meaning of each value.

After receiving $Req_{\mathcal{A}}$ from the application $\mathcal{A}$, the node N_A uses the SPAD discovery mechanisms to build a set $\mathcal{Q}$ of k discovered candidate QEAPs. The set $\mathcal{Q}$ is defined as: $\mathcal{Q} = \{Q_i\}_{1 \leq i \leq k}$, with D_{Qi} and L_{Qi} following the same inequalities as D_{Qx} and L_{Qx} in equation (6.16). Depending on the value of the QoS_{PREF} parameter, the SPAD module on N_A executes a particular QEAP selection scheme to select the *best* QEAP from $\mathcal{Q}$ to use between N_A and N_B.

When QoS_{PREF} = D or L

This is the case when $\mathcal{A}$ would like to use the discovered QEAP that offers the highest gain in delay (i.e. D) or the one with the highest gain in packet-loss rate (i.e. L). The QEAP Q_{sel} that will be used for $\mathcal{A}$'s traffic, is selected as follows:

$$\text{if } QoS_{PREF} = \mathsf{D}: \quad Q_{sel} = Q_x \in \mathcal{Q} \mid D_{Qx} = min(D_{Qi})_{\ 1 \leq i \leq k} \tag{6.17}$$

$$\text{if } QoS_{PREF} = \mathsf{L}: \quad Q_{sel} = Q_x \in \mathcal{Q} \mid L_{Qx} = min(L_{Qi})_{\ 1 \leq i \leq k} \tag{6.18}$$

with D_{Qx} and L_{Qx} the delay and packet-loss rate on a given QEAP Q_x.

When $QoS_{PREF} = \mathsf{T}$

This is the case when the application $\mathcal{A}$ would like to use the discovered QEAP that offers the highest possible throughput. In [127], the authors propose a model to accurately estimate the achievable throughput of a TCP-friendly flow on a path with a given delay and loss probability. Assuming that a TCP-friendly congestion control module is available within the Middleware framework for *overlay application* (as illustrated in figure 4.4), and that this module cooperates with the SPAD module to ensure that the traffic on a given QEAP Q_x has a TCP-friendly behavior, then the model from [127] can provide an estimate of the throughput T_{Qx} that could be achieved on this QEAP. This throughput estimate is given by the following equation:

$$T_{Qx} = \frac{s}{D_{Qx}\sqrt{\frac{2L_{Qx}}{3}} + 4D_{Qx}\sqrt{\frac{3L_{Qx}}{8}}L_{Qx}(1+32L_{Qx}^2)} \tag{6.19}$$

with s being the size of a data packet. To derive this expression from [127], this analysis assumes that the TCP-friendly traffic on a QEAP is equivalent to a TCP flow with a Retransmission Timeout (RTO) equal to $4 * RTT$, and for which an ACK acknowledges every packet. Furthermore, to allow the use of L_{Qx} as an estimate of the loss probability on Q_x, this analysis also assumes a uniform loss distribution, a quasi-constant RTT value, and a maximum of one packet loss per RTT. Finally, for simplicity, s is fixed to 1460 bytes, which is a typical value of TCP's Maximum Segment Size. Using equation (6.19), the QEAP Q_{sel} that will be used for the traffic of application $\mathcal{A}$, is selected as follows:

$$Q_{sel} = Q_x \in \mathcal{Q} \;\mid\; T_{Qx} = max(T_{Qi})_{\;1\leq i\leq k} \tag{6.20}$$

According to equations (6.19) and (6.20), the application $\mathcal{A}$ can also use the parameter $QoS_{PREF} = \mathsf{T}$ to select the QEAP that offers both the lowest possible delay and packet-loss rate, among the discovered ones. Although such scheme minimizes both delay and packet-loss rate, it is biased towards the packet-loss rate as shown on equation (6.19) and figure 7.6 from section 7.5.

When $QoS_{PREF} = \mathsf{DT}$ or LT

This is the case when the application $\mathcal{A}$ would like to use the discovered QEAP that offers both the highest possible delay gain and throughput (i.e. DT), or both the highest possible packet-loss rate gain and throughput (i.e. LT). Let U_{Qx} be the utility function of a given QEAP Q_x, and defined as follows:

$$U_{Qx} = \alpha\; G_D(Q_x) + (1-\alpha)\; G_L(Q_x) \tag{6.21}$$

$$with \begin{cases} 0.5 < \alpha < 1 & \text{if } QoS_{PREF} = \mathsf{DT} \\ 0 < \alpha < 0.5 & \text{if } QoS_{PREF} = \mathsf{LT} \end{cases} \tag{6.22}$$

with $G_D(Q_x)$ and $G_L(Q_x)$ being the delay gain and packet-loss gain that are provided by Q_x, defined as follows:

$$G_D(Q_x) = \frac{D_{REF} - D_{Qx}}{D_{REF}} \qquad \text{and} \qquad G_L(Q_x) = \frac{L_{REF} - L_{Qx}}{L_{REF}} \tag{6.23}$$

Depending on the value of QoS_{PREF}, the SPAD module fixes the value of α, which determines the utility associated to each Q_i. For a given set $\mathcal{Q}$, equations (6.16), (6.21) and (6.22) lead to a discrete linear optimisation problem. Solving this problem for a particular set $\mathcal{Q}$ gives an α value, which allows the selection of the Q_x that best meet the required QoS (e.g. highest delay gain and throughput). Based on experimental results, this thesis propose the following heuristic values: $\alpha = 0.8$ if $QoS_{PREF}=$ DT, and $\alpha = 0.2$ if $QoS_{PREF}=$ DT. Using equation (6.21), the QEAP Q_{sel} that will be used for $\mathcal{A}$'s traffic, is selected as follows:

$$Q_{sel} = Q_x \in \mathcal{Q} \mid U_{Qx} = max(U_{Qi})_{\ 1 \leq i \leq k} \tag{6.24}$$

This selection scheme uses a linear utility function U_{Qx} to optimize both delay and throughput, or both packet-loss rate and throughput. However, as previously stated, the throughput computation is more sensitive to the packet-loss parameter. Therefore, other types of utility function (i.e. non-linear) might provide a better optimization of both throughput and delay/packet-loss rate. The study of such utility functions is not within the scope of this thesis and can be the subject of future research works.

6.5 TOWARDS A COMPLETE SPAD SYSTEM

The previous section described the main 3 schemes that constitute the QEAP discovery and selection functionalities of the proposed SPAD architecture. These schemes collectively propose a solution to the problem that this thesis aims at addressing, namely the provision in a distributed manner of enhanced QoS between the hosts of an *overlay application* (section 2.2). Some other functionalities can be included to this contribution to obtain a complete and fully deployable SPAD architecture. These functionalities require further investigations and do not pertain to the problem scope of this thesis. This section briefly discusses some examples of such functionalities and the related issues.

When a SPAD peer R receives a request to be a *relay* node in a QEAP Q, it has to determine if it has enough available resources to relay the data packets on Q, while i) guaranteeing the data QoS requirements, and ii) ensuring that QoS characteristics of other current relayed traffic remain satisfactory. An admission control scheme within the SPAD *Admission Control* module (figure 6.3) performs such a functionality. An admission control scheme requires the following elements:

- a model of the traffic characteristics on a given call/connection (i.e. a given QEAP in the SPAD system)
- a model of the connection request arrival and departure (i.e. QEAP requests in the SPAD system)
- a set of admission policies or criteria

The set of admission policies/criteria are used to determine the acceptance or rejection of a given call/connection request, based on its traffic characteristics and the currently available and utilized resources on the node. The performance of an admission control scheme can

be measured as the acceptance probability of a call/connection, considering a particular arrival and departure model. Several admission control schemes have been proposed in various context such as Asynchronous Transfer Mode (ATM) networks [128], integrated service networks [129], or wireless networks [130]. The SPAD *Admission Control* module can use a scheme based on one these various contributions.

The monitoring of utilized QEAPs, and the eventual failure recovery strategy to adopt are other examples of functionalities that can be added to the SPAD system prior to its deployment. As introduced in 6.3.1 and 6.4.4, some recent research contributions [117, 126] propose some techniques to monitor QoS parameters on a given QEAP without interfering with the data traffic being sent on it. A QEAP monitoring function within the SPAD module (figure 6.3) can use such techniques, and trigger a failure recovery function upon QoS degradation on the monitored QEAP. Various strategies can be adopted for such a recovery function. For example after selecting a QEAP to utilize between two hosts, a SPAD module can temporarily store a fixed number of other QEAPs (previously discovered during the QEAP search phase), and use them as backup when the main selected QEAP fails to maintain the required QoS. Such a strategy provides a short failure recovery latency, but requires that a SPAD peer maintains more states for each utilized QEAP, thus limiting the scalability of this scheme.

Finally, the SPAD system as described in this chapter only provide functions to enhanced the QoS between any two given host of an overlay application. However, a typical overlay application S, as illustrated in figure 3.1, involves more than 2 peers. Thus, a complete SPAD architecture has to provide enhanced QoS to the entire association of hosts in S. Such a goal can be achieved by composing individual QEAPs between the hosts in S, according to the properties and the composition rules of QoS parameters, as discussed in [83]. For example, the QoS parameters *latency* and *bandwidth* are additive and concave, respectively; and Wang and Crowcroft [83] has demonstrated that it is feasible to design a QoS routing scheme with a polynomial complexity, which optimizes both an additive and a concave parameter (e.g. composition of delay and bandwidth parameters). The SPAD module (figure 6.3) can use these results to select and compose individual QEAPs to provide enhanced QoS to an entire overlay application.

6.6 CHAPTER SUMMARY

This chapter presented the main contribution of this thesis, namely SPAD (Super-Peer based Alternate path Discovery). SPAD is a distributed system that enhances the QoS between two given peers of an *overlay association*. To achieve this goal, SPAD discovers, selects, and utilizes alternate composite Internet paths with enhanced QoS characteristics (i.e. QEAPs) between these peers.

This chapter first introduced an initial simple distributed scheme that assessed the feasibility of discovering QEAPs in a distributed manner within a cooperative environment. Then it presented the proposed SPAD system, starting with an overview of its architecture and the related design assumptions, and following with detailed descriptions of its QEAP discovery schemes and its QEAP selection scheme. For simplicity, these descriptions were first focused on *Delay-QEAP*, then section 6.4.4 introduced the minor modifications re-

quired to include *Loss-QEAP* discovery and selection capabilities. Finally, this chapter briefly discussed some examples of other functionalities that can be included to a complete deployable SPAD architecture.

