A realistic named data networking architecture for the Internet of things

Amar Abane

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Centre d'Études et de Recherche en Informatique et Communications

Faculté de Génie Électrique et d'Informatique

Laboratoire de Recherche en Informatique

THÈSE DE DOCTORAT

présentée par : **Amar ABANE**

soutenue le : **02 Décembre 2019**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Spécialité : **Informatique**

A Realistic Named Data Networking Architecture for the Internet of Things

THÈSE dirigée par	
Mme. Bouzefrane Samia	*Professeur, CNAM*
M. Daoui Mehammed	*Professeur, UMMTO*
RAPPORTEURS	
Mme. Benzaid Chafika	*Maître de conférences, USTHB*
M. Baccelli Emmanuel	*Directeur de recherche, Inria Saclay*
EXAMINATEURS	
M. Afifi Hossam	*Professeur, Telecom ParisTech*
M. Laghrouche Mourad	*Professeur, UMMTO*
INVITÉ	
M. Muhlethaler Paul	*Directeur de recherche, Inria Paris*

Résumé

L'Internet des objets (IdO) utilise l'interconnexion de milliards de petits appareils informatiques, appelés «Objets», pour fournir un accès à des services et à des informations partout dans le monde. Cela a été rendu possible par la démocratisation des smartphones et des ordinateurs portables, et plus important encore, par le caractère abordable des dispositifs d'acquisition de données à ressources limitées et des technologies de communication sans fil correspondantes. Cependant, la pile de protocoles IP sur laquelle est basée l'IdO a été conçue il y a plusieurs décennies dans un but totalement différent, et les fonctionnalités de l'IoT soulignent désormais les limites de l'IP. Néanmoins, l'IP peut toujours prendre en charge les systèmes IdO via des couches logicielles intermediaires, à savoir CoAP, 6LoWPAN, RPL, REST et autres solutions. Toutefois, les efforts considérables déployés dans les solutions IP actuelles consistent simplement à faire en sorte que les périphériques IdO prennent en charge la suite de protocoles IP existante, alors que de nombreuses autres nouvelles fonctionnalités doivent être incluses dans le réseau. En parallèle aux efforts d'adaptation de l'IP à l'Internet des objets, des architectures alternatives basées sur les réseaux orientés information (Information Centric Networking) promettent de satisfaire nativement les applications Internet émergentes. L'une de ces architectures est appelée réseau de données nommées (Named Data Networking). Nos objectifs à travers le travail rapporté dans ce manuscrit peuvent être résumés en deux aspects. Le premier objectif est de montrer que NDN est adapté à la prise en charge des systèmes IdO. Pour y parvenir, une discussion détaillée sur les limites de l'IP et les fonctionnalités

NDN est présentée, suivie de la conception et du déploiement d'une architecture NDN-IdO réaliste montrant la simplicité de NDN. De plus, un outils de simulation des reseaux NDN pour l'IdO et une modélisation des reseaux NDN sans fil sont proposés pour quantifier et analyser les avantages de NDN dans notre contexte. Le deuxième objectif est la conception de deux solutions de communication legères pour les réseaux sans fil contraints avec NDN. Ces deux solutions prennent en compte la technologie IEEE 802.15.4 et utilisent uniquement des communications de diffusion. La première solution repose sur une technique d'apprentissage par renforcement et fonctionne au niveau de la couche réseau, tandis que la seconde est inspirée de l'accès au support basé sur les priorités et fonctionne au niveau de la couche liaison.

Mots clés : Réseaux de données nommées, Réseaux orientés information, Internet des Objets, IEEE 802.15.4, Réseaux sans fil.

Abstract

The Internet of Things (IoT) uses the interconnection of billions of small computing devices, called "Things", to provide access to services and information all over the world. This has been made possible by the democratization of smartphones and laptops, and more importantly by the affordability of resource-constrained data acquisition devices and corresponding wireless communication technologies. However, the IP protocol stack which IoT uses currently has been designed decades ago for a completely different purpose, and IoT features now highlight the limitations of IP. That said, IP can still support IoT systems through adaptations and middleware, namely CoAP, 6LoWPAN, RPL, REST and other solutions. However, the significant efforts expended in current IP solutions is just to make IoT devices support the existing IP protocol suite, whereas many other new features have to be included in devices. While adapting IP for the IoT might be seen as cutting corners, alternative architectures based on the Information Centric Networking (ICN) paradigm promise to natively satisfy emerging Internet applications. One of these architectures is Named Data Networking (NDN). Our objectives through the work reported in this manuscript can be summarized in two aspects. The first objective is to show that NDN is suitable to support IoT networking. To achieve that, a detailed discussion on IP limits and NDN features is presented, followed by the design and deployment of a realistic NDN-IoT architecture that shows the simplicity of NDN. Moreover, a simulation framework for NDN-IoT and a model for NDN wireless forwarding are proposed to quantify and analyse the advantages of NDN. The second objective is the design of two so-

lutions for lightweight forwarding in constrained wireless networks. These two solutions consider the IEEE 802.15.4 technology and use only broadcast communications. The first solution is based on a reinforcement learning scheme and operates at network layer, while the second is inspired of priority-based medium access and operates at link-layer.

Keywords: NDN, ICN, IoT, IEEE 802.15.4, Broadcast, Wireless Networks

Dedication

In memory of my father Abane Boussad.

In memory of my granduncle Abane Ramdane, hero of the Algerian war.

***

À la mémoire de mon père Abane Boussad.

À la mémoire de mon grand-oncle Abane Ramdane, héros de la guerre d'Algérie.

Contents

List of Tables

List of Figures

Résumé de la Thèse

Une Architecture NDN realiste pour l'Internet des Objets

Introduction

L'Internet des objets (Internet of Things, IoT) utilise l'interconnexion de milliards de petits appareils informatiques, appelés «Things», pour fournir un accès à des services et à des informations partout dans le monde (Figure 1). Cela a été rendu possible par la démocratisation des smartphones et des ordinateurs portables, et plus important encore, par le caractère abordable des dispositifs d'acquisition de données à ressources limitées tels que les capteurs et des technologies de communication sans fil correspondantes. À titre d'exemple, les coûts des capteurs ont diminué de près de 200% entre 2004 et 2016. En conséquence, quatre fois plus d'objets seront connectés à Internet d'ici la fin de cette année. Le marché est évidemment impacté par cette évolution. On prévoit que le marché de l'IoT passera de 656 millions de dollars en 2014 à 1,7 milliards en 2020. L'impact économique de l'IoT pourrait être compris entre 3,1 et 3,9 milliards de dollars par an d'ici 2025.

En pratique, les appareils de l'IoT sont alimentés par batterie, disposent de processeurs à faible puissance et quelques dizaines de Kb de mémoire. Dans un système IoT, ces périphériques contraints communiquent entre eux ou avec les applications des utilisateurs via Internet. Cette communication est réalisée via la suite de protocoles TCP/IP et l'interconnexion sans fil est réalisée à l'aide de technologies sans fil à faible consommation.

Cependant, la suite de protocoles IP a été conçue il y a plusieurs décennies dans un but totalement différent, et les fonctionnalités de l'IoT soulignent désormais les limites de l'IP. Par exemple, la sécurité est toujours centrée sur les canaux de communication lorsque les données elles-mêmes doivent être sécurisées. De plus, les systèmes IoT ont besoin d'une prise en charge efficace pour la dénomination et la découverte des ressources, ce qui n'est pas facile à déployer avec l'IP dans des infrastructures contraintes. Cela dit, l'IP peut toujours prendre en charge les systèmes IoT via des adaptations de protocoles. C'est la raison pour laquelle nous entendons parler de CoAP, 6LoWPAN, RPL, REST et autres solutions. Toutefois, les efforts considérables déployés dans les solutions IP actuelles consistent simplement à faire en sorte que les périphériques IoT supportent la suite de protocoles IP existante, alors que de nombreuses autres nouvelles fonctionnalités doivent être incluses dans les périphériques. Ainsi, si le support de la communication dans les appareils pouvait être simplifié et rendu robuste, il constituerait un catalyseur fondamental pour un écosystème IoT global. De plus, la simplification des solutions de communication pour les applications IoT réduira considérablement les coûts de développement.

Alors que l'adaptation de l'IP à l'Internet des objets se poursuit toujours, des architectures alternatives suivant le paradigme des réseaux orientés sur les contenus (Information Centric Networking, ICN) promettent de satisfaire nativement les applications Internet émergentes. L'une de ces architectures est appelée réseaux de données nommées (Named Data Networking, NDN). Le projet NDN a été financé par la National Science Foundation (NSF) dans le cadre du programme Future Internet Architecture (FIA). Dans un réseau NDN, l'entité principale est le contenu, tel qu'une video, une page Web, etc. Les opérations de communication sont effectuées sur les noms des ces contenus, et les hôtes (sans adresses logiques) récupèrent les contenu nommés directement du réseau. Des fonctions importantes sont obtenues via ce principe; telles que la communication de bout en bout sans établir de connexion ni de résolution de nom à adresse. De plus, aucune session consommateur-fournisseur n'a besoin d'être maintenu, ce qui fournit une prise en charge native des interruptions de connexion résultant de la mobilité.

Un réflexe naturel que nous pouvons avoir lorsque nous entendons parler de NDN est de savoir comment tirer parti de ses fonctionnalités sans attendre durant des décennies la future architecture de l'Internet. Cependant, des modifications fondamentales doivent être apportées aux équipements, aux protocoles et aux applications actuelles basés sur l'IP, car le paradigme NDN fonctionne sur des noms des contenus plutôt que sur des adresses d'hôtes. En outre, tant que les solutions IP fonctionnent pour les applications actuelles, il reste difficile de convaincre les utilisateurs d'IP et les industriels des avantages de NDN. Heureusement, ces dernières années, de nombreuses études ont examiné l'utilité de NDN dans l'IoT, le rendant de plus en plus puissant. Avec tout ce travail motivant, de réels déploiements de NDN peuvent être envisagés. Bien que NDN ne soit pas prêt pour les déploiements IoT globaux, des conceptions réalistes enrichiront les expériences NDN et aideront à déterminer ce qui est nécessaire pour faire de NDN une réalité.

Pour y parvenir, l'intégration de NDN dans les appareils de l'IoT et des technologies sans fil à faible consommation est primordiale en raison de l'importance de ces technologies dans les solutions IoT. De plus, l'IoT est encore au stade de développement, même avec l'IP, ce qui permet de faire de NDN un élément important des solutions IoT dans un court laps de temps. Pour saisir cette opportunité, nous avons basé notre travail sur un déploiement réaliste de NDN dans l'IoT. Le concept de réalisme dans ce contexte comprend de nombreux aspects. Tout d'abord, il vise à intégrer NDN dans l'infrastructure Internet actuelle. Autrement dit, les scénarios qui ne peuvent pas être déployés maintenant ne sont pas pris en compte. Deuxièmement, réaliste signifie utiliser NDN pour concevoir des solutions pour les équipements IoT actuels tels que les cartes de prototypage. Troisièmement, l'objectif d'une approche réaliste est de fournir une solution NDN-IoT viable qui doit être facile à utiliser, à faible coût, simple et légère. Enfin, l'approche réaliste envisagée ici s'inscrit dans le même esprit de deux ou trois amis cherchant à lancer un produit informatique révolutionnaire grâce à une start-up.

Des études indiquent que 50% de toutes les solutions IoT sont développées par des start-ups ou des petites entreprises qui souvent ne possèdent pas les ressources financières pour concevoir et réaliser un produit IoT prêt à être commercialisé. Par conséquent, pour réussir à créer une application IoT, les startupers utilisent souvent une validation de principe (Proof of Concept, PoC) pour montrer que leur solution peut connaître un succès commercial. Ceci est généralement réalisé à l'aide de systèmes sur puce (System on Chip, SoC), de micro-contrôleurs et d'ordinateurs à carte unique tels que Arduino, BeagleBoard et Raspberry Pi. De plus, le fait de disposer d'un prototype opérationnel augmente considérablement les chances de financement d'une entreprise.

Cette thèse propose une vision en deux étapes sur la manière dont ICN/NDN peut être considéré dans l'IoT. La première étape est une opportunité d'explorer certaines solutions IP populaires pour l'IoT et de comprendre comment NDN peut être introduit dans les systèmes IoT. Après cela, les communications NDN dans les réseaux sans fil restreints est identifié comme l'un des problèmes à résoudre pour que NDN devienne une réalité dans les solutions IoT. La deuxième étape nous permet d'explorer en détail les communications NDN sans fil dans les réseaux soumis à des contraintes, ce qui est un aspect important à la fois pour l'IoT et NDN. Par conséquent, nos objectifs à travers le travail rapporté dans ce manuscrit peuvent être résumés en deux aspects. Le premier objectif est de montrer que NDN est adapté à la prise en charge d'un réseau IoT. Pour y parvenir, une discussion détaillée sur les limites IP et les fonctionnalités NDN sera présentée en premier lieu, suivie de la conception et du déploiement d'une architecture NDN réaliste pour l'IoT utilisant IEEE 802.15.4 (NDN-802.15.4), qui montre la simplicité de NDN et la faisabilité de notre approche. De plus, un outil de simulation pour NDN-802.15.4 et un modèle mathématique pour les communications NDN sans fil sont proposés pour quantifier et analyser les avantages de NDN. Le deuxième objectif est la conception de deux solutions de transfert légeres dans des réseaux sans fil contraints. Ces deux solutions prennent en compte la technologie IEEE 802.15.4 et utilisent uniquement des communications de diffusion. L'utilité de la diffusion dans les réseaux NDN sera également étudiée. En bref, la première solution repose sur un schéma d'apprentissage par renforcement et fonctionne au niveau de la couche réseau, tandis que la seconde repose sur un accès au cannal de transmission (CSMA) basé sur les priorités et fonctionne au niveau de la couche liaison.