The proposed SPAD system has the following properties. First, it is a fully distributed system with no single point of failure, where each entity cooperates as an equal peer. More precisely, SPAD is based on an unstructured super-peer system, where cooperative super-peers process QEAP queries issued by their associated normal-peers. These normal-peers receive back incrementally-gathered information on nodes that could potentially act as relays in QEAPs. Second, SPAD is located within a middleware framework, which resides on end-host machines. Therefore, it can be easily and incrementally deployed, as it does not require any change to the current Internet routing mechanisms, or core network components, such as routers inside Autonomous Systems (ASes). Given these properties, SPAD is an adequate solution to the provision of enhanced QoS between the elementary application components that compose an overlay application.

Chapter 7

SPAD Performance Evaluation

7.1 INTRODUCTION

This chapter presents and discusses the performance evaluation of the QEAP discovery and selection schemes, as described in the previous chapter 6. These evaluations are based on experiments with the measurement data sets from the RIPE-TTM, the NLANR-AMP, and the PlanetLab All-Pair-Ping projects, as introduced in section 5.3.

Section 7.2 presents the evaluation results of the simple preliminary QEAP discovery scheme from section 6.2. Then, section 7.3 and 7.4 provides the performance results of the SPAD reactive and proactive information exchange schemes, as described in section 6.4.2 and 6.4.3, respectively. Section 7.5 compares and evaluates the different QEAP selection schemes from section 6.4.5. Finally, section 7.6 discusses some limits of the performed experiments.

The various experimental results from this chapter collectively show that the proposed SPAD system is efficient in:

- discovering in a distributed manner existing QEAPs within communities of cooperative peers on the internet,
- selecting the QEAP that *best* meets some application QoS requirements, among the set of discovered QEAPs.

Thus, this chapter validates the approach and solution that this thesis proposes to address the problem of providing enhanced QoS to peers within an *overlay association*.

Table 7.1 Performance of the Simple QEAP Discovery Scheme (RIPE-TTM 25/05/04)

QEAP discovery algorithm	Message cost	Percentage of discovered *Delay-QEAPs* among all existing ones		
		Number of bootstrap nodes		
		2	3	4
Brute-force search	$2.\mathcal{N}$	100%	100%	100%
Search within a list of random candidate relay nodes	$2.log\mathcal{N}$	46.2%	46.1%	46.3%
Only first search-level of the simple QEAP discovery scheme	$2.log\mathcal{N}$	61.8%	71.1%	76.1%
Complete simple QEAP discovery scheme (first and second search-levels)	$2.log\mathcal{N}$ to $2.(1 + 2.log\mathcal{N})$	77.6%	82.8%	88.0%

7.2 EVALUATION OF A SIMPLE QEAP DISCOVERY SCHEME

The experiments within this section were performed on the 25/05/04 RIPE-TTM data set. The 12/07/04 RIPE-TTM data set provided similar results, not presented in this dissertation. Since the considered data set contains 47 nodes, the size of the candidate relay list in the following experiments is fixed to 6 (i.e. $log_2\ 47 \approx 6$), as described in section 6.2.2.

The following experiments were performed to evaluate the simple QEAP discovery scheme from section 6.2. First, 10^3 pairs of nodes were randomly selected at a given time stamp within the data set. For each pair $(N_X; N_Y)$, 2 to 4 bootstrap nodes were randomly selected, and the corresponding candidate relay lists L_X and L_Y were built, as described in section 6.2.2. Then for each category of bootstrap node number (i.e. 2, 3 and 4), the first search-level of the simple QEAP discovery scheme was performed to discovery any existing QEAP between all the pairs of nodes. The average success rate for this step was recorded. The second search-level was then performed on the remaining pairs for which no QEAPs were previously discovered. The average success rate for this search-level is added to the success rate of the first search-level to obtain the total success rate of the proposed simple QEAP discovery scheme. Finally, these procedures were repeated 10 times for each pairs to obtain an average that takes in account the randomly selected bootstrap nodes. Table 7.1 presents the results of these experiments. As a reference comparison, the two first rows of this table display the results for a *brute-force* QEAP search algorithm, similar to the one used in section 5.4.1, and the results for a modified first search-level with candidate relay lists which contain randomly selected nodes. The third row shows the results for only the first search-level of the evaluated scheme. The fourth row shows the results for the full QEAP discovery scheme (both search-levels).

As expected, the *brute-force* search algorithm finds all the existing QEAPs, hence the 100%. The second row shows that randomly selecting the nodes to include in the candidate list of N_X results in the discovery of only about 46% of the existing QEAPs by the first search-level. The third row shows that on average, for a number of 4 bootstrap nodes, the first search-level manages to discover 76.1% of the existing QEAPs. This number grows up to 88% for the complete scheme. Thus, if some alternate paths with enhanced

QoS exist for a given communication between two hosts within the RIPE-TTM data set, the proposed scheme discovers at least one of these alternate paths in about 88% of the cases (for an initialization phase with 4 bootstrap nodes). This result offers a satisfactory performance/complexity trade-off considering the communication complexity of the *brute-force* search and the poor performances of the random search. From table 7.1, one can also infer that for 4 bootstrap nodes, in about 86.5% of the successful QEAP discovery, the message cost is equal to $2.log\,\mathcal{N}$. This cost is equal to $2.(1+2.log\,\mathcal{N})$ in the remaining 13.5% of the successful QEAP discovery.

7.3 EVALUATION OF SPAD REACTIVE SCHEME

The performance evaluation of the SPAD reactive scheme is based on the following experiments. From the available 107 nodes in the NLANR-AMP data sets, 10 nodes were randomly selected to act as super-peers. The remaining 97 nodes were evenly associated as normal-peers to these 10 super-peers, and the size of their candidate relay list (RList) was set to 6. These RLists and the derived LETs on each super-peer were initialized as described in section 6.4.1. Then, the BRITE [124] topology generator was used to connect the super-peers in a power-law topology based on the Barábasi-Albert model. Finally, a random pair of nodes $(N_X; N_Y)$ was selected at a random time stamp, and different schemes were performed to discover any existing QEAPs between these nodes. The following results are averaged over 10^4 trials.

Figure 7.1 presents a performance comparison between the SPAD reactive QEAP discovery scheme, the former simple QEAP discovery scheme (SIMP) from section 6.2, and a random search scheme (RAND) where a node looking for a QEAP queries a fixed-size list of random candidate relay nodes. The Performance axis corresponds to the percentage of existing QEAP that are discovered by each method. The SPAD QEAP discovery was performed first for values of TTL_Q between 1 and 4 (4 being the diameter of the experimental super-peer network). Then the average sizes of the response list for each TTL_Q category were recorded, and these values were used to set the sizes of the candidate relay lists in the SIMP and the RAND schemes. For a $TTL_Q > 1$, the SPAD reactive scheme significantly outperforms the 2 other methods, discovering 77.58% ($\pm$ 0.5726) of existing QEAP for $TTL_Q = 4$. The performance of SPAD is function of the reach of a query, i.e. the more super-peers a query reaches, the more relevant information will be received by the source node, and the more potential relay nodes will be known to the source. Yang and Garcia-Molina [7] established that the reach, in a power-law super-peer network, is a function of the existing number of super-peers, their average node degree, and the request's TTL_Q. They provided an extensive study of the influence of these parameters on the scalability of such networks.

Table 7.2 presents the search cost in terms of messages being exchanged between super-peers, and latency related to these exchanges. The latency cost does not include processing delay within super-peers. As expected, both costs increase with TTL_Q. The latency associated with the chance of discovering 77.10% of existing QEAPS is still close to 200ms. For several applications, such as Voice-over-IP, this is a reasonable connection set-up time, since the resulting QEAP would probably be used for several minutes, and would provide

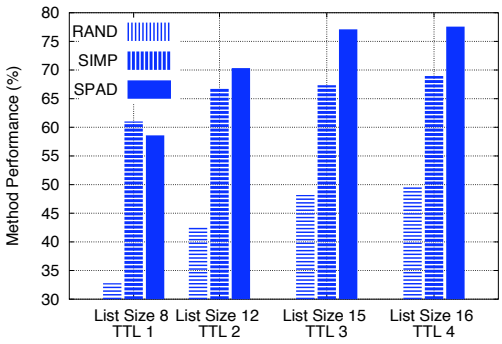


Figure 7.1 Performance comparison between SPAD reactive scheme and other QEAP discovery schemes (NLANR-AMP 30/01/03).

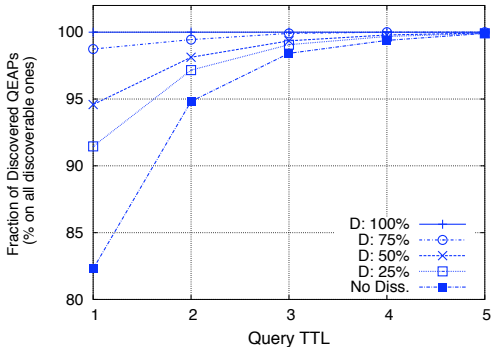
Table 7.2 Performance of the SPAD Reactive Scheme (NLANR-AMP 30/01/03)

TTL	Percentage of discovered *Delay-QEAPs* among all existing ones	Search Cost (message number)	Search Cost Latency (ms)	Size of the Response List (number of entries)	
				With Filter	Without Filter
1	58.59%	2.85	97.6	8.26	11.18
2	70.33%	6.12	160.8	11.98	17.22
3	77.10%	9.28	208.4	15.22	22.75
4	77.58%	10	222.1	15.87	23.86

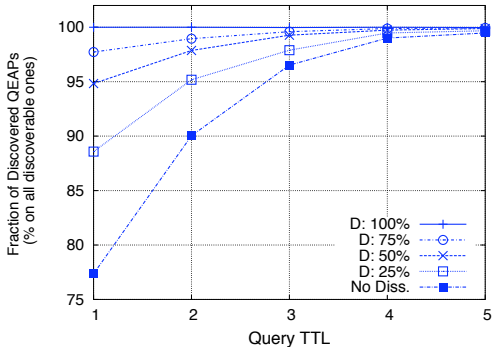
better delay than the default path for that duration. Table 7.2 also shows the average size of responses generated by a request for a version of the SPAD reactive scheme with response filtering on super-peers (as described in figure 6.9) and a version without filtering. Using response filtering on super-peers significantly decreases the size of responses sent back to the originator of a request, thus decreasing the associated bandwidth cost.