IP et ICN : Les défis de l'IoT

En moins de 40 ans, les réseaux de protocole Internet (IP) ont créé l'Internet et son contenu actuel, mais les réseaux IP n'ont pas été conçus pour cela. C'est une réflexion intéressante pour résumer l'évolution de l'Internet et sa situation actuelle avec l'IoT et les applications émergentes. Dans cette partie de la thèse, nous allons expliquer cette idée et montrer pourquoi elle est vraie. Nous considérons que l'IoT est un bon point de vue pour comprendre les limites de l'IP car il apporte des exigences qui reflètent l'évolution de la technologie et de la société. Par conséquent, nous étudions comment l'architecture IP prend en charge les systèmes IoT et nous exposons les solutions et leurs inconvénients (Figure 2). Nous pensons que le fait de discuter l'approche IP pour l'IoT et ses faiblesses est un moyen rationnel de montrer l'opportunité que représente le concept ICN pour la construction de meilleurs systèmes IoT. De plus, nous montrons que les solutions IP pour l'IoT ressemblent aux fonctionnalités natives offertes par ICN dans la couche réseau. Nous présentons donc l'approche réseau ICN en tant que solution regroupant les fonctionnalités essentielles requises pour répondre efficacement aux exigences de l'IoT.

Le concept de ICN a été introduit par Ted Nelson en 1979. Vingt ans plus tard, l'architecture TRIAD (Translating Relaying Internetwork Architecture) intégrant Active Directory (ADI) a été proposée comme architecture Internet de nouvelle génération pour éviter les recherches DNS. En 2002, Brent Baccala a publié un article présentant les différences entre la mise en réseau orientée hôte et la mise en réseau orientée données. En 2006, l'architecture de réseau orientée données (DONA) à Berkeley proposait la première architecture ICN, suivie quelques années plus tard (2009) par PARC qui annonçait l'architecture CCN et l'implémentation open source CCNx. En septembre 2010, le projet NDN a été fondé dans le cadre du programme FIA (Future Internet Architecture) de la NSF.

Les fonctionnalités natives d'ICN sont prises en charge différemment d'une réalisation à l'autre. Il est à noter qu'aucune architecture ICN n'a été spécialement conçue pour l'IoT. Néanmoins, l'approche ICN est toujours dans une phase de recherche, ce qui est une opportunité pour concevoir de futures architectures en tenant compte les applications de l'IoT.

Parmi les architectures ICN apparues ces dernières années, la mise en réseau de données nommées (NDN) est une des plus prometteuse. Par conséquent, plutôt que de présenter les fonctionnalités ICN abstraites, nous verrons dans la suite de la thèse comment la conception NDN fournit des fonctionnalités ICN et comment elle peut prendre en charge les applications IoT, de manière native ou avec quelques adaptations simples.

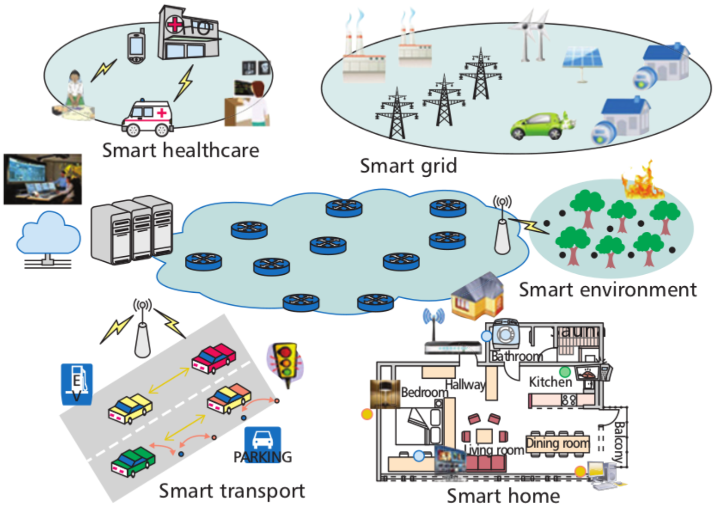


Figure 1: Aperçu général de l'IoT [Amadeo et al. 2016]

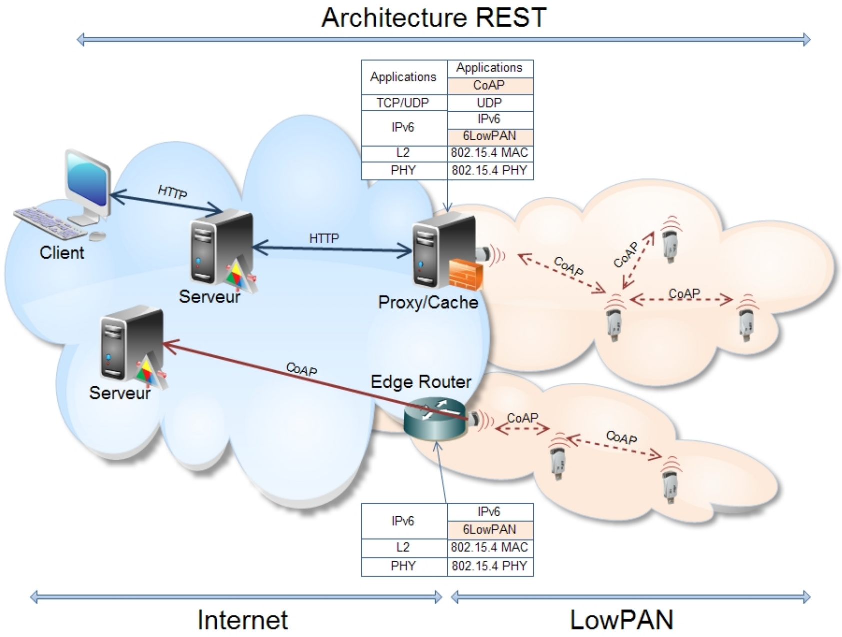


Figure 2: Les solutions IP pour l'IoT [Wikipedia a]

Les réseaux de données nommées et l'IoT

Nous avons compris précédemment que la pile de protocoles IP ne convient pas à un écosystème global de l'IoT. Au cours des dernières années, NDN est devenu une architecture prometteuse capable de supporter de manière efficace les contraintes de l'IoT. Pour donner une idée simple de NDN, on peut l'imaginer comme le modèle de requête-réponse de HTTP exécuté au niveau de la couche réseau. La principale différence avec HTTP réside dans le fait que NDN prend en charge le modèle requête-réponse par le biais de paquets portant des noms comme information principale, et que toutes les opérations de mise en réseau (acheminement, routage, etc.) fonctionnent sur ces noms et non sur des adresses réseau binaires. Cependant, nous devons observer que NDN est plus qu'un cas de décalage de HTTP vers la couche réseau. Deux différences importantes doivent être soulignées: (i) Dans le NDN, les paquets de données sont immuables; c'est-à-dire qu'une fois qu'une donnée est produite avec un certain nom, elle ne peut plus être modifiée. Lorsqu'une nouvelle version des données est disponible, le producteur doit générer un nouveau paquet avec un nouveau nom. (ii) chaque paquet de données est auto-sécurisé en portant une signature numérique qui lie son nom à son contenu. Cette signature est générée par le producteur au moment de la création du paquet. Lors de la récupération des données, le consommateur vérifie la signature pour s'assurer que le contenu correspond au nom demandé et a bien été produit par la bonne entité. Cette approche de sécurité fournit à NDN une sécurité basée sur le contenu au lieu de sécuriser les canaux de communication entre deux hôtes.

Pour montrer comment NDN convient aux architectures IoT, nous rapportons des études et des propositions que nous considérons comme une source d'inspiration pour les contributions présentées plus loin dans cette thèse. Malgré les diverses études liées à NDN pour l'IoT, il n'existe actuellement aucun chemin de développement clair pour le réseau ICN/NDN qui pourrait être utilisé pour montrer la supériorité du NDN sur l'IP. En pratique, des différences fondamentales telles que la mise en cache et la dénomination des données rendent difficile l'établissement de comparaisons directes équitables entre NDN et IP. Par conséquent, notre objectif est de fournir un environnement complet permettant d'étudier les avantages de NDN dans les déploiements IoT actuels. Plutôt que de discuter des défis NDN globaux pour le futur Internet, nous avons choisi d'étudier la faisabilité d'un déploiement IoT avec NDN dans un scénario typique avec des équipements IoT populaires (e.g., Arduino). Par conséquent, la première étape consiste à concevoir et à déployer une architecture réaliste basée sur NDN, qui souligne les principaux défis à relever. Dans ce qui suit, nous proposons une telle architecture, en précisant ses composants et les principaux défis qu'elle soulève.

Une architecture NDN réalise pour l'IoT

Nous avons compris jusqu'ici que NDN est potentiellement plus approprié que l'IP pour construire des solutions IoT efficaces. Son modèle de communication ne nécessite ni l'attribution ni la gestion d'adresses et opère directement sur le contenu nommé des applications. Il sécurise le contenu de bout en bout quels que soient les protocoles de transport ou les canaux. Cela produit des paquets de données sécurisés réutilisables et donne à NDN une prise en charge native de la mise en cache, de la diffusion et de la multidiffusion. De plus, NDN n'utilise pas de format de paquet prédéfini ni d'exigences minimales en matière de MTU, et sa simplicité permet de mettre en œuvre des implémentations avec une taille de code plus petite dans les périphériques. Cependant, le déploiement pratique de NDN doit être défini pour tirer parti de ces fonctionnalités dans les solutions IoT actuelles. En effet, l'intégration de NDN dans l'infrastructure Internet actuelle est vitale et aura un impact sur de nombreux aspects liés au réseau et aux applications. Actuellement, un déploiement global de NDN n'est pas réalisable en raison de la différence entre les paradigmes IP et NDN. Pratiquement, pour déployer le protocole NDN sur des réseaux IP, les hôtes et les routeurs doivent au minimum prendre en charge le routage basé sur le nom, le traitement de paquets et mettre en œuvre certaines stratégies d'acheminement. De plus, les solutions à court terme nécessitent la coexistence de NDN et IP dans le même réseau global. Pour cela, nous devons nous assurer que les périphériques NDN et IP n'interfèrent pas les uns avec les autres, tout en garantissant qu'un tel déploiement entraînera des avantages croissants pour les applications.

Gardant cela à l'esprit, l'intégration de NDN telle qu'elle est envisagée dans cette thèse se veut réaliste et progressive. C'est-à-dire que nous proposons une conception basée sur NDN pour l'IoT pouvant coexister avec l'infrastructure IP et les équipements IoT actuels. Par cela, nous visons à rendre NDN facilement accessible en l'activant du côté des objets de l'IoT. En d'autres termes, nous fournissons aux appareils de l'IoT une identité de couche réseau (L3) centrée sur les données, plus naturelle et plus efficace que l'identité IP actuelle. Nous commençons tout d'abord par identifier et discuter des approches d'intégration NDN possibles en fonction des travaux correspondants. Au niveau de la couche réseau, un déploiement NDN nécessite que les entités prennent en charge le routage et le traitement des paquets basés sur les noms, ainsi que la mise en œuvre de stratégies d'acheminement et des procédures de sécurité. En outre, davantage d'espace de stockage est nécessaire pour la mise en cache et les structures de données nécessaires à l'acheminement des paquets. Considérant l'infrastructure de réseau IP globale, le NDN peut être déployé en superposition sur IP, il peut remplacer IP en tant que protocole de réseau natif sur la couche liaison (e.g., NDN sur Ethernet), ou IP et NDN peuvent coexister dans le même réseau.

La première approche, la superposition, est facile à déployer et crée une couche uniforme centrée sur le contenu. Le banc d'essai NDN déployé modialement est un exemple d'une telle approche. Cependant, cette solution crée une complexité et une surcharge pour le protocole réseau sous-jacent, et les applications basées sur IP doivent supporter les noms NDN pour pouvoir utiliser le réseau comme décrit plus haut. De plus, la superposition considère NDN comme un protocole de transport ou d'application pour IP, et ne permet donc pas une coexistence entre les deux protocoles de réseau (c'est-à-dire IP et NDN). Plus important encore, la mise en œuvre de piles NDN et IP n'est pas réalisable avec des périphériques contraints de l'IoT qui peuvent à peine supporter la pile IP actuelle.

La seconde approche, qui consiste à déployer NDN en tant que protocole de réseau natif, ne fonctionne que pour les environnements qui n'ont pas besoin de communiquer avec les réseaux IP globaux, tels que les réseaux de véhicules isolés ou les réseaux locaux.

La troisième et dernière approche consiste à faire coexister IP et NDN au sein du réseau mondial. Cette approche peut soit utiliser NDN au cœur du réseau et conserver l'IP à la périphérie du (NDN-core), soit déployer NDN en périphérie du réseau et maintenir l'IP au cœur du réseau (NDN-edge). Avec une approche NDN-core, les applications IP n'ont pas besoin d'être modifiées, mais un déploiement mondial de NDN en tant que protocole réseau natif n'est actuellement pas réalisable, comme indiqué précédemment. A titre d'exemple exceptionnel, le projet POINT a dû travailler avec les FAI pour déployer un prototype du monde réel dans lequel une architecture ICN est utilisée au cœur du réseau. Le prototype introduit ensuite l'ICN dans le réseau central sans modifier le reste (c'est-à-dire la périphérie) de l'Internet.

Avec une approche NDN-edge, le réseau principal continue de fonctionner sur IP, tandis que les applications et les périphériques fonctionnent avec du NDN natif. Cette solution est facile à déployer et ne nécessite pas de modifications profondes de l'infrastructure. De plus, il fournit une intégration progressive du NDN. Dans les NDN-core et les NDN-edge, la coexistence d'IP et de NDN peut être obtenue en utilisant des nœuds périphériques tels que des passerelles pour la conversion entre les noms NDN et les informations de pile de protocole IP. Par exemple, le hICN de Cisco code les noms en tant que adresses IPv6 pour permettre le traitement des paquets hICN par des routeurs basés sur ICN et sur IP, et Zhang et. Al. ont proposé un système à double pile pour la coexistence de commutateurs NDN et de commutateurs IP dans des réseaux locaux. Au niveau de l'application, lorsque les piles NDN et IP doivent coexister, une translation entre les paquets NDN et IP est nécessaire. Celle-ci peut se faire au niveau TCP/UDP ou HTTP.