7.4 EVALUATION OF SPAD PROACTIVE SCHEME

The following experiments were performed to evaluate the performance of the SPAD proactive scheme. Multiple simulation networks were built by randomly selecting 15 and 30 super-peers (i.e. $\mathcal{N}_S$={15 or 30}) among the available 177 nodes in the PlanetLab data sets, and by randomly assigning the remaining nodes as associated normal-peers. This resulted in about 11 and 5 normal-peers per super-peer, respectively. The super-peers were then organized in power-law topologies using the BRITE [124] topology generator. The Rlists of the normal-peers and the LETs of the super-peers were initialized (with a fixed size of 10 for a Rlist). The information dissemination scheme was then performed on the simulation networks with different target reaches. In an ideal case, where the set of super-peers remains constant and where the proactive scheme has been operating for



(a) $\mathcal{N}_S$=15 and m=3 (PlanetLab 11/04/05)



(b) $\mathcal{N}_S$=15 and m=2 (PlanetLab 11/04/05)

Figure 7.2 Fraction of discovered QEAPs among all the discoverable ones, for different TTL_Q (i.e. Query TTL) and information dissemination reach D.

a sufficient duration, a given information (e.g. a LET entry from a super-peer) would reach all the super-peers. This case corresponds to an achieved target reach D of 100% (i.e. D=100%-reach). However, due to the dynamic behavior of peer-to-peer networks, such a 100%-reach state might not be achieved at a given time. Therefore, to evaluate the performance of the proposed proactive scheme in these non-ideal cases, these experiments specifically set the reach of the information to some given target values. Equations (6.5) and (6.9) from section 6.4.3 were used to determine a pair (P_D ; TTL_D) corresponding to a probabilistic information dissemination which converges to a requested target reach. As mentioned in section 6.4.3, a similar target reach can be achieved by using different values of P_D and TTL_D. Finally at a random time stamp, a random pair of node (N_X; N_Y) was selected, and a QEAP search process was performed using the reactive scheme with different query TTL (TTL_Q). This experiment was repeated 10^4 times to obtain the following averaged results.

Figure 7.2(a) and 7.2(b) present the result for $\mathcal{N}_S$=15 ($\mathcal{N}_S$=30 provides similar results). For a TTL_Q = {1, 2, or 3}, a SPAD architecture with the information dissemination scheme allows the discovery of more QEAPs than a SPAD architecture without this scheme. Moreover, for a given TTL_Q, a higher dissemination reach provides a higher number of discovered QEAPs. Assuming that super-peers initially set P_D and TTL_D to aim at D=100%-reach, starting from a state of 0%-reach, dissemination reach will increase as time progresses and super-peer network remains quasi-stable. For example with TTL_Q=1 and a minimum node degree m=3, a dissemination reach of D=50% (i.e. the LET information of each super-peer has reach 50% of all the super-peers) allows the discovery of about 95% of the discoverable QEAPs [1], while only about 82.5% are discovered without the dissemination scheme (labeled "No Diss." on the figures). This difference increases as the minimum node degree of the network decreases. On figure 7.2(b), for TTL_Q=1 and m=2, the difference between the two scenarios (i.e. D=50%-reach and No-Dissemination)

[1] A QEAP is discoverable (i.e. can be discovered) if there exists a SPAD super-peer that holds some information about it. Within a given SPAD community at a given time, it is possible that some existing QEAPs could not be discovered because there are no super-peers that store information about them.

Table 7.3 Search Cost in Messages for Different TTL_Q (PlanetLab 11/04/05)

Network Topology	Total number of messages generated by super-peers for each TTL_Q value				
	Values of TTL_Q (i.e. TTL of the QEAP Request)				
	1	2	3	4	5
$\mathcal{N}_S$=15 ; m=2	4.6	10.3	13.1	14.3	14.8
$\mathcal{N}_S$=15 ; m=3	6.1	12.1	14.0	14.7	14.9
$\mathcal{N}_S$=30 ; m=2	4.8	14.6	22.2	26.2	28.1
$\mathcal{N}_S$=30 ; m=3	6.6	18.9	25.1	27.7	29.9

Table 7.4 Search Latency for TTL_Q (PlanetLab 11/04/05)

	Values of TTL_Q				
	1	2	3	4	5
QEAP Search Latency (ms)	190.9	380.5	572.4	762.0	951.9

in terms of discovered QEAPs is about 17.5%, compared to the 12.5% for m=3. Indeed, the more connections a super-peer has, the more other super-peers a request will reach at each TTL_Q steps (i.e. the more information it will access), hence the decrease in the performance difference between D=50%-reach and "No-Diss." scenarios.

For one QEAP request processing, table 7.3 presents the average number of messages generated by super-peers for different values of TTL_Q. Table 7.4 shows the average QEAP search latency depending on the request TTL_Q. This latency takes into account the request forwarding delay, and the response delay on the reverse path. It does not account for processing time on super-peers. It is only a function of the number of hops on the corresponding request path, and does not depend on network topology. As expected, table 7.3 shows that when TTL_Q=5 (the network diameter), the message costs stabilize just under the numbers of super-peers ($\mathcal{N}_S$=15 and $\mathcal{N}_S$=30). Indeed, once a query has reached all existing super-peers, it cannot be forwarded anymore. The steep increase in message cost for initial TTL_Q values is due to the power-law nature of the topologies, i.e. requests reach a highly connected super-peer after one hop, which results in higher message generation at the next TTL step.

These tables and figures provide sufficient information to quantify the message cost and search latency gains of the proposed information dissemination scheme. For example, if we assume a SPAD community with $\mathcal{N}_S$=15 and m=3 and sufficient initial time to have a 50%-reach dissemination, the information dissemination scheme allows the discovery of around 95% of the discoverable QEAPs at an average cost of 6.1 messages and a latency of 190.9 ms. Achieving a similar performance without the proposed proactive scheme requires 12.1 messages, and a latency of 380.5ms.

7.5 EVALUATION OF SPAD QEAP SELECTION SCHEME

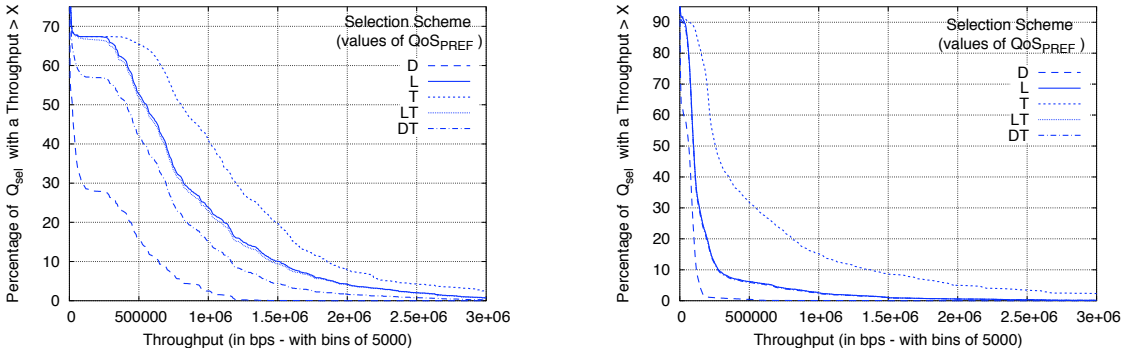
The following experiment was performed to evaluate the four QEAP selection schemes, as presented in section 6.4.5. First, the same *brute-force* search process as in section 5.4.1

was performed to discover all existing *Delay-QEAPs* and *Loss-QEAPs* within the NLANR-AMP and the PlanetLab data sets. Then for a given pair of nodes that has a set $\mathcal{Q}$ of existing QEAPs, each QEAP selection scheme was executed in turns on $\mathcal{Q}$, and the selected QEAP Q_{sel} was recorded in each cases. When one of the schemes selected a different Q_{sel} than the other schemes, the characteristics of these different selected QEAPs were compared in regards to their delay gain, their gain in packet-loss rate, and their achievable throughput (using equation (6.19)). The same processing and comparison were performed on each pair of nodes with existing QEAPs. Figures 7.3, 7.4, and 1 present the summary of these comparisons for the NLANR-AMP and the PlanetLab data sets. For simplicity, this experiment used QEAP requests with D_{REQ}=D_{REF} and L_{REQ}=L_{REF}. Using a $D_{REQ} < D_{REF}$ and/or a $L_{REQ} < L_{REF}$ will not change the qualitative characteristic of the results. The graphs related to the NLANR-AMP and the PlanetLab data sets follow similar trends, thus the remainder of this section only discusses the graphs based on the NLANR-AMP data set.

Figure 7.3(a) shows that when QoS_{PREF}=T, the corresponding selection scheme succeeds in providing the QEAPs with the maximum achievable throughput. In a similar manner, figures 7.4(a) and 7.5(a) also demonstrate that the QoS_{PREF}=L and the QoS_{PREF}=D selection schemes succeed in providing the QEAPs with the maximum gain in packet-loss rate and delay respectively. The y-axis on figures 7.4(a) and 7.5(a) are truncated to focus on the most significant part of the graphs. All the graphs in these figures converge directly to a gain of 100% below the y-axis truncated values (i.e. 20 and 60 respectively).

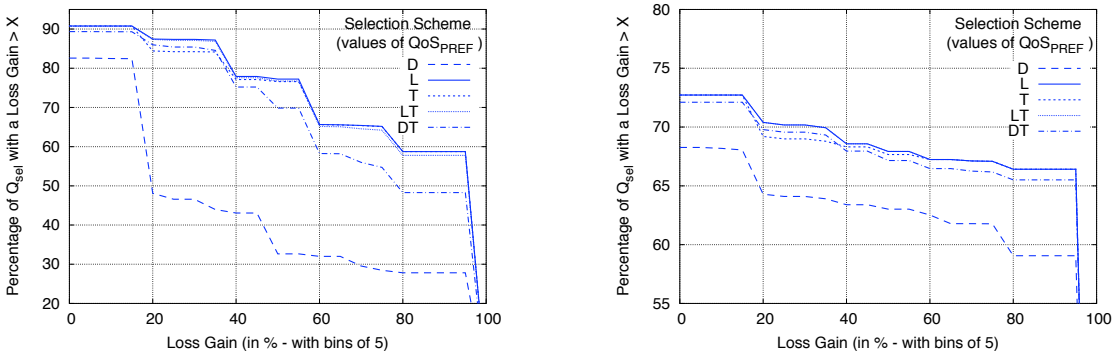
On figure 7.4(a), the cumulative distribution does not reach 100% for positive values of the gain in packet-loss rate. Thus, the QoS_{PREF}=L selection scheme did select QEAPs with negative gain in packet-loss rate, i.e. QEAPs with worse packet-loss rate than the default path! As explained previously, the QEAP characteristic comparison was performed each time that any of the selection scheme returns a different Q_{sel} than the others. In some of these cases, the set $\mathcal{Q}$ did not contain any *Loss-QEAP*, but only *Delay-QEAPs* with worse packet-loss rate (i.e. within the right-bottom quadrant of figure 5.4). Therefore, the execution of the QoS_{PREF}=L selection scheme on $\mathcal{Q}$ returned the *Delay-QEAPs* with the less negative gain in packet-loss rate. This explains the maximum value of 90% for the cumulative distribution at 0% of gain in packet-loss rate within these experiments. In a real-world deployment of a SPAD prototype, this selection scheme will logically not select any QEAP that provide a negative gain. Similar considerations also apply to figure 7.5(a) and the QoS_{PREF}=D selection scheme.

From figures 7.3(a) and 7.5(a), one can conclude that the QoS_{PREF}=DT selection scheme managed to provide QEAPs with higher gain in delay than the QoS_{PREF}={T, L, LT} schemes, and higher achievable throughput than the QoS_{PREF}={D, L, LT} schemes. Thus, it managed to maximize both the delay gain and the achievable throughput parameters on selected QEAPs. However, figures 7.3(a) and 7.4(a) show that the QoS_{PREF}=LT selection scheme failed to maximize both the gain in packet-loss rate and the achievable throughput on selected QEAPs. Indeed on figures 7.3(a), the LT curve is lower than the L curve, instead of being higher. The following two arguments can explain these results. First, Moon [115] demonstrated that on a given Internet path, an increase in packet-loss rate implies an increase in delay in the same timeframe (i.e. correlation between delay and packet-loss



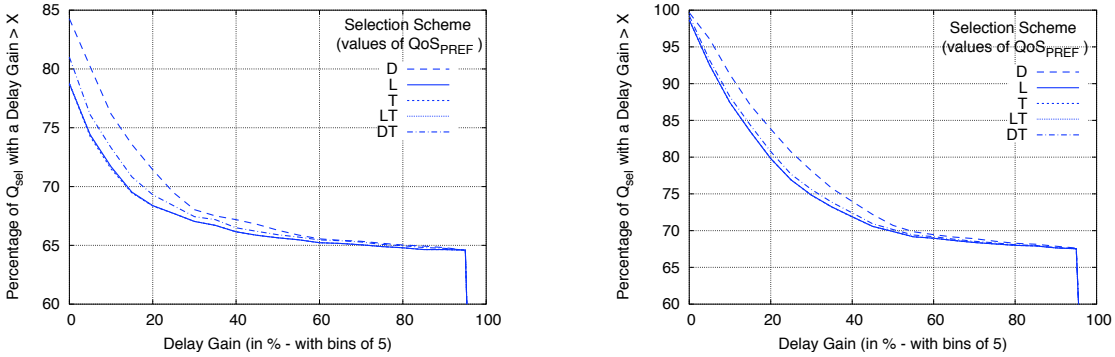
(a) NLANR-AMP 17/06/04 (b) PlanetLab 11/04/05

Figure 7.3 Comparison of the throughput on the selected QEAPs (Q_{sel}) from different Selection Scheme (Cumulative Distribution).



(a) NLANR-AMP 17/06/04 (b) PlanetLab 11/04/05

Figure 7.4 Comparison of the gain in packet-loss rate on the selected QEAPs (Q_{sel}) from different Selection Scheme (Cumulative Distribution).



(a) NLANR-AMP 17/06/04 (b) PlanetLab 11/04/05

Figure 7.5 Comparison of the gain in delay on the selected QEAPs (Q_{sel}) from different Selection Scheme.

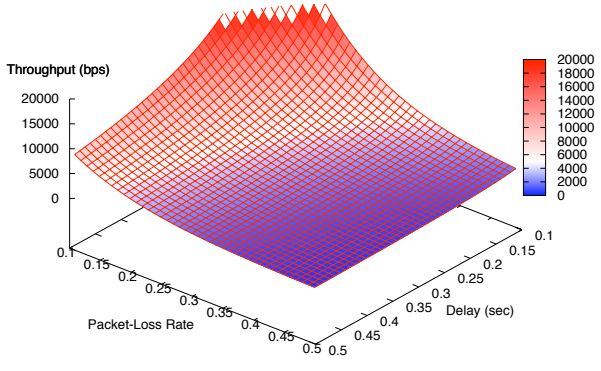


Figure 7.6 Variations of the computed TCP-friendly throughput (from equation (6.19)) for different values of delay and packet-loss rate.

rate on a given Internet path). Therefore if Q_y is the alternate path that provides the lowest packet-loss rate in the set $\mathcal{Q}$, then there is a small probability that another path Q_z (with $L_{Qz} > L_{Qy}$) would have a sufficiently low delay to compensate in equation (6.19) and allow $T_{Qz} > T_{Qy}$. Second, as illustrated on figure 7.6, the computed TCP-friendly throughput (using equation (6.19)) is more sensitive to packet-loss rate variations than delay variations within the ranges [0.1 ; 0.5] and [0.1s ; 0.5s], respectively[2]. Therefore, if such a path Q_z exists in the set $\mathcal{Q}$, since equation (6.19) is more sensitive to packet-loss variations, there is also a small probability that the resulting T_{Qz} would be superior to T_{Qy}. These arguments explains the inefficiency of the QoS_{PREF}=LT selection scheme to maximize both packet-loss gain and achievable throughput, using a linear-based utility function.

7.6 DISCUSSION

The above performance evaluations were all based on experiments using delay measurements between real Internet hosts. These measurements provide a realistic "snapshot" of Internet delay characteristics, thus ensuring that the simulation environments accurately mimic the characteristics of a real world environment. However, these simulations do not account for dynamic changes in network connectivity. Indeed for each trial, the simulated environment is not modified to reflect the changes due to the resources being used by any previously discovered QEAPs. To address this issue, and perform evaluations in a dynamic environment, a possible approach is to deploy a SPAD prototype on a large scale virtual test-bed, such as the one provided by the PlanetLab consortium [131]. A simple SPAD prototype was designed and developed as part of this thesis research work. It has been successfully tested on a simple test-bed with 4 hosts that each runs a super-peer and several normal-peers. These hosts are connected via an emulated network provided by another machine running the DummyNet [132] tool. Appendix B provides a detailed description of

[2]These ranges include the values from the studied data sets (see the previous tables and figures).

this prototype and its use in a demonstration setting. Further tests and development are required prior to deploying this prototype on a resource-shared test-bed like PlanetLab.

Another limit of the above experimental approach is the relatively small numbers of available nodes in the considered data sets [10, 9, 8], as presented in table 5.1. These numbers do not allow extensive scalability evaluations. A possible approach to overcome this limitation would be to derive a statistical model for Internet end-to-end delay and packet-loss rate distributions (possibly based on the measurements from these data sets), and combine this model with a topology generator to create large simulation networks. Some contributions study such statistical model [133, 134].

Finally, one could also argue that the nodes in the studied data sets are not representative of the average real Internet host. Indeed, these data set nodes are either on corporate or academic networks, and consequently have more bandwidth and less latency than hosts using dial-up or asymmetrical connections. However as discussed in chapter 1, SPAD is designed for a community of third party service providers competing to propose software elements for *overlay applications*, and cooperating to deliver QoS within these *overlay applications*. As these service providers are business-oriented entities, it is realistic to assume that they have network connectivity at least similar to the hosts from the studied data sets.

7.7 CHAPTER SUMMARY

This chapter presented the performance evaluation of the QEAP discovery and selection schemes, as described in the previous chapter 6. These evaluations are based on experiments with the same measurement data sets as in section 5.3. The results of these experiments are summarized as follows:

- the simple QEAP discovery scheme from section 6.2 allows the discovery of 77.6% to 88.0% of the existing *Delay-QEAPs* within the RIPE-TTM data sets, depending on the value of some initialization parameters,
- the SPAD reactive information exchange scheme from section 6.4.2 outperforms the previous simple QEAP discovery scheme, discovering more existing *Delay-QEAPs* within the NLANR-AMP data sets (e.g. 77.58% compared to 68.95%),
- the SPAD proactive information exchange scheme from section 6.4.3 complements the reactive scheme and significantly reduces the message costs and the latency of a QEAP search process (e.g. a cost of 6.2 messages for a latency of 190.9ms to discover 95% of discoverable QEAPs within the PlanetLab data sets, compared to 12.1 messages and 380.5ms without the proactive scheme),
- the SPAD QEAP selection schemes from section 6.4.5 allows the selection of the optimal QEAP among a set of potential ones, according to the user's QoS preference (e.g. selecting the QEAP with the maximum throughput and the minimum latency).

These various results collectively showed that the proposed SPAD system succeeds in discovering and selecting existing QEAPs within communities of distributed cooperative

peers on the internet. They validate the approach and solution that this thesis proposes to address the problem of providing enhanced QoS to peers within an *overlay association*.

Chapter 8

Conclusion

This chapter concludes this dissertation. It presents a summary of the contributions that were made within this thesis, and proposes some directions on future research work that would complement these contributions.

8.1 PROBLEM SUMMARY

The first three chapters (i.e. 2, 3, 4) of this dissertation presented and consecutively refined the problem that this thesis aimed at addressing, and the approach that it proposed to achieve this goal.

Chapter 2 introduced the emerging challenge of providing services to users of mobile network-enabled devices. The number of such mobile devices has recently increased at a fast pace, and the users have been requesting more and more complex services/functionalities from them[1]. However, these mobile devices have physical and economical constraints that limit their embedded capabilities. To address this challenge, chapter 2 proposed an approach where the network (instead of the devices) provided the required complex functionalities to the users, in the form of composite services built from distributed cooperating application elements.

Such an approach relied on two computing and networking concepts, namely *service composition* and *overlay networks*. Chapter 3 introduced these concepts, presented some pertaining challenges, and reviewed some related research contributions. These reviews indicated that: i) the *overlay network* paradigm offered an adequate substrate to deploy composite services, and ii) the issue of providing Quality of Service to the users of composite services has not been thoughtfully addressed by previous contributions. This chapter also intro-

[1]For example: receiving on a hand-held device a video-conference stream with on-the-fly video adaptation and subtitle generation in another language, while having the same stream stored in a remote office computer for future reference.

duced the term *overlay application* to refer to a composite service, i.e. an application that was composed of distributed cooperating software elements.

Chapter 4 provided an overview of the notion of QoS from different perspectives within the networking stack model. This chapter noted that the QoS perceived by the users of an overlay application significantly depended on the QoS experienced on the end-to-end paths that connected the elements composing this *overlay application*. Thus, chapter 4 continued with the review of some classic proposals aiming at providing end-to-end QoS on Internet paths. Then it discussed the alternate approach taken by this thesis to achieve the same goal: using virtual composite paths with enhanced QoS characteristics compared to the default Internet paths. Finally, chapter 4 highlighted the open issue of discovering, selecting, and constructing these virtual paths in a distributed manner within a community of cooperative peers.

Thus this thesis aimed at providing enhanced QoS between peers involved in an overlay application. To achieve this goal, this thesis proposed to use QoS enhanced Alternate Paths (QEAPs), and provided a complete system to discover, select and utilize these QEAPs in a distributed manner.

8.2 CONTRIBUTION SUMMARY

The last three chapters (i.e. 5, 6, 7) of this dissertation presented and discussed the contributions made within this thesis.