Pour éviter la traduction, il est possible d'adopter un déploiement hybride combinant des approches de NDN-edge avec superposition de NDN sur IP afin d'obtenir le maximum d'intégration possible de NDN (Figure 3). Cette combinaison est réaliste dans la mesure où les appareils IoT à ressources limitées implémentent uniquement le NDN et que les équipements puissants prendront en charge le NDN sur IP.

L'architecture NDN-802.15.4 que nous proposons est basée sur cette solution hybride. Lorsqu'elle est appliquée à l'IoT, l'intégration NDN-edge correspond au déploiement de NDN en périphérie de l'IoT. En d'autres termes, le NDN est utilisé là où le contenu est produit et consommé. D'une part, les périphériques IoT exécutent des applications NDN natives via une technologie de couche de liaison sans fil. D'autre part, l'utilisation de protocoles de transport basés sur NDN sur IP tels que UDP permet aux applications sur ordinateurs et smartphones de communiquer avec des périphériques IoT via NDN. En plus d'être tout à fait réalisable dans l'infrastructure Internet actuelle, cette approche tire parti de toutes les fonctionnalités NDN telles que le nom du contenu, la sécurité centrée sur les données et la mise en cache. De plus, la conception de l'IoT étant à ses débuts, en particulier pour l'IoT bas de gamme, cette approche constitue un point de départ raisonnable pour créer des périphériques compatibles NDN avec des applications natives NDN sans perte de temps. De plus, l'intégration de NDN à partir de la périphérie du réseau permet une intégration progressive et incrémentale. L'expérience des déploiements locaux conduira à une architecture NDN plus forte et diverses possibilités peuvent être envisagées à long terme.

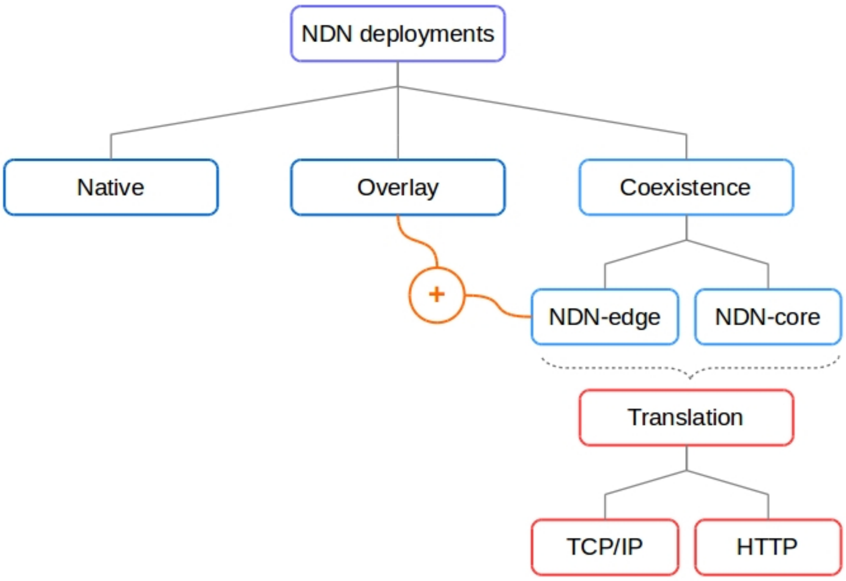


Figure 3: Intégrations possibles de NDN

Après avoir identifié la bonne approche d'intégration de NDN, la conception d'une architecture NDN-802.15.4 (Figure 4), comprenant des mécanismes spécifiques à NDN, est présentée et discutée. Grâce à l'architecture NDN-802.15.4, nous visons à former un duo NDN-IoT viable avec les objectifs suivants:

1. Étudiez comment l'intégration du paradigme NDN aux technologies sans fil à faible débit et basse consommation peut être conçue par rapport à l'intégration IP (e.g., 6LoWPAN).

2. Montrez qu'il est possible d'activer NDN dans les périphériques IoT en explorant

des mécanismes qui ne peuvent pas être envisagés avec IP.

3. Activez la vision du NDN dans les applications IoT réelles pour tirer parti des fonctionnalités de NDN.

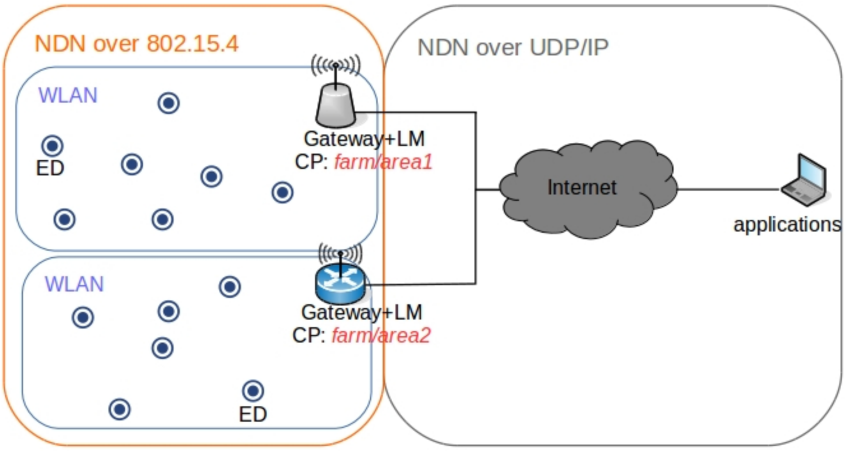


Figure 4: Architecture NDN-802.15.4

Cependant, la fourniture de mécanismes de prise en charge des paquets NDN via IEEE 802.15.4 ne constitue que la première étape de l'intégration NDN-802.15.4 à laquelle nous aspirons. Un aspect majeur du réseau doit être examiné, ce qui aura inévitablement un impact sur la façon dont les communications NDN sont gérées dans les environnements sans fil à faible débit. Cette préoccupation importante concerne le transfert de paquets dans les réseaux maillés IEEE 802.15.4. En effet, outre la réduction de la taille des paquets, une stratégie de transmission légère pour les réseaux maillés sans fil est nécessaire pour compléter l'architecture NDN-802.15.4.

Comme mentionné précédemment, l'utilisation de NDN directement sur la couche de couche liaison est un choix judicieux dans les réseaux sans fil. Cela soulève diverses questions sur la manière de concevoir des stratégies de transfert. Premièrement, nous devons déterminer si les adresses MAC unicast doivent être mappées sur des noms NDN ou si un transfert de diffusion est plus efficace. Deuxièmement, bien qu'une stratégie de transfert soit prise en charge au niveau de la couche réseau NDN, elle peut avoir un impact sur les composants de la couche liaison sous-jacents tels que l'algorithme CSMA. Pour répondre à ces questions, les bancs d'essai peuvent être utilisés comme outils d'évaluation réels, mais ils ne sont pas suffisants car ils sont limités par le nombre de nœuds, les possibilités de scénario et la précision des mesures. Pour permettre une évaluation rapide et à grande échelle des stratégies de transfert sans fil, un simulateur prêt à l'emploi pour le NDN sans fil dans l'IoT est nécessaire. Des modèles analytiques sont également nécessaires pour améliorer davantage la précision et comprendre l'impact des paramètres. Dans ce qui suit, nous décrivons comment nous prenons en charge l'évaluation des conceptions NDN-802.15.4 à l'aide de trois outils qui constituent notre environnement d'évaluation: le banc de test, l'outil de simulation et le modèle mathématique.

Outils d'évaluation

Nous présentons dans cette partie de la thèse les outils que nous avons développés et utilisés pour nos propositions. Comme vu précédemment, les communications sans fil via IEEE 802.15.4 est une fonctionnalité importante à prendre en charge dans notre architecture NDN-802.15.4. Cependant, il convient de noter que les communications sans fil NDN, en particulier dans les réseaux contraints, est légèrement différent de la communication NDN habituellement utilisée dans les réseaux câblés. La raison principale est que, en utilisant une radio sans fil, les nœuds n'ont pas la possibilité de choisir entre différentes interfaces pour transmettre les paquets; tous les paquets sont transmis via la même interface réseau. De plus, le transfert de paquets doit traiter la redondance si la diffusion est utilisée. De toute évidence, le principal défi d'une stratégie de communication sans fil consiste à réduire la consommation des ressources et les retransmissions inutiles, tout en maintenant une disponibilité et une diffusion efficaces des données. Pour pouvoir concevoir et tester des stratégies de transfert sans fil, nous avons mis en place un environnement d'évaluation comprenant trois outils complémentaires: un réseau de test, un outil de simulation basé sur le simulateur OMNeT ++ et un modèle mathématique.

Premièrement, le réseau de test reflète l'architecture NDN-802.15.4 présentée plus haut, avec quelques nœuds fixes. Il permet de mesurer des opérations à petite échelle telles que le delais de communication, l'utilisation de la mémoire et le délai de compression des paquets. Deuxièmement, pour évaluer des scénarios plus complexes, nous avons développé l'outil de simulation NDN-OMNeT. NDN-OMNeT reflète également les communications passerelle-à-appareil et appareil-appareil de notre architecture, mais permet une évaluation à plus grande échelle, avec des noeuds mobiles ou non, avec différentes topologies de réseau et différentes solutions de couche liaison telles que 802.11 et 802.15.4. Le principal objectif de cet outil est de comparer et d'évaluer rapidement les stratégies de communication sans fil. Troisièmement, nous modélisons la stratégie de transmission de base de NDN dans les réseaux sans fil proposée dans la littérature et décrite plus loin dans cette thèse. Le modèle mathématique est utilisé pour étudier l'efficacité de NDN dans une topologie de réseau sans fil simple sous différentes popularités de contenu.

Les strategies d'acheminement NDN-802.15.4 proposées

Pour déployer NDN dans des périphériques IoT, l'une des principales fonctionnalités à prendre en charge est la communication NDN dans un réseau maillé sans fil à faible débit, tel que les réseaux IEEE 802.15.4. Cette partie de la thèse présente nos solutions de communication dans les réseaux maillés sans fil sous contrainte en général, et IEEE 802.15.4 en particulier. Les stratégies de communication sans fil NDN reposent généralement sur un mécanisme de diffusion et d'apprentissage. Cette approche utilise une phase dans laquelle les Interest sont diffusés jusqu'à ce que le contenu soit trouvé, puis les Interest suivantes sont transmis avec plus de précision en fonction des informations apprises. Par conséquent, l'utilisation de la diffusion est nécessaire dans les réseaux sans fil NDN. De plus, les résultats de l'évaluation rapportés plus loin dans cette thèse suggèrent que la diffusion est le modèle le mieux adapté pour créer une stratégie d'acheminement efficace, concernant la mobilité des hôtes et la disponibilité du contenu.

Les stratégies de communication allégée pour les environnements NDN sans fil sont rares dans la littérature. De plus, à notre connaissance, aucune solution n'a étudié éxplicitement les communications NDN dans les réseaux maillés IEEE 802.15.4. Par conséquent, nous commençons par étudier d'abord les principales approches d'acheminement pour NDN figurant dans la littérature connexe. Deuxièmement, les stratégies basées sur la diffusion étant simples et efficaces pour la diffusion de données et conformes au modèle de communication natif ICN/NDN, nous devons déterminer comment utiliser la diffusion tout en réduisant les besoins en temps système, en mémoire et en traitement. Pour cela, nous étudions l'impact du modèle de diffusion dans les réseaux sans fil restreints en termes de redondance des données, de nombre de paquets transmis, de disponibilité des données et de précision des décisions. Notre étude considère deux scénarios: (i) un réseau sans fil simple avec une topologie en arborescence binaire et des nœuds fixes et (ii) un réseau relativement complexe avec une topologie en grille et des nœuds mobiles. Cette étude aboutit à un ensemble de directives de conception pour une stratégie de communication sans fil légère basée sur la diffusion.

Enfin, avec toutes les considérations nécessaires, nous proposons deux solutions conçues à deux niveaux différents. Au niveau de la couche liaison, nous proposons une adaptation de l'algorithme CSMA utilisé dans la spécification IEEE 802.15.4 pour améliorer les communications NDN dans une topologie simple. Au niveau de la couche réseau, nous proposons une stratégie légère d'acheminement des paquets basée sur le renforcement, destiné à prendre en charge des scénarios de réseau complexes. L'idée dans les deux approches est de rendre la diffusion aussi précise que le monodiffusion, en termes de décisions d'acheminement et de nombre de trames transmises. Par conséquent, nous nous concentrons sur la conception de techniques de compromis légères capables de maintenir des performances satisfaisantes dans différents scénarios de communication et configurations de réseau.

Contributions de la thèse

Les contributions impliquées dans cette thèse peuvent être présentées dans l'ordre dans lequel elles ont été conduites comme suit.

NDN sur IEEE 802.15.4

Ces dernières années, plusieurs projets de recherche ont démontré la capacité de NDN à prendre en charge les applications IoT émergentes telles que la domotique, les villes intelligentes et les applications d'agriculture intelligente. Motivé par cela, notre premier travail consistait à intégrer NDN dans la technologie IEEE 802.15.4 afin de mieux prendre en charge les applications IoT qui nécessitent en outre une faible consommation d'énergie.

À cette fin, nous avons conçu un schéma de communication NDN sur IEEE 802.15.4 basé sur des modules radio ZigBee. Ce travail peut être considéré comme notre premier pas vers une utilisation réaliste de NDN dans une architecture IoT.

Architecture NDN-IoT réaliste

La technologie sans fil IEEE 802.15.4 permet aux périphériques contraints de communiquer avec un débit de données, une taille de charge utile et une plage de distance satisfaisants, le tout avec une consommation d'énergie réduite. Pour fournir aux périphériques IoT une identité Internet globale, 6LoWPAN définit l'adaptation IPv6 pour la communication sur IEEE 802.15.4. Motivés par le fait que 6LoWAP nécessite toujours des protocoles supplémentaires pour prendre en charge d'autres exigences IoT, nous avons décidé de concevoir une architecture IoT réaliste basée sur NDN et notre intégration de NDN avec IEEE 802.15.4. Les problèmes d'intégration sont discutés avec certaines solutions et la conception proposée a été mise en œuvre dans un scénario réel d'agriculture intelligente. Des résultats de simulation sur la consommation d'énergie et la surcharge du réseau ont été rapportés par rapport aux solutions IP telles que AODV.

Simulateur NDN-OMNeT

Les solutions NDN-IoT nécessitent également une évaluation précise au niveau du réseau et du système. Ce travail présente NDN-OMNeT, notre module NDN pour le simulateur OMNeT++. Conçu pour les appareils et les passerelles de l'IoT, cet outil est capable de simuler des scénarios NDN aux limites du réseau et du système. L'implémentation est présentée et utilisée pour étudier un aspect typique de l'intégration NDN dans des périphériques IoT.

Une stratégie de communication pour NDN dans les réseaux sans fil IoT

Pour compléter l'architecture NDN-IoT basée sur IEEE 802.15.4, une communication à sauts multiples entre périphériques doit être prise en charge. Par conséquent, nous présentons une stratégie de transmission allégée pour NDN sur IEEE 802.15.4. La stratégie utilise des communications de diffusion et est conçue pour réduire au maximum la charge du réseau, tout en maintenant des performances satisfaisantes dans différents scénarios d'application IoT. Pour transférer des contenus nommés sans adresses de nœud, la stratégie repose sur une technique d'apprentissage par renforcement qui fournit des décisions de transfert précises, basées sur la diffusion, avec un temps système réduit. Cette solution, appelée stratégie d'acheminement légère basée sur le renforcement (R-LF), fonctionne au niveau du réseau (L3) avec la couche de liaison IEEE 802.15.4 existante.

Modélisation et amélioration des communications NDN sur IEEE 802.15.4

Pour concevoir un solide déploiement NDN-IoT, des adaptations des technologies de l'IoT actuelles seront nécessaires. Des exemples de telles technologies sont les normes sans fil à faible consommation telles que IEEE 802.15.4. Dans cette contribution, nous voulons gérer les communication NDN au niveau de la couche liaison (L2). À cette fin, nous explorons le transfert NDN via IEEE 802.15.4 en deux étapes. Premièrement, nous modélisons une stratégie de transfert sans fil NDN simple, basée sur la diffusion dans des réseaux simples. Le modèle proposé prend en compte la popularité du contenu et estime le nombre moyen de trames transmises par demande et le temps moyen d'aller-retour. Deuxièmement, sur la base du modèle et des observations expérimentales du transfert basé sur la diffusion au niveau du réseau (L3), nous constatons que le niveau de couche liaison (L2) peut être adapté pour réduire les effets de diffusion en termes de redondance des paquets, de temps d'aller-retour et consommation d'énergie. Par conséquent, nous élaborons une adaptation de l'algorithme CSMA (Carrier-Sense Multiple Access) de IEEE 802.15.4 pour améliorer la stratégie de transfert modélisée. Les résultats préliminaires obtenus montrent qu'une adaptation de l'algorithme CSMA devrait être envisagée pour améliorer certaines technologies actuellement prises en charge par NDN.

Conclusion

Le travail effectué dans cette thèse a commencé avec un objectif principal qui était d'activer NDN dans des environnements IoT périphériques. Pour y parvenir, nous avions besoin d'une combinaison d'outils complémentaires nous permettant d'atteindre des objectifs partiels et qui font maintenant partie du travail mondial. De manière générale, nous avons cherché à tirer le meilleur parti possible de NDN pour l'appliquer à l'IoT. À cette fin, une architecture réaliste NDN-802.15.4 a été conçue et construite en considérant la technologie sans fil IEEE 802.15.4. Après avoir identifié l'intégration de NDN dans l'IoT périphérique comme étant l'approche la plus réaliste, les principaux problèmes d'intégration ont été discutés et certaines propositions ont été avancées en revoyant certaines solutions IP pour l'IoT, telles que 6LoWPAN. Les mécanismes proposés témoignent de la flexibilité de NDN pour la prise en charge de technologies sans fil à faible débit telles que IEEE 802.15.4. L'architecture NDN-802.15.4 obtenue a pour objectif de former un duo nouveau et fort du NDN-IoT. Plus important encore, le transfert NDN léger dans les réseaux sans fil avec diffusion a été étudié. Les résultats obtenus montrent que nous pouvons utiliser les communications par diffusion pour prendre des décisions de transfert relativement précises avec des couts réduits et des performances satisfaisantes. En bref, nous avons pu préserver les avantages de la radiodiffusion tout en réduisant ses inconvénients.

Lors de la recherche sur le transfert sans fil avec NDN, un environnement comprenant des outils d'évaluation différents mais complémentaires a été mis au point. Ces outils peuvent fournir des mesures du monde réel, des résultats de simulation et une analyse mathématique du transfert NDN dans des réseaux sans fil. En outre, cet environnement d'évaluation peut être à nouveau exploité à d'autres fins connexes.

Globalement, la principale limitation que nous pouvons identifier dans ce travail est l'absence de comparaison directe des performances entre NDN et IP. Bien que cela puisse être utile, cela peut s'expliquer par plusieurs raisons. Premièrement, nous nous appuyons beaucoup sur les discussions sur les limitations IP et les fonctionnalités natives NDN pour montrer la supériorité du NDN, incontestable dans de nombreux aspects tels que la sécurité, la mise en cache native et la simplicité. Deuxièmement, les comparaisons directes entre NDN et IP ne sont pas concluantes en raison de la différence de paradigme évoquée au début de la thèse. Troisièmement, les implémentations NDN, y compris celle présentée dans ce document, en sont encore au stade expérimental, tandis que les solutions IP sont matures et généralement bien optimisés.

Enfin, la dernière pensée pour conclure sur les contributions présentées dans ce manuscrit est la suivante. Ils ne représentent pas la finalisation d'un travail, mais plutôt un point de départ qui ouvre la voie à des étapes plus passionnantes. Un exemple est la conception d'un produit ou d'une plate-forme IoT, basée sur NDN, comme décrit plus loin.

Nous fournissons un scénario simple dans lequel le travail présenté dans cette thèse peut être utilisé pour concevoir un PoC pour un produit IoT avec NDN. Comme indiqué dans l'introduction, le développement d'une PoC peut être la clé de la création et de la commercialisation d'un produit IoT. Premièrement, il faut trouver une idée originale sur le service ou l'application à concevoir et définir un cas d'utilisation précis. Cela peut être, par exemple, une serre connectée. Une serre connectée est une installation agricole qui utilise des capteurs pour capturer des données sur la croissance des plantes, l'irrigation, l'utilisation de la lutte antiparasitaire et l'éclairage, et les envoyer à un serveur local ou sur le Cloud. En utilisant les données collectées, une application Web permet aux agriculteurs de configurer les paramètres du système et de prendre des décisions, tandis qu'une application mobile génère des alertes et des rapports sur les performances de la serre. La plupart des solutions de serre connectées peuvent utiliser des dispositifs simples prenant en charge plusieurs types de capteurs, utiliser une connectivité sans fil à faible débit, consommer peu d'énergie et pouvant être insérées dans le sol ou fixées à des tiges. La communication entre les périphériques IoT et Internet peut être réalisée via un réseau maillé IEEE 802.15.4 standard, à travers lequel les nœuds échangent des données et transmettent des messages envoyés par des capteurs jusqu'à ce qu'ils atteignent la passerelle. Plusieurs passerelles peuvent être installées dans la serre, permettant aux capteurs de se connecter à Internet et de transmettre des données au serveur. Le coût d'une solution IoT personnalisée pour la serre est estimé entre 100 000 et 150 000 dollars. La somme couvre le développement de systèmes embarqués, une application Web et un client mobile pour les notifications d'alerte, ainsi que des services de conseil concernant le choix des composants matériels. La deuxième étape consiste à créer un PoC. Selon la description de projet ci-dessus, notre architecture NDN-802.15.4 inclut toutes les fonctionnalités nécessaires à la création d'un prototype opérationnel, y compris la sécurité. Avec le prototype opérationnel, les mesures et les résultats de la simulation, nous pensons que des démonstrations étonnantes peuvent être faites et que les investisseurs peuvent gagner un intérêt particulier.

Au moment de la rédaction de ce manuscrit, les travaux sont toujours en cours. Premièrement, le schéma ND-CSMA proposé a été conçu pour une topologie d'arbre binaire. Ainsi, il n'est pas prévu de travailler sur des réseaux complexes tels que des topologies de grille avec mobilité. Les résultats obtenus avec la stratégie R-LF suggèrent qu'une adaptation plus sophistiquée de CSMA devrait être envisagée en exploitant les informations R-LF telles que le coût fournis par apprentissage. Autrement dit, une meilleure stratégie d'acheminement peut être obtenue en combinant les atouts de R-LF avec l'efficacité de ND-CSMA afin de réduire le temps d'aller-retour. Cette idée est actuellement à l'étude. Le deuxième travail en cours concerne la mise en œuvre du banc de test, que nous visons à améliorer avec une pile NDN plus efficace comprenant des PIT et des FIB légers pour réduire l'utilisation de la mémoire, basée par exemple sur des filtres de bloom. Enfin, le dernier travail réalisé dans cette thèse était le modèle analytique pour le transfert sans fil avec NDN. Cependant, il est actuellement simple et limité à la stratégie de transfert de base NDN dans une topologie d'arbre binaire. L'étape suivante consiste à prendre en charge des topologies plus complexes dans le modèle. Nous visons à modéliser une topologie de réseau similaire au DAG utilisé dans RPL et à effectuer des comparaisons analytiques supplémentaires entre RPL et NDN dans un réseau maillé sans fil. Dans l'intervalle, l'outil de simulation NDN-OMNeT propose de nouvelles fonctionnalités afin d'offrir aux utilisateurs davantage de possibilités.

General Introduction

Context of the Thesis

The Internet of Things (IoT) uses the interconnection of billions of small computing devices, called “Things”, to provide access to services and information all over the world. This has been made possible by the democratization of smartphones and laptops, and more importantly by the affordability of resource-constrained data acquisition devices and corresponding wireless communication technologies. As an example, sensor costs dropped by almost 200% between 2004 and 2016 [Andrei Klubnikin a]. Consequently, four times more devices than people are expected to be connected to the Internet by the end of this year [IoT online store]. The market is obviously impacted by this evolution. IoT is expected to grow from US$656 billion in 2014 to US$1.7 trillion in 2020. The economic impact of the IoT could be between US$3.9 trillion and US$11.1 trillion per year by 2025.

In practice, IoT devices are battery powered, have tens to thousands of Megahertz CPU and tens to thousands of Kilobytes memory. In an IoT system, these constrained devices communicate with one another or with user applications over the Internet. This communication is achieved through the TCP/IP protocol suite and the wireless interconnection is achieved with low-power and lossy wireless technologies. However, the IP protocol suite was designed decades ago for a completely different purpose as shown further, and IoT features now highlight the limitations of IP. For example, most of security protocols for IP are based on communication channels while the data itself needs to be secured. Moreover, IoT systems need efficient support for resource naming and discovery, which is not easy to deploy with IP in constrained infrastructures.

That being said, IP can still support IoT systems through adaptations and middleware.

This is why we hear about CoAP, 6LoWPAN, RPL, REST and other solutions. However, the significant efforts expended in current IP solutions is just to make IoT devices support the existing IP protocol suite, whereas many other new features have to be included in devices. Thus, if the support of communication in devices could be simplified and made robust, it would be a fundamental enabler for a global IoT ecosystem. Moreover, simplifying communication solutions for IoT applications would significantly reduce development cost.

While adapting IP for the IoT might be seen as cutting corners, alternative architectures based on the Information Centric Networking (ICN) paradigm promise to natively satisfy emerging Internet applications. One of these architectures is Named Data Networking (NDN). The NDN project was funded by the National Science Foundation (NSF) under the Future Internet Architecture (FIA) program. The main entity in NDN is the content. Networking operations are performed on names, and hosts (without logical addresses) request named-content directly from the network. Native features come along with this principle, such as communication without establishing end-to-end connections and name-to-address resolution. Moreover, no consumer-provider path or session needs to be maintained, which provides a native support of connection disruption resulting from mobility.

A natural reflex when hearing about NDN, is to ask how we can take advantage of its features without waiting decades for the future Internet architecture. However, fundamental modifications must be made to the current IP-based networking equipment, protocols and applications since the NDN paradigm operates on content names rather than host addresses. Furthermore, as long as IP solutions work for current applications, convincing IP-enthusiasts and industrial players about the benefits of NDN will remain difficult. Fortunately, in recent years, many studies have investigated the suitability of NDN for the IoT, making NDN increasingly powerful. With all this motivating work, real deployments of NDN can now be envisioned. Although NDN is not ready for worldwide IoT deployments, real-world designs will enrich NDN experiments and help to figure out what is needed to make NDN a reality.

However, despite the increasing interest that NDN is gaining, the success of the NDN project (as well as other ICN projects) is not guaranteed. Indeed, NDN is currently an academic project handled by students and universities, and an industry support has not emerged yet. Consequently, a great attention should be given when using NDN for the IoT, particularly in the case of a thesis. For example, if the NDN project fails, the contributions of a thesis may loose their importance if they are too much focused on a pure and isolated NDN scenario. For these reasons, we believe that enabling ICN principles (through NDN) in current IoT designs is a more suitable approach than considering an isolated NDN solution for the IoT.