Chapter 5 presented the first contribution, namely an extensive description and characteristic analysis of the fore-mentioned QEAPs. This analysis complemented and extended the work presented in previous studies [4, 5, 6]. It indicated that within the Internet there exists many virtual composite paths that provide significant benefits (such as lower packet delays and packet loss rates) to the data packets that use them, compared to the default Internet paths. Moreover, this analysis also indicated that the gains on these QEAPs are on average sufficiently constant to benefit the majority of the current Internet data streams. These analysis results motivated the use of QEAPs to provide enhanced QoS between peers involved in an *overlay application*.

Chapter 6 presented the second and main contribution of this thesis, namely SPAD, a Super-Peer based Alternate path Discovery system. SPAD is an original architecture which allows the discovery and selection of QEAPs in a distributed manner. It is based on a community of cooperative peers grouped as an unstructured Super-Peer network. These peers use the SPAD distributed mechanisms to discover, select, construct and utilize QEAPs. SPAD is a fully distributed system with no single point of failure. It empowers the end-users at the edge of the network, allowing them to directly discover and select QEAPs, without having to rely on any central entity or third-parties within the core of the network (i.e. at the ISP or AS level). Moreover, SPAD is designed to be part of a middleware framework, which resides on the end-user machines. Therefore, it can be easily and incrementally deployed, as it does not require any change to the current Internet routing mechanisms or core components, such as AS routers. Through its novel approach to the discovery and the

selection of QEAPs, SPAD enables the provision of QoS on the connections between the elementary application components that form an *overlay application*.

An extensive performance evaluation of SPAD was provided in chapter 7. This evaluation was based on simulation experiments, using measured data from three different community of peers on the Internet. These experiments indicated that: i) the SPAD QEAP discovery schemes succeeded at discovering a significant amount of existing QEAPs within each considered community, and ii) the SPAD QEAP selection scheme succeeded at deciding which QEAP *best* met the user's QoS preferences within a set of discovered ones.

Thus, this thesis demonstrated that it was feasible to provide QoS to an *overlay application* by using alternate Internet paths resulting from the compositions of independent consecutive paths. Furthermore, it also demonstrated that it was possible to discover and compose these independent paths in a distributed manner within an community comprising a limited large number of autonomous cooperating peers.

8.3 FUTURE RESEARCH DIRECTIONS AND PERSPECTIVES

The proposed SPAD system has been designed and evaluated for a bounded community of peers, such as the community of service providers as introduced in chapter 2. Compared to the entire Internet, this community has a smaller number of hosts. The generalized use of QEAPs and the deployment of distributed QEAP discovery/selection schemes on all hosts of a vast network like the Internet is not within the focus of this thesis and requires further studies. Such studies possibly involve

- an extensive evaluation of the SPAD schemes in a dynamic environment (as suggested and discussed in section 7.6),
- the design of an advanced admission control scheme to be deployed on potential relay nodes (as suggested and discussed in section 6.5),
- the design of an adequate congestion control scheme to ensure that SPAD traffic coexists in a *friendly* manner with other type of traffic, e.g. TCP traffic. Such a congestion control scheme can operate independently on each hop of an *overlay association*, or as a global scheme on an entire *overlay association*. Moreover, the same choice also occurs within an given QEAP (i.e. a hop of an *overlay association*), where congestion control can also be deployed independently on each path components, or globally on the entire set of paths composing the considered QEAP. The selection between these types of possible schemes requires further investigations.

Lua et al. [107] propose a taxonomy of peer-to-peer systems with two main categories, namely systems based on a *structured* or an *unstructured* model. As stated in section 4.4.3, the proposed SPAD system is based on the *unstructured* model. This design decision is arbitrary and further studies are required to decide which model is more adequate to QEAP discovery and selection. In the early stage of the research work related to this thesis, a very basic QEAP discovery scheme based on the *structured* model was proposed [108]. The scheme presented in section 6.2 inspired this basic *structured* scheme, which is also based

on a community of cooperating peers. It proposes to embed these peers within a virtual space according to their relative delays, and to use a modified distance function on a given peer N_A to compute/discover potential QEAPs $[N_A \rightarrow N_X \rightarrow N_B]$. The performance of this scheme is significantly limited by the non-metric nature of the latency QoS parameter[2]. Some existing contributions provide methods to embed non-metric pairwise proximity data into virtual metric spaces, without losing any existing clustering properties among the considered data sets. The "constant shift embedding" technique from Roth et al. [135] is an example of such a method. Investigating the feasibility/performance of a *structured* QEAP discovery scheme, which possibly uses such an embedding method, constitutes a challenging potential research direction.

This thesis proposes the use of single virtual Internet paths to enhance the Quality of Service between pairs of Internet hosts. The study of other type of applications for these virtual Internet paths is another interesting research avenue. For example, one can use multiple virtual Internet paths coupled with network coding and/or Forward Error Correction (FEC) methods to maximize the utilization of the overlay links between nodes involved in simultaneous *overlay applications*. In another example, one can use multiple virtual paths in conjunction with multicast schemes on overlay networks to enhance the QoS perceived by the receiving peers. In these examples, the use of multiple virtual paths also raises a fairness issue. When a user A sends its data traffic over multiple virtual paths, he/she utilizes more network resources than a user B who only require one path. This result in a potential fairness issue, when for example a congestion event occurs on a given network point (i.e. router) shared by these users, the competition between the traffic from user A and B might potentially be biased towards user A. Ensuring the fairness of these multiple path schemes with the currently existing communication schemes is another interesting challenge.

Furthermore, this thesis also proposes the use of virtual Internet paths that enhance only two particular QoS parameters, namely latency and packet-loss rate. However, there exists many applications that require QEAPs, which enhance other types of QoS parameters (such as jitter, bandwidth), or a complex constrained combination of multiple QoS parameters. Several works [82, 83, 84] studied this multi-constraint path problem in the context of QoS routing (as presented and discussed in section 4.3.1), but few contributions studied such a problem in the context of overlay networks and overlay paths. As introduced in section 3.3, the overlay network model virtualizes an existing network infrastructure for the above users. Such a feature can potentially lead to the design of new multi-constraint path selection schemes, or allow the deployment of modified existing QoS routing schemes. The design, modification or deployment of such multi-constraint routing schemes in the context of overlay network constitutes another possible research direction.

Finally, one can also study the feasibility of using the SPAD system as the base of a business-oriented service, similarly to QRON [60] and OverQoS [48]. For example, an association of business partners might form a SPAD community, and act as a gateway to provide or sell QEAPs to other Internet users. This business-oriented context raises other potential research problems such as the type of Service Level Agreement (SLA) to use

[2] As demonstrated in chapter 5, Internet delays do not necessarily satisfy the triangle inequality, hence the existence of QEAPs.

between the "customers" and the SPAD peers, the design of efficient distributed trust and accountability mechanisms between these peers, the detection and management of eventual misbehaving peers, or the type of peering agreements and market relationships between the service providers hosting the SPAD peers.

8.4 FINAL NOTES

Sharing the same view as recent contributions [2, 3], this thesis foresees the emergence of pervasive computing and networking environments, where the network (i.e. the next generation Internet) will not only be a communication medium, but also an endless source of *composable services* available to the end-users. Through real-world measurement studies, analytical models, and experimental simulations, this thesis proposed a coherent architecture to enhance the Quality of Service in particular instances of such a pervasive computing and networking environment. The core approach taken by this thesis, i.e. virtualizing the underlying network communication paths from the user/application point of view, is consistent with other recent approaches and reflections on the future of the Internet [136, 137].

Appendix

A. Computation of $M+1$ in a Stationary Power-Law Topology

Section 6.4.3 introduced $M+1$, the average number of direct neighbors for a given node in a graph, and provided a mechanism that allows a node to evaluate $M+1$ in a distributed manner within a dynamic topology. To complete the analysis of the subsection "*integration of a probabilistic flooding in SPAD*" of section 6.4.3, this appendix describes an analytical computation of $M+1$ in a power-law topology that has reached a stationary regime.

In a stationary topology, Barabási and Albert [138] provide the following expression for the probability $P(k)$ of a node to have k direct neighbors:

$$P(k) = \frac{2m^2}{k^3} \tag{1}$$

with m being the minimum node degree of a node. Thus, by definition the expected value $E(k)$ of the average number of direct neighbors for a node is given by:

$$E(k) = \sum_{k=m}^{\infty} kP(k) = \sum_{k=m}^{\infty} k2m^2k^{-3} = 2m^2 \sum_{k=m}^{\infty} \frac{1}{k^2} \tag{2}$$

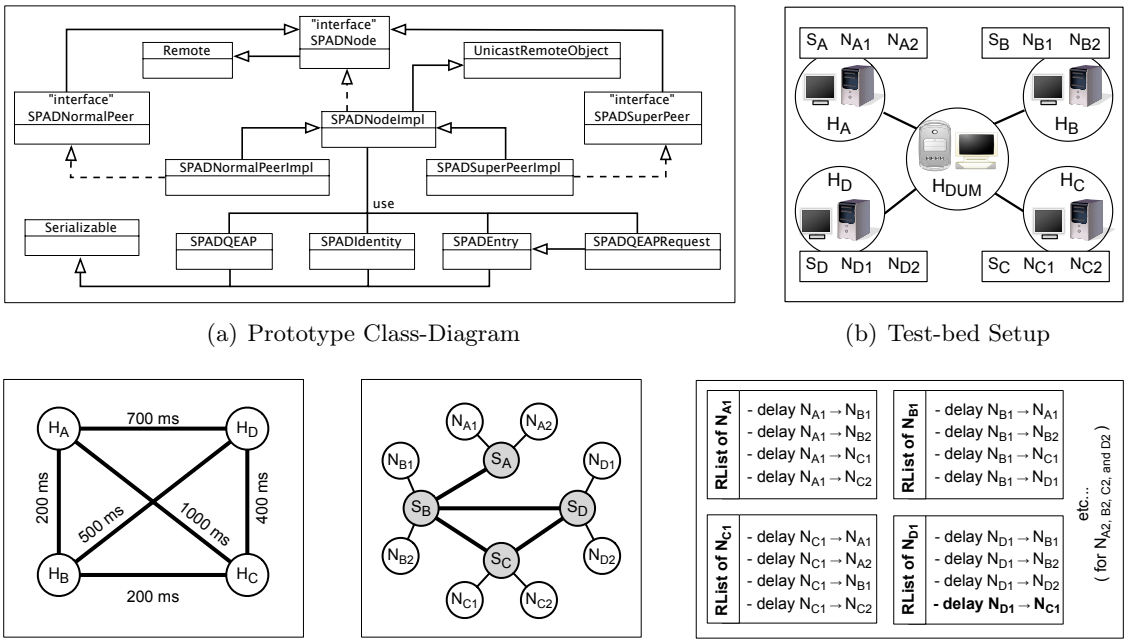
Since $S = \sum_{k=1}^{\infty} \frac{1}{k^2}$ is the Riemann series, which converges to $\frac{\pi^2}{6}$, equation (2) provides:

$$E(k) = 2m^2(\frac{\pi^2}{6} - (\sum_{k=1}^{m-1} \frac{1}{k^2})) \tag{3}$$

In a stationary power-law topology, the above result (3) provides an analytical expression of $M+1$, the average number of direct neighbors for a given node.