We believe that integrating NDN in low-end IoT devices over a low-power wireless technology is a suitable approach to use NDN in the IoT, given the importance of constrained-devices in IoT systems. Moreover, low-end IoT is still in a development stage even with IP, which provides an opportunity to make NDN an important part of IoT solutions in a short period of time. To take this opportunity, we base our work on a realistic NDN deployment in the IoT. The concept of realistic in this context includes many aspects. First, it aims to enable NDN in the current Internet infrastructure. That is, scenarios that can not be deployed now are not considered. Second, realistic means using NDN to design solutions that work on current IoT equipment such as prototyping boards. Third, the objective of a realistic design is to provide a viable NDN solution for the IoT that must be easy-to-use, low-cost, simple and lightweight. We can think of the realistic approach envisioned here as the development path followed by two or three startupper-friends aiming to launch a revolutionary IT product. According to Gartner, 50% of all IoT solutions are developed by start-ups or small companies that often do not possess the financial resources to design and produce a market-ready IoT product from scratch [Andrei Klubnikin a]. Therefore, to succeed in creating an IoT application, startuppers often use a Proof-of-Concept (PoC) to show that their solution can find commercial success [Hemendra Singh] . This is commonly achieved using systems on a chip (SoCs), micro-controllers and single-board computers like Arduino, BeagleBoard and Raspberry Pi. Moreover, having a working prototype significantly increases a company's chances of getting funded.

Objectives and Contributions

At a global level, this thesis proposes a vision about how ICN/NDN can be considered in the IoT. This first step is a chance to explore some popular IP-based solutions for the IoT, and figure out how NDN can be brought into IoT systems. After that, the NDN forwarding in constrained wireless networks is identified as one of the issues to address to make NDN a reality in IoT solutions. This second step allows us to explore in details the wireless forwarding in constrained networks, which is an important aspect in both IoT and NDN.

Hence, our objectives through the work reported in this manuscript can be summarized in two aspects. The first objective is to show that NDN is suitable to support IoT networking. To achieve that, a detailed discussion on IP limits and NDN features will be presented first, followed by the design and deployment of a realistic NDN architecture for the IoT using IEEE 802.15.4 (NDN-802.15.4) that shows the simplicity of NDN. Moreover, a simulation framework for NDN-802.15.4 and a model for NDN wireless forwarding are proposed to quantify and analyse the advantages of NDN. The second objective is the design of two solutions for lightweight forwarding in constrained wireless networks. These two solutions consider the IEEE 802.15.4 technology and use only broadcast communications. The utility of broadcast in NDN networks will also be studied. In short, the first solution is based on a reinforcement learning scheme and operates at the network layer, while the second is based on priority-based medium access and operates at the link-layer.

The contributions involved in this thesis can be presented in the order in which they were conducted as follows.

NDN over IEEE 802.15.4. In recent years, several research projects have demonstrated the ability of NDN to support emerging IoT applications like home automation, smart cities and smart farming applications. Motivated by this, our first work was to integrate NDN with IEEE 802.15.4 to give NDN a better support for IoT applications that are known to require wireless sensing/actuating abilities, mobility support and low power consumption. To this end, we designed an NDN-over-802.15.4 communication scheme based on ZigBee radio modules. This work can be considered as the first step towards a realistic use of NDN in low-end IoT architectures.

Realistic NDN-802.15.4 architecture. The IEEE 802.15.4 wireless technology allows constrained devices to communicate with a satisfactory data rate, payload size and distance range, all with reduced energy consumption. To provide IoT devices with a global Internet identity, 6LoWPAN defines the IPv6 adaptation to communicate over IEEE 802.15.4. Motivated by the fact that 6LoWAP still needs additional protocols to support other IoT requirements, we decided to design a realistic IoT architecture based on NDN and our NDN-802.15.4 integration. Integration challenges are discussed with some solutions and the proposed design has been implemented in a real-world smart agriculture scenario. Simulation results have been reported on energy consumption and network overhead in comparison to IP-based solutions such as AODV.

NDN-OMNeT simulator. NDN solutions for IoT need accurate evaluation at both the network and system levels. This work introduces NDN-OMNeT, our NDN framework for the OMNeT++ simulator. Designed for low-end devices and gateways of the IoT, the framework is capable of simulating NDN scenarios at the boundary of the network and the system. The framework implementation is presented and used to study a typical aspect of NDN integration in IoT devices.

A lightweight forwarding strategy for NDN in low-end IoT. To complete the NDN-802.15.4 architecture, multi-hop communication between devices must be supported. Therefore, we present a lightweight forwarding strategy for NDN over IEEE 802.15.4. The forwarding strategy uses broadcast communications and it is designed to reduce network overhead to the bare minimum, while keeping satisfactory performance in different IoT application scenarios. To forward named content items without node addresses, the strategy is based on a reinforcement-learning technique that provides accurate broadcast-based forwarding decisions with a reduced overhead. This solution, called Reinforcement-based Lightweight Forwarding strategy, operates at the network level (L3) with legacy IEEE 802.15.4 link-layer.

Modeling and Improving NDN forwarding over IEEE 802.15.4. To design a strong NDN-802.15.4 deployment, adaptations of current IoT technologies will be required. Examples of such technologies are low-power wireless standards such as IEEE 802.15.4.

In this contribution we want to handle NDN forwarding at the link layer (L2). For that purpose, we explore NDN forwarding over IEEE 802.15.4 according to two steps. First, we model a simple broadcast-based NDN wireless forwarding strategy over constrained networks. The proposed model considers content popularity and estimates the average number of frames transmitted per request and the mean round-trip time. Second, based on the model and experimental observations of the broadcast-based forwarding at the network level (L3), we find that the link-layer level (L2) can be adapted to reduce broadcast effects in terms of packet redundancy, round-trip time and energy consumption. Hence, we develop an adaptation of the Carrier-Sense Multiple Access (CSMA) algorithm of the IEEE 802.15.4 to enhance the modeled forwarding strategy. Preliminary results show that an adaptation of the CSMA algorithm should be considered to improve some of the current technologies support of NDN.

Organisation of the Manuscript

This manuscript includes five chapters each containing five sections, except the last chapter which contains seven. Chapter 5 is the heart of this document as it gathers all the concepts introduced in the four first chapters, to tackle the question of lightweight forwarding in constrained networks with NDN. It is thus the most technical chapter and includes an extensive evaluation of the presented work. Before getting to that part, in Chapter 1, we discuss existing IP solutions for the IoT and their limits, which ends by introducing the ICN concept with NDN as one of the most promising alternatives to IP. Chapter 2 introduces NDN principles, communication process and some NDN-for-IoT related work, which we consider as inspiring and necessary for this document. In Chapter 3, we study the possibilities of deploying NDN in realistic solutions, then we choose the most realistic approach and present our NDN-802.15.4 architecture and its mechanisms. We finish this chapter by identifying NDN forwarding in wireless mesh networks as the main feature we need to support. Chapter 4 presents the three tools we develop in our environment to investigate the forwarding problem, namely: the NDN-802.15.4 testbed, the simulation framework NDN-OMNET, and the analytical model. Finally, in Chapter 5 we present our propositions for the wireless forwarding problem at two different levels, namely: a general network level solution and a link-layer solution with IEEE 802.15.4. Our contributions in this thesis start from Chapter 3.

Although the chapters are ordered and related, they are organized such that they may be read separately according to the reader's background. For example, a reader already familiar with IP limitations can start directly at Chapter 2; a reader who is familiar with NDN and its utility for IoT can start reading at Chapter 3; reader who is only interested in NDN evaluation environments can read Chapter 4, and a reader who is focusing on wireless forwarding strategies and evaluation can go directly to Chapter 5.

Chapter 1

IP vs. ICN: The IoT Challenge

1.1 Introduction

In less than 40 years, Internet Protocol (IP) networks have created the Internet with today's world of content, despite that fact that IP networking was not designed for it. This is a meaningful thought to summarize the evolution of the Internet and its current situation with the Internet of Things (IoT) and emerging applications. In this chapter, we shall explain this idea and show why it is true. We consider that the IoT is a good point of view to understand IP limits as it brings requirements that reflect the evolution of the technology as well as society itself. Therefore, we investigate how the IP architecture supports IoT systems, and we present the solutions and their shortcomings. Finally we introduce the ICN/NDN networking approach as a solution that gathers the essential features required to efficiently meet IoT requirements.

1.2 The Internet of Things

1.2.1 Description

Massive technological innovations towards electronic miniaturization, associated with affordable System on Modules (SOMs) and single board computers have fostered the emergence of billions of devices, through which people and "Things" are becoming connected over the Internet. This new trend of cyber-physical Internet (see Figure 1.1) is commonly known as the Internet of Things (IoT) [Atzori et al. 2010].

The IoT can be seen as the latest (and arguably the first) evolution of the Internet, and it is gaining an essential role in providing access to services and information all over the world. This has became possible through the interconnection of billions of small computing devices (called Things) that allow the physical world to be monitored and controlled. A global view of the IoT includes devices of various types, that can be resource-constrained, powerful and virtualized objects.

The figures relating to the IoT are astounding. In 2008, there were already more objects connected to the Internet than people. In 2016, the number of IoT connections grew by 45% to 410 million. By 2020, 90% of cars will be connected to the Internet, whereas only 10% were connected in 2012. In 2020, the total number of connected things will reach 50 billion for 4 billion connected people, with over 25 million apps.

Typically, an IoT system consists of a large number of wireless devices, deployed within an infrastructure (e.g., buildings, cities), and reachable over the Internet. Worldwide networking in today's IoT applications is widely supported by the TCP/IP protocol suite. The low-end IoT (i.e., the "Things" side) is built with battery powered devices, that have limited resources (i.e., 10s to 100s of MHz CPU and 10s to 100s of KB memory) and which are often mobile. The interconnection of these resource-constrained devices is achieved with low-power and lossy wireless networks (LLNs), which allow communication with a satisfactory data rate (10s to 100s kbps), payload size (10s to 100s bytes) and distance range (10s to 100s meters), and some low-power devices possess years of battery lifetime. LLN technologies such as IEEE 802.15.4, BLE and low-power Wifi provide the best compromise for resource-constrained devices.

Beside LLNs, lpWANs are a new form of wireless technologies for IoT applications. They were designed with the aim of offering equipment a means of communicating over very long distances (kilometers) with very low energy consumption. LPWAN solutions such as LoRa [LoRa Alliance] and Sigfox [Sigfox] are typically used for IoT applications in which equipment (e.g., sensors) generate small volumes of data and a reduced number of packets per hour.

Figure 1.2 depicts the typical components of the IoT architecture and their relationships.

Typically, IoT applications fetch data from devices, often use it for analytic and/or to make decisions using Artificial Intelligence (AI) algorithms, and may control other devices such as actuators. Table 1.1 summarizes IoT phases and their corresponding technologies. All these operations are commonly performed according to the request-response communication model.

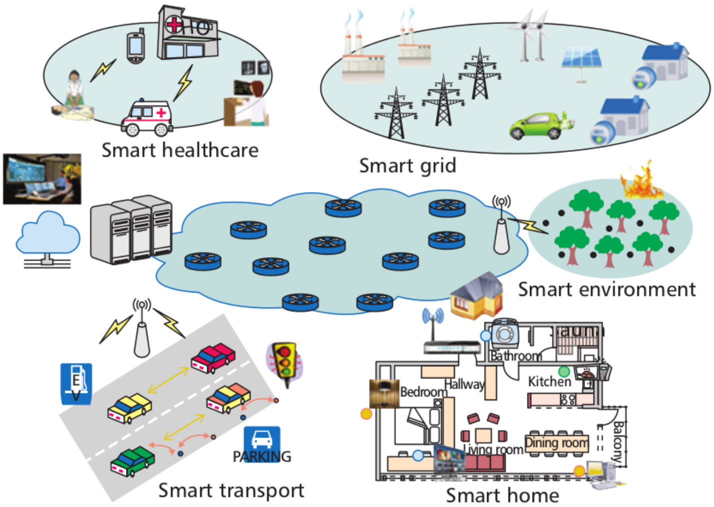


Figure 1.1: A global view of the Internet of Things [Amadeo et al. 2016]

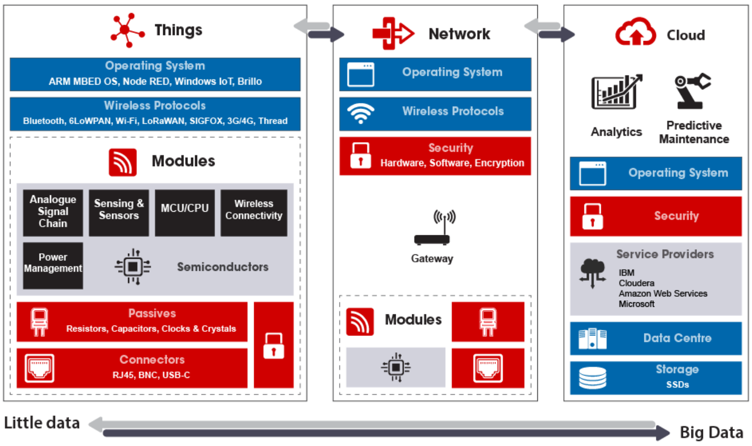


Figure 1.2: IoT architecture components [RS Components Ltd.]

Table 1.1: IoT phases and corresponding technologies

IoT phase	**Solutions**
1. Acquisition and sensing	RFID, WSN, Bluetooth, NFC
2. Data transmission	Ethernet, WiFi, VANET/MANET; 4G/5G
3. Data processing, analysis and management	Cloud computing, Big Data, Machine learning
4. Decision and action	Application, actuators

1.2.2 Examples

1- Smart homes are one of the most popular IoT applications [IoT online store]. In a smart home, IoT devices such as sensors and actuators are integrated to monitor and control homes using a smartphone or a personal computer. Various products, ranging from light switches to smart TVs are proposed to customers.

2- By 2050, the agriculture Internet of Things will boost food production by 70% [Andrei Klubnikin b]. IoT systems are deployed as precision agriculture solutions to help improve productivity by making better and faster decisions. Farmers can deploy IoT devices to collect data such as temperature/humidity in crops, or motion on livestock. The collected data feeds machine learning algorithms that can predict diseases or events such as frost. Decisions can then be taken accordingly.

3- We consider a specific precision agriculture scenario to illustrate the concepts presented in this document. A cow monitoring system that uses sensors (e.g. movement, temperature, microphone.) to monitor health, fertility and location of each cow individually. The collected data may be analyzed to detect whether a cow is sick, or to forecast cows' activities such as heat periods to make breeding decisions more accurate. The devices are generally installed in a collar on every cow. As the cows are mobile, data may be published from various places: inside, in the field, or in the milking parlor. The collected data can be visualized on the farmer's smartphone or stored/analyzed on the farm's main computer.

Figure 1.3 depicts the main application domains currently known in the IoT [Atzori et al. 2010].

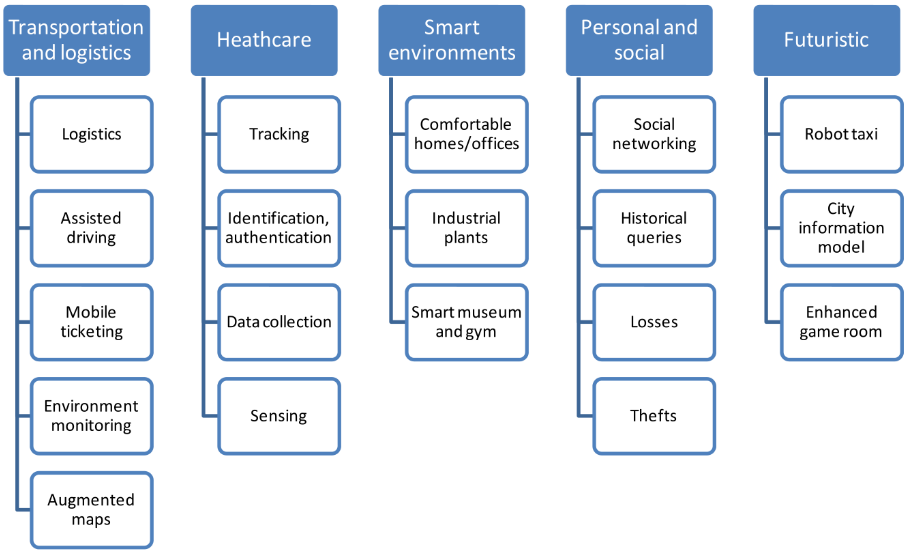


Figure 1.3: IoT applications domains and main scenarios

1.3 IoT puts IP to the test: Challenges and Shortcomings

1.3.1 Brief Story of IP

To understand how IoT features are currently supported by IP, a short recap of the creation and the vision of IP networking is useful. The revolutionary feature of IP networking is that it made it possible to concatenate networks of different kinds by abstracting their lower-level characteristics (i.e. Layer 2 technologies). That allowed networks to be uniformly connected regardless of their respective protocols, to form bigger networks, which were further concatenated to make the Internet. At the beginning, only a few computers were connected. They were very expensive as indeed were resources required to make them work, such as tape drives and printers. As resources were expensive and devices rare, the main usage of networking was to share these precious resources. For example, with networking, one printer can be used by two or more computers in a lab; that makes the machines busy while making the most of resources to improve productivity. For this reason (i.e. sharing resources between computers), networking protocols were developed on the basis of identifying hosts, ranging from computers to printers and so on. These protocols focused on having a point-to-point conversation between two hosts over the network. Logical addresses were then used to identify hosts. It is worth noting that data was not carried on computers at that time, but was kept on storage devices such as tapes and hard disk drives.

Naturally, this produced the Internet with an architecture that considers the computer/host as the main entity. Its communication model was an extension of the telephone model to support exchanging host-dependent packets. Therefore, IP's central feature was to deliver packets between a source host and a destination host. When exchanging data was needed, additional layers and protocols were developed, above this network layer, to support flow control, end-to-end reliability and user applications, creating today's Internet protocol stack (i.e. TCP/IP).

In the meantime, revolutionary Internet applications (e.g. the Web) shifted the focus from identifying hosts (IP) to identifying resources (URI) leading to the creation of the Domain Name System (DNS) service. Consequently, two namespaces following two different models are currently involved in the Internet protocol stack: IP addresses and resource names.

Figure 1.4 depicts a timeline of the Internet evolution, and some interesting facts.

A recent evolution of Internet, the IoT, puts IP networking to the test, and definitely highlights the mismatch between IP's host-centric paradigm and application needs, as is explained below.

1.3.2 IoT over IP

The request-response communication model used in the IoT to get data from sensors and send commands to actuators is commonly achieved using the REST (REpresentational State Transfer) architecture [Mueller 2013] just like in Web services. To enable the REST architecture in IoT applications, the IETF CoRE WG defined the Constrained Application Protocol (CoAP) standard [Shelby et al. 2014]. CoAP can be seen as a lighter version of the HTTP protocol. It is a data transfer protocol that provides a REST communication over UDP for constrained environments. CoAP can support caching through some adaptations.

CoAP and its variants make it possible to provide three communication primitives

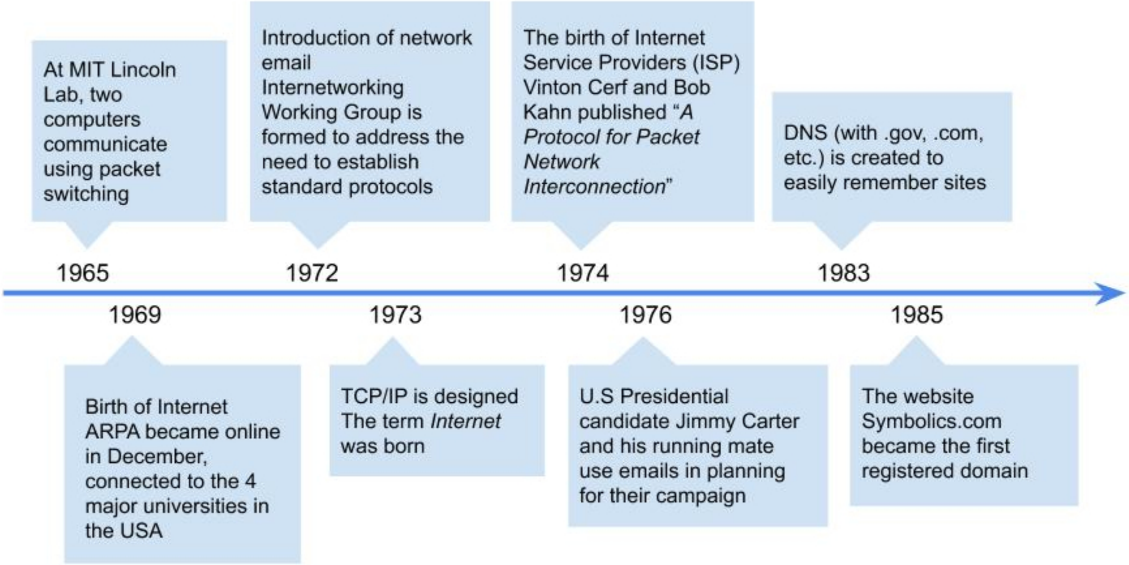

At MIT Lincoln Lab, two computers communicate using packet switching
1965
Introduction of network email Internetworking Working Group is formed to address the need to establish standard protocols
1972
The birth of Internet Service Providers (ISP) Vinton Cerf and Bob Kahn published "*A Protocol for Packet Network Interconnection*"
1974
DNS (with .gov, .com, etc.) is created to easily remember sites
1983
1969
Birth of Internet ARPA became online in December, connected to the 4 major universities in the USA
1973
TCP/IP is designed The term *Internet* was born
1976
U.S Presidential candidate Jimmy Carter and his running mate use emails in planning for their campaign
1985
The website Symbolics.com became the first registered domain

Figure 1.4: Internet evolution timeline

Table 1.2: COAP methods and usage example

Action example	**Description**
GET */temp*	Read a temperature from a sensor
POST */door*	Open or close a door
PUT */config*	Configure a sensor or actuator
DELETE */sensor/42*	Tell a controller to remove a sensor from the list that it uses for its actions

required in common IoT scenarios. First, pull is the common request response communication pattern (e.g., HTTP). Second, push allows devices publishing new events and data. Third, publish-subscribe mechanisms are useful when producers and consumers are decoupled in time, and data is not yet available when the request is issued. Delayed data delivery in publish-subscribe is supported in an extension, CoAP observe [Hartke 2015].

Table 1.2 gives an example of how the RESTful architecture is used in CoAP to manage an IoT environment.

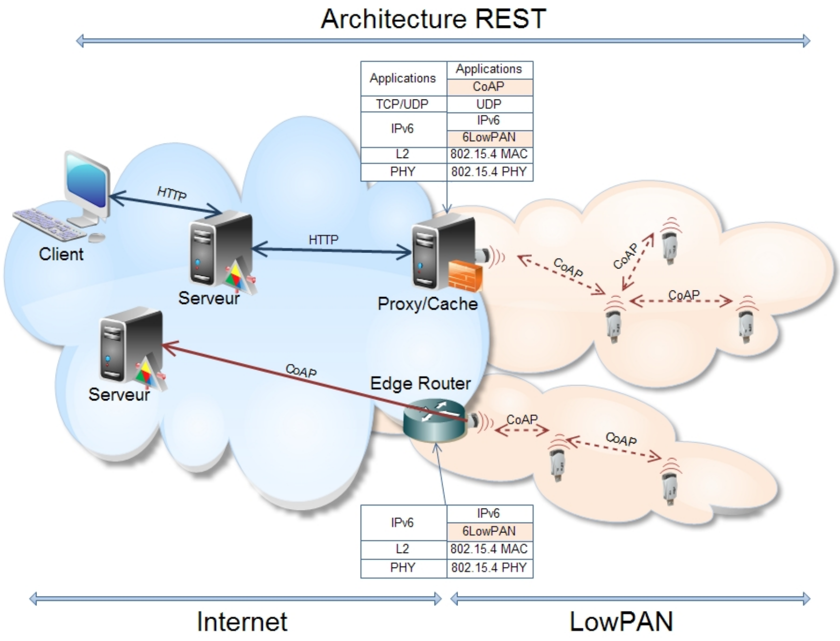
Another solution developed to support IoT scenarios with IP is the Message Queue Telemetry Transport (MQTT) protocol [OASIS]. MQTT is a broker-based publish-subscribe messaging protocol. Clients can publish content, subscribe to content, or both. Brokers (servers) distribute messages between publishing and subscribing clients. MQTT is a lightweight protocol suitable for constrained IoT devices.

MQTT relies on TCP to provide reliable communications. However, MQTT implements its own Quality of Service through three QoS levels. With QoS-0, a receiver gets a message at most once without re-transmission on the application layer. QoS-1 ensures that at least one receiver gets the published message. In other words, the sender stores its message until an acknowledgement is received, and re-transmits the message when an acknowledgement is missing. QoS-2 ensures that a message is received exactly once, to avoid packet loss or processing of duplicates at the MQTT receiver side.

MQTT-SN [IBM 2013] is an adaptation of MQTT to constrained networks such as IEEE 802.15.4. For example, publish/subscrie topic names (strings) are replaced by IDs (numbers) to reduce header complexity. Moreover, MQTT-SN is able to run on top of UDP unlike the initial MQTT.

Since the IoT became a reality, IETF WGs have made significant efforts to adapt the traditional TCP/IP stack to IoT systems. These efforts made the IoT more accessible to most users, and resulted in extensions to TCP/IP protocols and the appearance of various other protocols acting like middleware between the application layer and the network layer. However, we can observe that IP addressing is not expressive enough to manage device identity, data names and security at once. New requirements such as content discovery, caching, mobility and multicast communications make the task even more complicated for IP networking. To illustrate that, most IP solutions support IoT by implementing new features at the application layer using REST. This indicates that the TCP/IP stack has reached its limit to support these new requirements.

Figure 1.5 depicts the current IoT environment with its typical related solutions. In the following, we examine in detail the main requirements of the IoT, how current IP-based IoT systems support them, and what are the limits of these solutions.



Applications	Applications CoAP
TCP/UDP	UDP
IPv6	IPv6 6LowPAN
L2	802.15.4 MAC
PHY	802.15.4 PHY

IPv6	IPv6 6LowPAN
L2	802.15.4 MAC
PHY	802.15.4 PHY

Figure 1.5: Global IoT architecture with REST, CoAP and 6LoWPAN [Wikipedia a]

1.3.3 Requirements and Solutions

1.3.3.1 Heterogeneous constrained environment

The IoT is typically characterized by its diversity of communication technologies and limited resources. Current devices used in IoT deployments may use various communication technologies, and provide different interfaces with their respective protocols. Furthermore, they may adopt different addressing schemes and implement different middleware to bridge the gap between the network layer and the application layer. This creates a very heterogeneous environment in which applications and devices have to exchange data. Figure 1.6 depicts the three types of devices present in an IoT environment [Eclipse IoT White Paper 2017].