B. A Simple SPAD Prototype

This section describes a simple SPAD prototype, which was designed and developed as part of this thesis research work. This prototype was implemented for demonstration purpose, i.e. it demonstrate the feasibility of the schemes as described in chapter 6. As such, it has not been optimized yet for deployment on real Internet hosts, and cannot be used to perform dynamic quantitative evaluation of the SPAD schemes.



(a) Prototype Class-Diagram (b) Test-bed Setup

(c) Emulated Delays (d) SPAD Topology (e) RList Content of Normal Peers

Figure 1 Descriptions of the simple SPAD prototype and the demonstration test-bed built with it.

The SPAD prototype was developed in Java 1.4, using the Remote Method Invocation framework. This framework allows the methods of a Java object on a host N_A to be invoked by another remote Java object possibly on a distant host N_B. Figure 1(a) presents a simple UML (Unified Modeling Language) class-diagram of the simple SPAD prototype.

This prototype is used to illustrate the operation of the SPAD systems within a demonstration test-bed. Figure 1(b) presents this test-bed, where each host $H_{X\in\{A,B,C,D\}}$ executes a unique SPAD super-peer (S_X) and two SPAD normal-peers (N_{X1}, N_{X2}). These hosts are inter-connected via host H_{DUM}, which has 4 ethernet interfaces and executes the DummyNet software [132]. DummyNet simulates/enforces queue and bandwidth limitations, delays, and packet losses between ethernet interfaces. This tool emulates a basic wide-area network between the 4 hosts connected to H_{DUM}.

For the demonstration purpose, the following procedures are performed:

- the DummyNet tool on host H_{DUM} is configured to set the delays on the default end-to-end paths between hosts $H_{X\in\{A,B,C,D\}}$, according to figure 1(c).
- a super-peer (S_X) and 2 normal-peers (N_{X1}, N_{X2}) are executed on each hosts H_X
- the super-peers are connected as illustrated on figure 1(d), and the normal-peers have their RLists configured as illustrated on figure 1(e)
- the super-peers and normal-peers are initialized as described in chapter 6
- the SPAD proactive scheme (section 6.4.3) is activated on the super-peers

Then in a first experiment, a basic VoIP Java application is launched between node N_{A1} and N_{C1} without using the SPAD system. Thus, the voice data packets *uses* the default path between H_A and H_C, and the users experience a delay of 1000ms on both ends of the call, which is not compatible with interactive conversations.

In a second experiment, prior to launch the same application between the same nodes, the SPAD system is used to discover any existing *Delay-QEAP*[3]. In this example, the path $[N_{A1} \rightarrow N_{D1} \rightarrow N_{C1}]$ is indeed a $Delay - QEAP$ and is used to convey the voice data packets. The users experience a delay of 400ms, which is within the limits for interactive conversations [74].

[3] N_{A1} sends a QEAP-request to S_A, which processes it as described in chapter 6. As a result, N_{A1} receives some connectivity information (such as the entry "delay $N_{D1} \rightarrow N_{C1}$", originally from N_{D1}'s RList) that allows it to discover the QEAP $[N_{A1} \rightarrow N_{D1} \rightarrow N_{C1}]$.

List of Publications

[1] B. Thai, R. Wan, A. Seneviratne, and T. Rakotoarivelo.. Integrated Personal Mobility Architecture: a complete personal mobility solution. *In Mobile Networks and Applications (MONET), Vol 8. No.1, February, 2003.*

[2] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. Enhancing QoS Through Alternate Path: An End-to-End Framework. *IEEE Intl. Conference on Networking (ICN), Réunion Island, France, April, 2005.*

[3] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. A Structured Peer-to-Peer Method to Discover QoS Enhanced Alternate Paths. *IEEE Intl. Conference on Information Technology and Applications (ICITA), Sydney, Australia, July, 2005.*

[4] S. Ardon, M. Portmann, T. Rakotarivelo, P. Sénac, S. Zhou, M. Hogan and A. Seneviratne. OPENDIR : An Open Distributed Service Directory. *IEEE Intl. Conference on Industrial Informatics (INDIN), Perth, Australia, August, 2005.*

[5] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. A Super-Peer based Method to Discover QoS Enhanced Alternate Path. *IEEE Asia-Pacific Conference on Communications (APCC), Perth, Australia, October, 2005.*

[6] T. Rakotoarivelo, P. Sénac. Discovering Alternate Internet Paths to Enhance End-to-End QoS. *Poster Session, Student Workshop of ACM Conference on Emerging Network Experiment and Technology (CoNEXT), Toulouse, France, October, 2005.*

[7] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. A Proactive Scheme for QoS Enhanced Alternate Path Discovery in a Super-Peer Architecture. *IEEE GLOBECOM Conference, San Francisco, United States of America, November, 2006.*

[8] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. A Peer-to-Peer Scheme to Select QoS Enhanced Alternate Path. *IEEE Intl. Conference on Communication System Software and Middleware (COMSWARE), Bangalore, India, January, 2007.*

[9] T. Rakotoarivelo, P. Sénac, A. Seneviratne, M. Diaz. SPAD: a Distributed Middleware Architecture for QoS Enhanced Alternate Path Discovery. *Submitted for review at the Elsevier Computer Networks journal , July, 2006.*

Bibliography

[1] "Computer history museum," *http://www.computerhistory.org/*.

[2] M. Weiser, "The computer for the twenty-first century," *Scientific American*, vol. 265, pp. 94–104, Sep. 1991.

[3] B. Raman, S. Agarwal, Y. Chen, M. Caesar, W. Cui, P. Johansson, K. Lai, T. Lavian, S. Machiraju, Z. M. Mao, G. Porter, T. Roscoe, M. Seshadri, J. Shih, K. Sklower, L. Subramanian, T. Suzuki, S. Zhuang, A. Joseph, R. Katz, and I. Stoica, "The SAHARA model for service composition across multiple providers," in *IEEE Conference on Pervasive Computing*, Aug. 2002.

[4] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of internet path selection," in *Proceedings of ACM SIGCOMM Conference*, 1999.

[5] R. Beyah, R. Sivakumar, and J. Copeland, "Application layer switching: A deployable technique for providing quality of service," in *Proceedings of IEEE GLOBECOM Conference*, 2003.

[6] S. Tao, K. Xu, Y. Xu, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z. Zhang, "Exploring the performance benefits of end-to-end path switching," in *IEEE International Conference on Network Protocols (ICNP)*, 2004.

[7] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *IEEE Conference on Data Engineering*, 2003.

[8] "Planetlab all-pair ping project," *http://pdos.csail.mit.edu/s̃trib/pl_app/*.

[9] "Nlanr active measurement project," *http://watt.nlanr.net*.

[10] "Ripe traffic test measurement," *http://www.ripe.net/test-traffic/*.

[11] S. Rosen, "Electronic computers: A historical survey," *ACM Computing Surveys*, vol. 1, no. 1, pp. 7–36, 1969.

[12] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff, "The past and future history of the internet," *Communications of the ACM*, vol. 40, no. 2, pp. 102–108, 1997.

[13] S. Crocker, S. Carr, and V. Cerf, "New host-host protocol," *IETF RFC 33*, Feb. 1973.

[14] University of Southern California (USC) - Information Sciences Institute (ISI), "Internet protocol (IP)," *IETF RFC 791*, Sep. 1981.

[15] S. Deering and R. Hinden, "Internet protocol version 6 (IPv6)," *IETF RFC 2460*, Dec. 1998.

[16] E. C. Perkins, "IP mobility support for IPv4," *IETF RFC 3344*, Aug. 2002.

[17] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "Cans: Composable, adaptive network services infrastructure," in *Proceedings of the UNSENIX Symposium on Internet Technologies and Systems*, 2001.

[18] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0," *http://www.w3.org/TR/REC-xml/*, Aug. 2006.

[19] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session description protocol," *IETF RFC 4566*, Jul. 2006.

[20] D. Fensel, O. Lassila, F. VanHarmelen, I. Horrocks, J. Hendler, and D. McGuinness, "The semantic web and its languages," *IEEE Intelligent Systems*, vol. 15, no. 6, 2000.

[21] F. Manola and E. Miller, "Resource description framework (RDF)," *http://www.w3.org/TR/rdf-primer/*, Feb. 2004.

[22] D. Brickley and R. Guha, "RDF vocabulary description language 1.0: RDF schema," *http://www.w3.org/TR/rdf-schema/*, Feb. 2004.

[23] "The DARPA agent markup language (DAML)," *http://www.daml.org/*.

[24] "The ontology inference layer (OIL)," *http://www.ontoknowledge.org/oil/*.

[25] D. McGuinness and F. VanHarmelen, "OWL web ontology language overview," *http://www.w3.org/TR/owl-features/*, Feb. 2004.

[26] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web service description for the semantic web," in *The First International Semantic Web Conference (ISWC)*, Sardinia, Italy, June 2002.

[27] N. Mitra, "SOAP version 1.2 part 0: Primer," *http://www.w3.org/TR/2003/REC-soap12-part0-20030624/*, Jun. 2003.

[28] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (WSDL)," *http://www.w3.org/TR/wsdl*, Mar. 2001.

[29] E. Guttman, C. Perkins, J.Veizade, and M. Day, "Service location protocol v2," *IETF RFC 2608*, Jun. 1999.

[30] J. Rosenberg, H. Schulzrinne, and B. Suter, "Wide area network service location," *IETF Internet Draft*, 1997.

[31] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz, "An architecture for a secure service discovery service," in *ACM/IEEE Conference on Mobile Computing and Networking (MobiCom)*, 1999.

[32] L. Clement, A. Hately, C. VonRiegen, and T. Rogers, "UDDI version 3.0.2," *http://uddi.org/pubs/uddi_v3.htm*, 2004.

[33] "UDDI products and components," in *http://www.uddi.org/solutions.html*.

[34] S. Ardon, M. Portmann, T. Rakotarivelo, P. Senac, S. Zhou, M. Hogan, and A. Seneviratne, "OPENDIR : An open distributed service directory," in *IEEE Intl. Conference on Industrial Informatics (INDIN)*, Aug. 2005.

[35] X. Gu, K. Nahrstedt, and B. Yu, "SpiderNet: An integrated peer-to-peer service composition framework," in *IEEE International Symposium on High-Performance Distributed Computing*, Jun. 2004.

[36] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "One ring to rule them all: Service discovery and binding in structured peer-to-peer overlay networks," in *ACM SIGOPS European Workshop*, 2002.

[37] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "CHORD: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM Conference*, 2001, pp. 149–160.

[38] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *ACM SIGCOMM Conference*, 2001.

[39] A. Rowstron and P. Druschel, "PASTRY: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.

[40] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, "Complex queries in DHT-based peer-to-peer networks," in *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[41] S. Kamvar, B. Yang, and H. Garcia-Molina, "Addressing the non-cooperation problem in competitive p2p systems," in *Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.

[42] H. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J. Shih, L. Subramanian, B. Zhao, A. Joseph, and R. Katz, "ICEBERG: An Internet-core network architecture for integrated communications," *IEEE Personal Communications*, vol. 7, no. 4, Aug. 2000.

[43] S. Gribble, M. Welsh, R. von Behren, E. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J.Hill, A. Joseph, R. Katz, Z. Mao, S. Ross, and B. Zhao, "The Ninja architecture for robust internet-scale systems and services," *Computer Networks*, vol. 35, no. 4, Mar. 2001.

[44] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic composition of web services using semantic descriptions," in *Workshop on Web Services: Modeling, Architecture and Infrastructure (ICEIS2003)*, Apr. 2003.

[45] M. Matskin and J. Rao, "Value-added web services composition using automatic program synthesis," in *Workshop Web Services, e-Business, and the Semantic Web*, Toronto, Canada, May 2002.

[46] C. Chuah, L. Subramanian, A. Joseph, and R. Katz, "QoS provisioning using a clearing house architecture," in *8th International Workshop on Quality of Service*, 2000.

[47] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *ACM SIGCOMM*, Aug. 2001.

[48] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An overlay based architecture for enhancing internet qos," in *First Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.

[49] "The gnutella protocol," *http://rfc-gnutella.sourceforge.net/*.

[50] "AKAMAI - content delivery network," *http://www.akamai.com*.

[51] I. Clarke, T. Hong, S. Miller, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *IEEE Internet Computing*, vol. 6, no. 1, 2002.

[52] H.Rahul, M. Kasbekar, R. Sitaraman, and A. Berger., "Towards realizing the performance and availability of a global overlay network," in *Passive and Active Measurement Conference (PAM)*, 2006.

[53] "BitTorrent protocol," *http://www.bittorrent.org/protocol.html*.

[54] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," in *International Workshop on Peer-to-Peer Systems*, March 2002, pp. pp. 85–93.

[55] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralised application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications (JSAC)*, 2002.

[56] S. Deering, "Host extensions for ip multicasting," *IETF RFC 1112*, Aug. 1989.

[57] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, 2000.

[58] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Symposium on Operating System Design and Implementation (OSDI)*, Oct.2000.

[59] "PeerCast," *http://www.peercast.org/*.

[60] Z. Li and P. Mohapatra, "QRON: QoS-aware routing in overlay networks," *IEEE Journal on Selected Areas in Communications*, January 2004.

[61] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in *Proceedings of the First ACM Workshop on Hot Topics in Networking (HotNets), October 2002*, 2002.

[62] "Planetlab research projects," *http://www.planet-lab.org/php/slices.php*.

[63] Workshop Attendees, "Report of the national science foundation workshop on fundamental research in networking," Apr. 2003.

[64] J. Touch, "Dynamic Internet overlay deployment and management using the X-Bone," *Computer Networks*, Jul. 2001.

[65] Z. Li and P. Mohapatra, "The impact of topology on overlay routing service," in *IEEE INFOCOM*, 2004.

[66] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *IEEE INFOCOM*, 2006.

[67] International Telecommunicatino Union (ITU), "Basic reference model for open distributed processing - part 2 - descriptive model," *ITU-T Recommendation X.902 - ISO/IEC 10746-2*, 1995.

[68] A. Vogel, B. Kerherve, G. v. Bochmann, and J. Gecsei, "Distributed multimedia applications and quality of service: a survey," in *Conf. of the Centre for Advanced Studies on Collaborative research (CASCON)*, 1994.

[69] P. Ferguson and G. Huston, "Quality of service in the internet: Fact, fiction, or compromise?" in *The Internet Society Conference (INET)*, 1998.

[70] J. Gozdecki, A. Jajszezyk, and R. Stankiewiez, "Quality of service terminology in ip networks," in *IEEE Communications Magazine*, vol. 41, 2003.

[71] S. Ardon, "Integration de l'utilisateur dans la gestion de la qualite de service pour des environnements heterognes," *PhD Dissertation, Universite Pierre et Marie Curie*, 2002.

[72] G. Ghinea and J. Thomas, "Qos impact on user perception and understanding of multimedia video clips," in *ACM Multimedia*, 1998.

[73] International Telecommunicatino Union (ITU), "Framework recommendation for multimedia services," *ITU-T Recommendation F.700*, 1996.

[74] ——, "End-user multimedia qos categories," *ITU-T Recommendation G.1010*, 2001.

[75] H. Schulzrinne and S. Casner, "RTP profile for audio and video conferences with minimal control," *IETF RFC 3551*, Jul. 2003.

[76] J. Wroclawski, "The use of RSVP with IETF integrated services," *IETF RFC 2210*, 1997.

[77] J. Huard and A. Lazar, "On QoS mapping in multimedia networks," in *IEEE International Computer Software and Application Conference*, Aug. 1997.

[78] J. Kurose and K. Ross, *Computer Networking, A Top Down Approach.* Addison-Wesley, 2003.

[79] E. Exposito, M. Gineste, R. Peyrichou, P. Senac, and M. Diaz, "XQoS: XML-based QoS specification language," in *Intl. Conf. on MultiMedia Modeling*, Jan. 2003.

[80] G. Camarillo, W. Marshall, and J. Rosenberg, "Integration of resource management and session initiation protocol (SIP)," *IETF RFC 3312*, Oct. 2002.

[81] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," *IETF RFC 3261*, Jun. 2002.

[82] R. Guerin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Transaction on Networking*, vol. 7, no. 3, Jun. 1999.

[83] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[84] P. Paul and S. Raghavan, "Survey of qos routing," in *International Conference on Computer Communication (ICCC)*, 2002.

[85] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," *IETF RFC 1633*, Jul. 1994.

[86] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," *IETF RFC 2212*, Sep. 1997.

[87] J. Wroclawski, "Specification of the controlled-load network element service," *IETF RFC 2211*, Sep. 1997.

[88] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP)," *IETF RFC 2205*, Sep. 1997.

[89] J. Harju and P. Kivimaki, "Co-operation and comparison of diffserv and intserv: performance measurements," in *IEEE Intl. Conf. on Local Computer Networks (LCN)*, 2000.

[90] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie, "Aggregation of rsvp for ipv4 and ipv6 reservations," *IETF RFC 3175*, Sep. 2001.

[91] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *IETF RFC 2475*, Dec. 1998.

[92] B. Davie, A. Charny, J. Bennett, K. Benson, J. Leboudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB (Per-Hop Behavior)," *IETF RFC 3246*, Mar. 2002.

[93] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *IETF RFC 2597*, Jun. 1999.

[94] G. Stattenberger, T. Braun, M. Scheidegger, M. Brunner, and H. Stuttgen, "Performance evaluation of a linux DiffServ implementation," *Computer Communications Journal*, vol. 25, no. 13, 2003.

[95] "Vat audio conferencing tool, lawrence berkeley national laboratory," *http://www-nrg.ee.lbl.gov/vat/*.

[96] "INRIA videoconferencing system (ivs)," *http://www-sop.inria.fr/rodeo/ivs.html*.

[97] S. Ardon, P. Gunningberg, B. Landfeldt, Y. Ismailov, M. Portmann, and A. Seneviratne, "MARCH: a distributed content adaptation architecture," *International Journal of Communication Systems*, vol. 16, Oct. 2002.

[98] J. Postel, "Transmission control protocol," *IETF RFC 793*, Sep. 1981.

[99] E. Exposito, "Specification and implementation of a QoS oriented transport protocol for multimedia applications," *PhD Dissertation, Institut National Polytechnique de Toulouse*, Dec. 2003.

[100] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream control transmission protocol," *IETF RFC 2960*, Oct. 2000.

[101] E. Kohler, M. Handley, and S. Floyd, "Datagram congestion control protocol (DCCP)," *IETF RFC 4340*, Mar. 2006.

[102] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," *IETF RFC 3448*, Jan. 2003.

[103] E. Exposito, P. Senac, and M. Diaz, "FPTP: the XQoS aware and fully programmable transport protocol," in *IEEE Intl. Conference on Local Networks (ICON)*, Sep. 2003.

[104] D. G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[105] T. Fei, S. Tao, L. Gao, and R. Guérin, "How to select a good alternate path in large peer-to-peer systems," in *IEEE INFOCOM*, April 2006.

[106] "Skype," *http://www.skype.com*.

[107] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network scheme," in *IEEE Communications Survey and Tutorial*, March 2004.

[108] T.Rakotoarivelo, P. Senac, A. Seneviratne, and M. Diaz, "A structured peer-to-peer method to discover qos enhanced alternate paths," in *IEEE International Conference on Information Technology and Applications (ICITA)*, July 2005.

[109] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," *IETF RFC 1771*, Mar. 1995.

[110] V. Paxson, "End-to-end routing behavior in the Internet," *ACM SIGCOMM Computer Communication Review*, 1996.

[111] D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of tcp processing overhead," *IEEE Communication Magazine (vol. 27)*, no. 6, June 1989.

[112] S. Makineni and R. Iyer, "Measurement-based analysis of tcp/ip processing requirements," in *Annual International Conference on High Performance Computing (HiPC 2003)*, 2003.

[113] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path properties," in *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[114] N. Brownlee and K. Claffy, "Understanding internet traffic streams: Dragonflies and tortoises," *IEEE Communications*, 2002.

[115] S. Moon, "Measurement and analysis of end-to-end delay and loss in the internet," *PhD Dissertation, University of Massachussets at Amherst*, 2000.

[116] Y. Pyun and D. Reeves, "Constructing a balanced, (log(n)/loglog(n))-diameter super-peer topology for scalable p2p systems," in *Intl. Conference on Peer-to-Peer Computing (P2P)*, 2004.

[117] K. Gummadi, S. Saroiu, and S. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.

[118] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz, "Scalable and accurate identification of as-level forwarding paths," in *IEEE INFOCOM*, March 2004.

[119] "Ripe routing information service," *http://www.ripe.net/projects/ris/*.

[120] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *IEEE Conference on Peer-to-peer Computing (P2P)*, 2001.

[121] D.Stauffer and A. Aharony, *Introduction to Percolation Theory*. Taylor and Francis, 1992.

[122] A. Ganesh, A.-M. Kermarrec, and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols," *IEEE Transactions on Computers*, vol. 52, 2003.