Although request-response APIs standardize communications at application level, they require maintaining a mapping between device interfaces and the data identified through URIs. Mappings that make API available over a host-centric protocol stack become even more complicated when the environment changes dynamically due to mobility and coexistence of many different technologies. Although solutions such as ZigBee WSNs [ZigBee

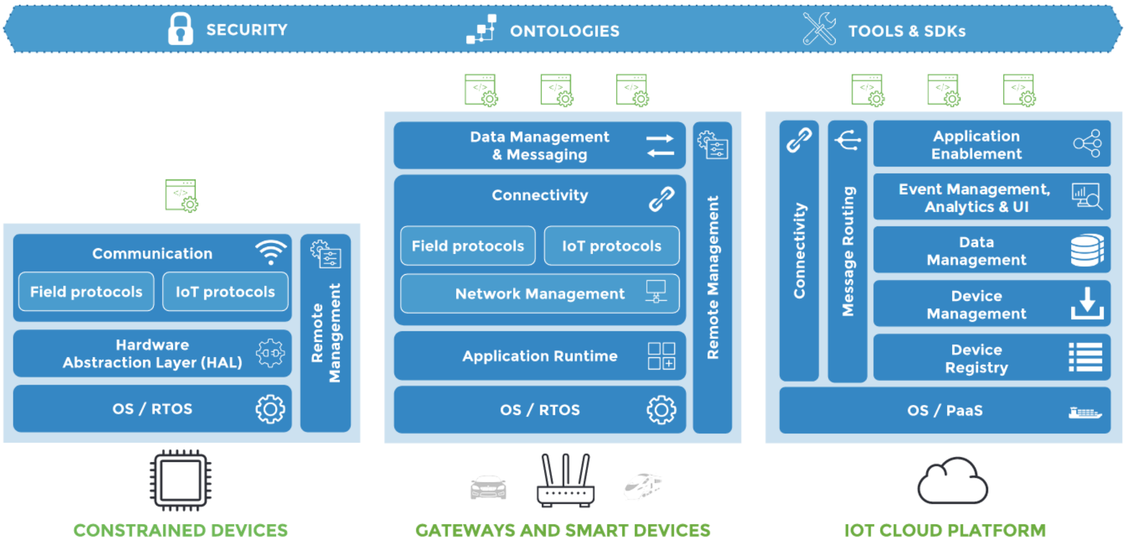


Figure 1.6: Types of devices in the IoT [Eclipse IoT White Paper 2017]

Alliance] and Cloud-based platforms provide satisfactory IoT applications based on the TCP/IP stack, they only support a specific technology or a particular network type, without a totally secured environment. In the cow monitoring example, the analyser requests data through its name, while the data is collected by a sensor with a certain IP address. Moreover, different technologies may be used at the same time according to the environment and deployment cost, which may also bring different link-layer addressing. Therefore, a continuous mapping between addresses and names is needed to ensure data availability.

Table 1.3 gives the typical resources available in constrained devices as defined in [Bormann et al. 2014]. Resource-constrained devices present in IoT systems provide limited communication and processing capabilities. To save energy, devices spend most of their lifetime in sleep mode. To deal with that, protocols must provide energy-efficient design which is currently not supported at the network layer. Rather, lower and upper layer solutions are combined to manage energy. This is done in the MAC layer through clustering, synchronization, etc., and in the transport layer through connection-less and asynchronous communications over UDP. Furthermore, approaches like IPv6 header compression may reduce communication time and save energy, but at the cost of more memory and processing requirements.

This takes us to the other limitation of IoT constrained environments: the reduced

Table 1.3: Classes of constrained devices

Class	**Data size (e.g., RAM)**	**Code size (e.g., Flash)**
Class 0	$\ll$ 10 KiB	$\ll$ 100 KiB
Class 1	$\sim$ 10 KiB	$\sim$ 100 KiB
Class 2	$\sim$ 50 KiB	$\sim$ 250 KiB

MTUs offered by low-power wireless links. One of the technologies that enables massive device deployment in the IoT is the IEEE 802.15.4 which has a frame size around 127 bytes of which no more than 100 bytes are available for the payload. The problem is that IP was not designed to support such reduced MTUs. IPv6 uses a fixed length header of 40 bytes to improve packet processing speed, and assumes a minimum MTU of 1280 bytes to avoid fragmentation. This is reasonable for traditional networks but the constraints of the IoT are not considered at all. To cope with this issue, 6LoWPAN [Montenegro et al. 2007] was introduced as an adaptation layer between the network layer and the link layer to enable IPv6 networking over IEEE 802.15.4. This layer includes the mechanisms needed to support small MTUs such as header compression and packet fragmentation. Header compression is used for IPv6 headers and UDP to reduce the size of the header in the majority of the transmitted packets, providing more space for application data. Packet fragmentation consists in splitting a standard IPv6 packet (i.e., 1280 bytes) into multiple link layer frames (i.e., 127 bytes). Although these two operations enable IPv6 in low-power wireless networks such as IEEE 802.15.4, they bring additional overhead and processing, and consume more memory which is already rare in IoT devices. Furthermore, the need for this adaptation layer shows one of the limits of traditional IP design to support resource-constrained IoT devices.

The standardization effort in the IETF has made the new standard for header compression, ROHC [Jonsson et al. 2006]. Its main advantage is its ability to withstand high error rates and delays. Although the effort initially focused on UDP because this protocol is the media flow vector, work is underway to improve the efficiency of TCP stream compression.

Nevertheless, the IP protocol only moves packets between two hosts. How to move a data object over the network is not defined in the IP specification (i.e. RC791) because it is a transport problem. In the TCP layer, data streams between a sender and a receiver are used to provide reliable packet delivery and congestion control. However, some TCP features do not accommodate constrained IoT environments. As an example, in the cow monitoring system, devices need to send only a small amount of data at one time. Establishing connections to send the temperature of a cow is not reasonable, and it is not feasible to maintain connections while IoT devices are frequently in sleep mode. To provide connectionless communication and reduce overhead, UDP is frequently used in IoT protocols such as CoAP. However, UDP only provides data stream multiplexing in the transport layer, which forces the application layer to implement complex transport functionalities when needed.

1.3.3.2 Wireless networks

As mentioned above, IoT devices typically use wireless links to communicate with the rest of the Internet. Devices then typically get Internet access through a gateway. To reach the gateway, devices are connected either in a star or mesh topology. While the star topology is easy to deploy and simple to manage, every device in the network must be able to reach the gateway in one hop in order to communicate. In the mesh topology, any device that can reach another device may receive and transmit packets. In this way, nodes may act as routers and relay packets from any node to the gateway, and vice-versa. In our cow monitoring example, the star topology can be adopted in closed environments where the cows are relatively close to the gateway. However, the mesh topology can be useful to provide connectivity in the fields, for example to keep using a technology with good reliability but a small communication range. Routing packets in a mesh network can be envisioned either at the network layer (known as route-over) or the link layer (known as mesh-under). In the mesh-under approach, multiple link-layer hops form a single IP hop, while the route-over approach associates each link-layer hop to an IP-hop. To support mesh-under routing, a link-layer addressing scheme is used to reflect the topology configuration, and nodes maintain link-state information with their neighbors. However, when the network topology changes, the address allocation process needs to be performed again to adapt to the new one.

The first routing protocol standardized by IETF to support LLNs is the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [Alexander et al. 2012]. RPL is a route-over solution that supports point-to-point, multipoint-to-multipoint and point-to-multipoint communication patterns. RPL organizes the network topology as a Directed Acyclic Graph (DAG) divided into Destination-Oriented DAGs (DODAGs). One RPL instance has one or more DODAG roots. When two nodes inside a DODAG communicate with each other, their packets are forwarded up to a common ancestor (or the root), then sent down to the destination. However, maintaining routing tables can be challenging in certain situations (e.g. nodes near the the root). Although a mode where only the root maintains the routing table exists, the full route information needs to be inserted in the packet by the root. Hence, memory usage is reduced on the other nodes but the header size is increased in some packets. Moreover, RPL has difficulties to support dense networks and mobile environments [Lamaazi et al. 2018].

In short, the challenges that IP faces in wireless mesh routing come from logical node addressing that requires explicit routing information to operate. Unlike in traditional IP networks, maintaining per-node and link-state information can rapidly become an issue in resource-constrained IoT devices, especially under dynamic topologies.

1.3.3.3 Resource discovery

In the cow monitoring scenario, some types of data need to be published in specific situations or at a given time (e.g., in the milking parlor). Therefore, devices may need to know if there is a service to which they can publish their data, or the analyser software may need to know which data are provided by each sensor. For that, a resource discovery mechanism is useful. Resource discovery in IP networks involves three operations in order to be effective: automatic assignment of network addresses for devices, automatic distribution and resolution of host names, and automatic location of network services, such as a printing device. In traditional networks, resource discovery is based on DNS, and called DNS-based Service Discovery (DNS-SD) [Cheshire and Krochmal 2013]. However, DNS-SD is designed to discover services identified by running programs whereas IoT applications handle more general resources such as services, devices, data and so on. DNS-SD often uses Multicast DNS (mDNS) communication for service discovery in the local network, but multicast traffic may not be supported efficiently in IP-based constrained networks. The main limit of this approach is that direct discovery of resources is frequently not feasible due to sleeping nodes, dynamic networks and link-layer constraints. Furthermore, multicast requires trusting the entire network rather than a designated DNS server, which makes it vulnerable to spoofing attacks.

For IoT environments, the IETF CoRE WG has developed a resource discovery mechanism based on CoAP. CoRA-RD [Lynn et al. 2019] uses a Resource Directory (RD) to store resources provided by hosts. However, such an important feature for the IoT would be more useful if were supported in the network layer as it could be combined with packet routing for example. This happened because TCP/IP layers are not designed to understand named resources used by applications. As a result, the Neighbor Discovery protocol for IPv6 can only discover configurations at the network layer and below, whereas resources are related to IP addresses and port numbers such as in DNS-SD.

1.3.3.4 Mobility

IP communications require an IP address for every device and every interface, and a valid address is required every time the location changes. In the cow monitoring scenario, devices (i.e., cows) are frequently moving, and each area may form its specific local network. Monitoring may be able to fetch and authenticate data from each cow regardless of its location on the farm, and with a minimum of latency. However, mobility in our scenario is relatively reduced compared to other IoT applications.

According to [Meisel et al. 2010], current Internet protocols are not suitable for highly mobile environments like MANETs and VANETs. To support node mobility with IP, most solutions (e.g., Mobile IP [Perkins 2002]) rely on a mapping between the stable IP address of the mobile and its changing address, and use traffic redirection. When its location changes, a mobile updates the mapping service. However, tracking and updating location changes of each mobile raises scalability issues. Moreover, this approach does not deal with content mobility, particularly when new communication patterns such as many-to-many are present. In addition, since IP security is based on IP addresses, the secured communication must be reestablished when a communicating node moves. This is inefficient when nodes move frequently. Opportunistic solutions have been proposed to take advantage of the broadcast nature of wireless technologies. For example, one approach [Biswas and Morris 2005] consists in framing data units within bundles and provides a binding of data names to host address. However, the content delivery is still based on IP addresses, and as long as IP address assignment and management is required, the limits remain the same.

1.3.3.5 Security

Security in IoT deployments is important as devices that interact with the physical world can be requested and controlled over the Internet. IP does not integrate security as a native feature and it has had to be designed later on and supported somewhere at data-link, network, transport or application layer.

The security model provided by IP is channel-based; that is, it provides and maintains a secured communication channel between two hosts. Hence, all IP-based security protocols (e.g., IPsec [Frankel and Krishnan 2011], TLS/SSL [Rescorla and Dierks 2008] , DTLS [Rescorla and Modadugu 2012]) are based on host network addresses. However, since security protocols are based on the TCP/IP model, they inherit its overhead issues for establishing a secured channel (e.g. two or more rounds of security handshake for TLS), and maintaining channel states requires more memory usage, which increases linearly with the number of peers the node is simultaneously communicating with. Moreover, channel-based security makes it difficult to ensure data security in the presence of caching for example, as it requires both client and server nodes to be online to secure the data. Nevertheless, securing a channel when exchanging data does not ensure that the content received is itself authentic, and the security of the content is the most important for applications. Moreover, securing channels in the case of a communication that involves mobile devices or multiple parties is cumbersome and increases network overhead. For example, mDNS requires trusting the entire network rather than a designated DNS server, which makes it vulnerable to spoofing attacks by any system within the multicast IP range. It can be used by attackers to get detailed knowledge of the network. Because of this, applications should still authenticate and encrypt traffic to remote hosts (e.g. via RSA, SSH, etc.) after discovering them through DNS-SD/mDNS. Figure 1.7 illustrates the security solutions involved in IoT deployments with 6LoWPAN.

In short, IP-based security does not meet IoT requirements because it secures channels between things whereas IoT applications require securing the things themselves together with their data. A typical example that illustrates a security situation is a device deployed in the cow monitoring system. Such a device may communicate over several interface types: low-power wireless, Ethernet or USB. Exchanging data and executing actions over each of these interfaces requires different networking stacks: power-aware communication over the wireless radio, IP for Ethernet, and file-based access for USB. Each stack has different security solutions. Further, securing a channel between devices must deal with device identity, verify data authenticity and express trust relationships.

To overcome the channel-based security limitations, the IETF CoRE WG proposed an Object Security for Constrained RESTful Environments (OSCORE) mechanism [Selander et al. 2019]. It is an object-based security solution for application-layer data protection over CoAP. OSCORE is designed for constrained nodes and provides end-to-end protection between endpoints communicating using CoAP. For that, each data object should carry authentication information such as digital signatures so that anyone receiving the data can verify its validity regardless of how the data is retrieved.