[123] F. Banaei-Kashani and C. Shahabi, "Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems," in *Intl. Symposium on Cluster Computing and the Grid*, 2003.

[124] "Brite topology generator," *http://www.cs.bu.edu/brite*.

[125] Q. Zhang, F. Yang, W. Zhu, and Y. Zhang, "A construction of locality-aware overlay network: moverlay and its performance," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, 2004.

[126] M. Allman, W. Eddy, and S. Ostermann, "Estimating loss rates with tcp," *ACM Performance Evaluation Review*, vol. 31, no. 3, pp. 12–24, December 2003.

[127] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *ACM SIGCOMM Conference*, 1998, pp. 303–314.

[128] H. Perros and K. Elsayed, "Call admission control schemes: a review," *IEEE Communications Magazine*, vol. 34, no. 11, Nov. 1996.

[129] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," in *ACM SIGCOMM*, 1995.

[130] M. Ahmed, "Call admission control in wireless networks: a comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 7, no. 1, 2005.

[131] "PlanetLab project," *http://www.planet-lab.org/*.

[132] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, 1997.

[133] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. V. Mieghem, "Analysis of end-to-end delay measurements in internet," in *Passive and Active Measurement (PAM)*, 2002, pp. pp. 26–33.

[134] A. Mukherjee, "On the dynamics and significance of low frequency components of internet load," *University of Pennsylvania, Technical Report MS-CIS-92-83*, 1992.

[135] V. Roth, J. Laub, M. Kawanabe, and J. Buhmann, "Optimal cluster preserving embedding of nonmetric proximity data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, Dec. 2003.

[136] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *IEEE Computer*, vol. 38, 2005.

[137] "GENI: Global Environment for Network Innovations," *http://www.geni.net/*.

[138] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

Glossary

AAC	*Advanced Audio Coding*
API	*Application Programming Interface*
AS	*Autonomous System*
ATM	*Asynchronous Transfer Mode*
BGP	*Border Gateway Protocol*
DAML	*DARPA Agent Markup Language*
EGP	*Exterior Gateway Protocol*
FEC	*Forward Error Correction*
IETF	*Internet Enginneering Task Force*
IGP	*Interior Gateway Protocol*
IP	*Internet Protocol*
IPv6	*Internet Protocol version 6*
ISP	*Internet Service Provider*
NCP	*Network Control Protocol*
NLANR	*National Laboratory for Applied Network Research*
OIL	*Ontology Inference Layer*
OWL	*Web Ontology Language*
P2P	*Peer-to-Peer*
QEAP	*QoS Enhanced Alternate Path*
QoS	*Quality of Service*
RDF	*Resource Description Framework*
RDFS	*RDF Schema*
RIPE	*Réseau IP Européen*
RSVP	*Resource ReserVation setup Protocol*
RTT	*Round Trip Time*
SLA	*Service Level Agreement*
SDP	*Session Description Protocol*
SDP	*Session Initiation Protocol*
SPAD	*Super-Peer Alternate path Discovery*
TFRC	*TCP Friendly Rate Control*
TCP	*Transmission Control Protocol*
TTL	*Time To Live*
W3C	*World Wide Web Consortium*
WSDL	*Web Service Description Language*
XML	*Extensible Markup Language*

List of Figures

3.1 Deployment of a Composite Service S between hosts X and Y. 32
3.2 Example of an Overlay Network. 38

4.1 A network topology example to illustrate QoS routing. 46
4.2 Resource Reservation within a Network implementing the IntServ model. . . 48
4.3 Transmission of a data flow across Networks implementing the DiffServ model. 49
4.4 Comparison between a QoS enhanced alternate path (from host A to B, via a relay host R) and the default Internet path 52

5.1 deployment of a QEAP between two hosts H_1^A and H_1^Z via a relay host H_1^C 58
5.2 Distribution of delay and packet-loss gains for the different data sets. 63
5.3 Cumulative distribution of the delay constancy duration on QEAPs. 65
5.4 Scatter-plot of QEAPs according to their Delay and Packet-Loss gains (by definition a QEAP cannot have both gains < 0). 66
5.5 Delay gain comparison: 3-hop versus 2-hop QEAPs (RIPE-TTM 25/05/05). 66

6.1 A preliminary simple distributed QEAP discovery scheme. 71
6.2 A Super-Peer network (SP: super-peers; NP: normal-peers). 74
6.3 (a) Structure of a SPAD module within a middleware framework on each SPAD peer, (b) Details of the "Discovery & Management module" 74
6.4 Comparison between *delayAtoB* and *delayBtoA* (RIPE-TTM 25/05/04). . . 76
6.5 Comparison between *delayAtoB* and *delayBtoA* (NLANR-AMP 30/01/03). . 76
6.6 Distribution of the standard deviation of the delay (U,V), for all AS pairs (PlaneLab 27/04/05). 77
6.7 Cumulative distribution on all AS pairs of the standard deviation of the delay (U,V), (PlaneLab 27/04/05). 77
6.8 Pseudocode for the query-processing algorithm on SPAD super-peers. 80
6.9 Pseudocode for the filter algorithm on SPAD super-peers. 80
6.10 Amount of reached nodes for different dissemination probabilities. 83
6.11 Graph and tree examples for the theoretical analysis of SPAD's information dissemination strategy. 84
6.12 Comparison of information dissemination on experimental topologies versus theoretical model implied by equations (6.5) and (6.9). 86
6.13 Pseudocode of the Incoming Entry Processing algorithm executed by SPAD super-peers. 87

7.1 Performance comparison between SPAD reactive scheme and other QEAP discovery schemes (NLANR-AMP 30/01/03). 100
7.2 Fraction of discovered QEAPs among all the discoverable ones, for different TTL_Q (i.e. Query TTL) and information dissemination reach D. 101
7.3 Comparison of the throughput on the selected QEAPs (Q_{sel}) from different Selection Scheme (Cumulative Distribution). 104
7.4 Comparison of the gain in packet-loss rate on the selected QEAPs (Q_{sel}) from different Selection Scheme (Cumulative Distribution). 104
7.5 Comparison of the gain in delay on the selected QEAPs (Q_{sel}) from different Selection Scheme. 104
7.6 Variations of the computed TCP-friendly throughput (from equation (6.19)) for different values of delay and packet-loss rate. 105

1 Descriptions of the simple SPAD prototype and the demonstration test-bed built with it. 116

List of Tables

5.1 Characteristics of the Experimental Data Sets 59
5.2 Quantity and Type of Existing QEAPs 61
5.3 Average Delay and Packet-Loss Gains on QEAPs 62
5.4 Comparison between 2-hop and 3-hop QEAPs 67

7.1 Performance of the Simple QEAP Discovery Scheme (RIPE-TTM 25/05/04) 98
7.2 Performance of the SPAD Reactive Scheme (NLANR-AMP 30/01/03) . . . 100
7.3 Search Cost in Messages for Different TTL_Q (PlanetLab 11/04/05) 102
7.4 Search Latency for TTL_Q (PlanetLab 11/04/05) 102

Découverte et Gestion Distribuée de Chemins Alternatifs à contraintes de Qualité de Service dans l'Internet

Résumé La convergence de récentes avancées technologiques permet l'émergence de nouveaux environnements informatiques pervasifs, dans lesquels des terminaux en réseaux coopèrent et communiquent de manière transparente pour les utilisateurs. Ces utilisateurs demandent des fonctionalités de plus en plus avancées de la part de ces terminaux. Etant données les limites intrinsèques des terminaux mobiles, ces fonctionalités, au lieu d'être directement implémentées dans les terminaux, sont appelées à être fournies par des *fournisseurs de services* situés à la périphérie du réseau. Ce derniers devient alors une source illimitée de services, et non plus seulement un medium de communication. Ces services, ou *applications d'overlays*, sont formés de plusieurs éléments applicatifs distribués qui coopèrent et communiquent entre eux via un réseau de recouvrement dynamique particulier, une *association d'overlay*. La Qualité de Service (QdS) perçue par les utilisateurs d'une *application d'overlay* dépend de la QdS existant au niveau des chemins de communications qui forment l'*association d'overlay* correspondante.
Cette thèse montre qu'il est possible de fournir de la QdS à une *application d'overlay* en utilisant des chemins Internet alternatifs, résultant de la composition de chemins distincts. De plus, cette thèse montre également qu'il est possible de découvrir, sélectionner, et composer d'une manière distribuée ces chemins élémentaires, au sein d'une communauté comprenant un nombre important d'entités paires (telles que les précédents *fournisseurs de services*). Les principales contributions de cette thèse sont : i) une description et une analyse des caractéristiques de QdS de ces chemins alternatifs composés, ii) une architecture originale appelée SPAD (Super-Peer based Alternate path Discovery), qui permet la découverte et la sélection de manière distribuée de ces chemins alternatifs. SPAD est un système complètement décentralisé, qui peut être facilement et incrémentalement déployé sur l'Internet actuel. Il permet aux utilisateurs situés à la périphérie du réseau de découvrir et d'utiliser directement des chemins alternatifs.

Mots clés Qualité de Service, Réseaux de Recouvrement, Système Pair-à-Pair, Réseaux de Services

---

Distributed Discovery and Management of Alternate Paths with enhanced Quality of Service in the Internet

Abstract The convergence of recent technology advances opens the way to new ubiquitous environments, where network-enabled devices collectively form invisible pervasive computing and networking environments around the users. These users increasingly require extensive applications and capabilities from these devices. Recent approaches propose that cooperating *service providers*, at the edge of the network, offer these required capabilities (i.e *services*), instead of having them directly provided by the devices. Thus, the network evolves from a plain communication medium into an endless source of services. Such a service, namely an *overlay application*, is composed of multiple distributed application elements, which cooperate via a dynamic communication mesh, namely an *overlay association*. The Quality of Service (QoS) perceived by the users of an *overlay application* greatly depends on the QoS on the communication paths of the corresponding *overlay association*.
This thesis asserts and shows that it is possible to provide QoS to an *overlay application* by using alternate Internet paths resulting from the compositions of independent consecutive paths. Moreover, this thesis also demonstrates that it is possible to discover, select and compose these independent paths in a distributed manner within an community comprising a limited large number of autonomous cooperating peers, such as the fore-mentioned *service providers*. Thus, the main contributions of this thesis are i) a comprehensive description and QoS characteristic analysis of these composite alternate paths, and ii) an original architecture, termed SPAD (Super-Peer based Alternate path Discovery), which allows the discovery and selection of these alternate paths in a distributed manner. SPAD is a fully distributed system with no single point of failure, which can be easily and incrementally deployed on the current Internet. It empowers the end-users at the edge of the network, allowing them to directly discover and utilize alternate paths.

Keywords Quality of Service, Overlay Networks, Peer-to-Peer Systems, Service-oriented Networks