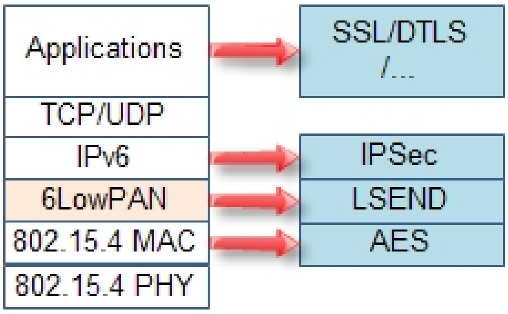


Figure 1.7: Security solutions in IoT with 6LoWPAN [Wikipedia a]

1.3.3.6 Caching

In our cow monitoring application, devices are frequently in sleep mode to save energy. Hence, a device related to a cow may not be online when a data collection is triggered by the analyser. In addition, such a dynamic scenario makes it difficult to maintain connections between hosts. To deal with this, caching and proxying techniques can be included in the system. Caching and proxying mainly aim to save bandwidth and reduce latency. For example, a node can store content to serve similar future requests. A proxy node can prefetch and store content on behalf of sleeping nodes, and the same approach can be used by content producers that select proxy nodes to act on their behalf (reverse-proxy). However, these techniques are in contradiction with the TCP/IP protocols which require that both the client and a the server are online during the communication. Consequently, caching/proxying techniques are implemented at application level (CoAP and HTTP) and create several issues in IoT environments. The selected nodes need to be explicitly chosen, and additional computation and communication is required to find the appropriate node for each communicating host. In dynamic environments, nodes need to frequently reconfigure proxies and update information. Security protocols are based on end-to-end connections, but storing and reusing cached data makes it unsecured.

It is worth observing that issues raised by caching are basically due to addressing hosts combined with the fact that content is completely opaque to the network layer. Indeed, caching handles application data while network protocols are based on IP addresses, and ensuring a continuous binding between data and hosts generates communication overhead and extra complexity in network configuration.

1.4 From IP limitations to ICN

1.4.1 Summary of IP-for-IoT Efforts

The IETF is making considerable efforts to design protocols for constrained environments based on IP. The Constrained RESTful Environments (CoRE) group proposes CoAP to allow IoT devices to exchange data as in the Web, and OSCORE for securing data objects at application level. The 6LoWPAN-WG is handling the adaptation of IPv6 over low-power low-rate networks such as IEEE 802.15.4, by designing mechanisms for IPv6 header compression to deal with small MTUs. The ROLL WG is developing routing strategies and self-configurable mechanisms in low power networks and is working closely with 6LoWPAN-WG. IETF has standardized RPL for communication in constrained wireless networks. The Light-Weight Implementation Guidance (LWIG) working group is helping to build minimal and interoperable IP-capable devices for constrained environments. The Thing-2-Thing Research Group (T2TRG) focuses on issues that may influence standardization processes in the IETF, to form the real IoT in which constrained devices can communicate with each other and with the global Internet. Figure 1.8 shows the IP-based standardization efforts located in a typical network protocol stack.

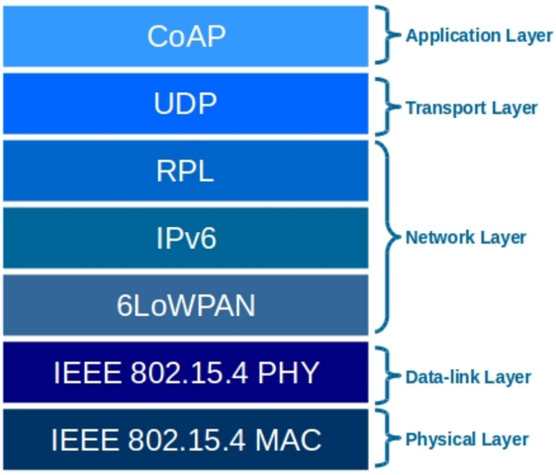


Figure 1.8: IP standardization efforts for IoT

However, by looking at the solutions proposed to provide a viable IoT over IP, we observe that the most important functionalities are implemented in the application layer, using REST as a common architecture for communication. Therefore, the heart of the current IP-based IoT architecture is the application layer with REST instead of the network layer with IP, as it is supposed to be.

This is because the host-based IP model can not support IoT requirements by itself; it needs a richer and more flexible architecture (e.g. REST) to provide caching, resource discovery and efficient security. This situation is schematized in Figure 1.9, in which a complete IP-based IoT stack is represented. Even though these solutions (i.e., implemented in application-layer) may hide IP limitations to the final users of the IoT, the conceptual mismatch between the application-layer features and the TCP/IP architecture is a reality, in particular for developers, and often leads to more complex solutions.

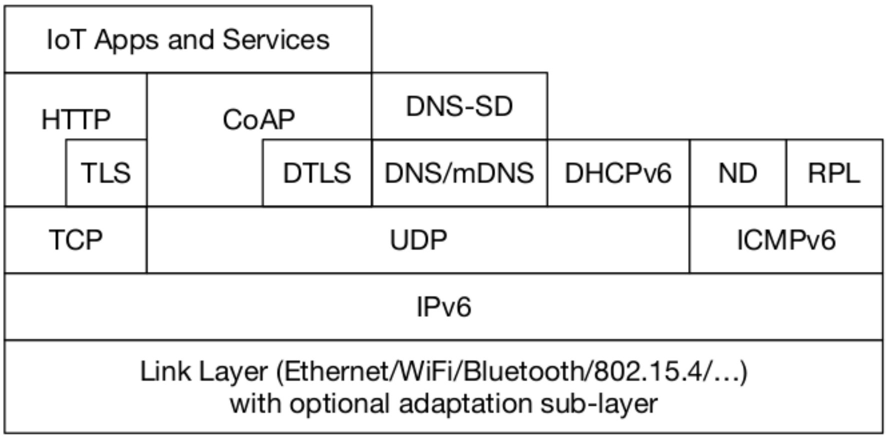

IoT Apps and Services
HTTP
CoAP
DNS-SD
TLS
DTLS
DNS/mDNS
DHCPv6
ND
RPL
TCP
UDP
ICMPv6
IPv6
Link Layer (Ethernet/WiFi/Bluetooth/802.15.4/...)
with optional adaptation sub-layer

Figure 1.9: Current IP-based IoT stack [Shang et al. 2016a]

1.4.2 Shifting to Information Centric Networking

One can easily imagine what will happen if we move the functionalities provided by the current IP-based IoT stack (see Figure 1.9) from the application layer to the network layer. In addition to being completely feasible, this can be more efficient, will reduce complexity in the application layer and greatly simplify application development. The stack obtained will then be pretty close to the Information-Centric Networking (ICN) paradigm [Ahlgren et al. 2012]. To show that, Table 1.4 summarizes the IP-based solutions for IoT discussed abovecompared to the main ICN native features, which will be presented below. We observe that the recent solutions which currently make IoT over IP feasible correspond exactly to the ICN features, except that they are provided by the core network in ICN.

ICN is attracting great interest to design the future Internet architecture. Unlike the host-centric IP networking, ICN operates with natural names. In ICN, every piece of content is identified by a unique name which applications use to request and retrieve data. Content names are independent from the host location, which means that a content item keeps the same name everywhere; at the content producer, caches and consumers. This feature is combined with self-secured content to provide reusable packets and enable in-

Table 1.4: IP-based solutions for IoT vs. ICN features

IoT requirements	**IP-based efforts**	**ICN native features**
Resource naming	DNS, URI	Named content
Application data security	Object-based security	Self-secured packets
Request-response model	CoAP, REST	Consumer driven model
Small MTU	6LoWPAN	No fixed packet size
Caching	At application layer	In-network caching
Content dissemination/discovery	Multicast, CoRA-RD	Broadcast/multicast friendly

network caching, since a packet is independent from its source and destination nodes. By considering data names rather than host addresses, ICN can match most IoT applications that focus on the content regardless of where it is located or how it is transported. IoT deployments for ICN are investigated within the ICN Research Group (ICNRG) of the IRTF [Kutscher et al. 2016].

1.4.3 ICN Principles

In recent years, various ICN architectures have been proposed such as NDN, SNAIL, PURSUIT and NetInf. Although they have different protocol designs, they share the same following principles:

- Content abstraction
- Content-centric naming and security
- Connectionless receiver-driven communication model

With these principles, all ICN architectures provide two main features: name-based networking operations and native in-network caching.

Content abstraction: ICN architectures work on Named Data Objects (NDOs). An NDO can be a web page, a photo, sensor data, or any object that computers can store and access. The NDO has a name that remains the same everywhere in the network. This means that copies of an NDO (e.g. copies in different caches) are all equally able to satisfy requests. Depending on the architecture, NDOs can be full objects or divided into packets.

Content-centric naming and security: To identify NDOs, ICN names need to be globally unique. An ICN architecture can adopt hierarchical, flat, or attribute-value names. Hierarchical names are URI-like names of variable length. Flat names are in the form P:L, where P is the ciphered hash of the public key of the content owner, and L is a unique label that identifies one content item from that owner. In the attribute-value naming, each attribute has a name, a type and a set of possible values.

In ICN, the content is self-authenticated and uniquely identified, no matter where it is. To provide an authenticity verification mechanism that works regardless of content location, ICN establishes a binding between the content, its name and the entity that created it, to allow integrity and authenticity verification. The source of the content signs together the content, its name and its origin immediately before introducing the content-object into the network. To verify whether received content is legitimate, the user just checks the signature. A security mechanism based on the content allows protection and trust to be carried in the packet itself, rather than relying on secured communication channels.

Connectionless receiver-driven communication model: Figure 1.10 gives an example of a typical data retrieval in ICN. In practice, the cow monitoring deployment with an ICN architecture does not require allocating an address to every device including cows. Rather, a consumer (e.g., farmer's smartphone) collects a cow's data by simply sending requests to the network for the named content it needs. A request is similar to a question in the form: "Does anybody have a content that matches this names?" and the content returned by the network is the answer. Hence, retrieving data in ICN is receiver-driven and consists of two phases: (i) issuing and forwarding a request from the consumer to the producer or a cache in between, and (ii) the delivery of the content back to the requester. An ICN architecture supports content discovery either through name-based routing (NBR) or using a lookup-based resolution system (LRS).

With NBR, the consumer requests content by issuing an Interest, which is forwarded hop-by-hop by intermediate nodes. The forwarding is based on a table and uses name matching to figure out which interface the Interest is sent to. Once the content has been found, it is sent back to the consumer by following the reverse patch of the Interest. To provide a reverse path for the content packet, each forwarder locally keeps a trace of forwarded Interests until the content is received or after a timeout. The forwarding table

Table 1.5: ICN projects/architectures comparison

Project	**NDN**	**MobilityFirst**	**PURSUIT**	**NetInf**
Naming	Hierarchical	Flat	Flat with structure	Flat with structure
Human-readable names	Possible	No	No	No
Security	Signature and Trust Schema	Signature, PKI indep.	Signature, PKI indep.	Signature and content hash
NDO granularity	Packets	Objects	Objects	Objects
Forwarding	-Name-based -Stateful (Interest) -Reverse path (Data)	-Distributed name resolution maps names to locators (i.e., network addresses) -forwarding based on locators	-Name resolution and rendezvous -Source routing with Bloom Filters in packets	-Hybrid name resolution and name-based forwarding -Reverse path or direct IP connection
Communication	Pull-based	Pull-based	Publish/Subscribe	Pull-based

is basically populated by a routing protocol (e.g. using names advertising).

With LRS, the request is handled by a resolution system. Each architecture using LRS adopts data structures to collect and provide information to create forwarding paths. The content is then forwarded to the consumer according to the resolution system's decisions.

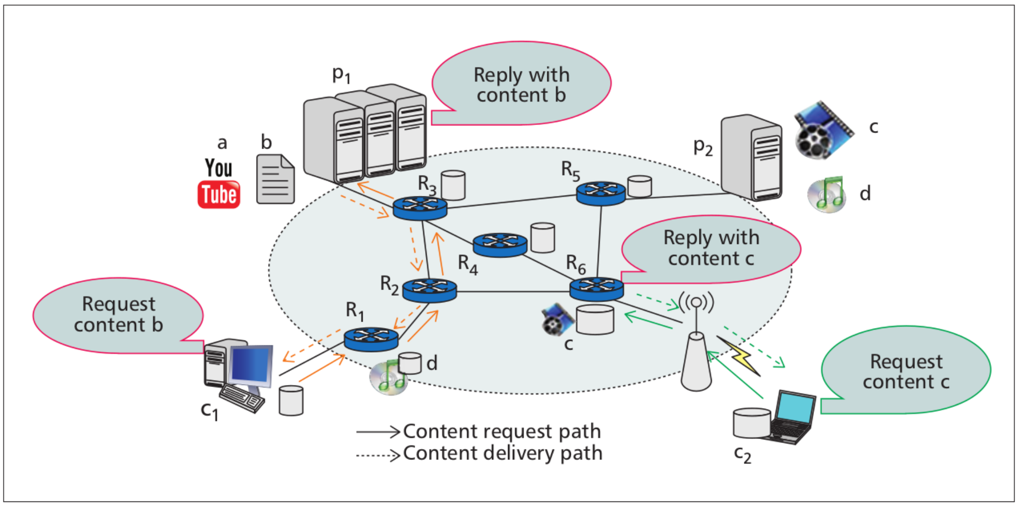


Figure 1.10: Illustration of the ICN communication paradigm [Amadeo et al. 2016]

Table 1.5 gives an overview of the best-known ICN architectures, adapted from [Ahlgren et al. 2012] and [Amadeo et al. 2016].

1.5 Conclusion

In this chapter, we introduced ICN after investigating and analysing IP solutions for the IoT. We then deduced that these solutions resemble the native features offered by ICN in the core network. We believe that discussing the IP-based approach for IoT and its shortcomings is a rational way to show the opportunity that ICN represents for building better IoT systems.

It is worth noting that no ICN architecture was especially designed for the IoT. Nevertheless, the ICN approach is still in a research phase, which is an opportunity to design future architectures with the IoT in mind.

Among the ICN architectures that have emerged in recent years, Named Data Networking (NDN) is a promising one. As mentioned above, the native features of ICN are supported differently from one realization to another. Therefore, rather than presenting the abstract ICN features, we will see in the next chapter how the NDN design provides ICN features, and how it can support IoT applications, either natively or with some simple adaptations.

Chapter 2

Named Data Networking for the Internet of Things

2.1 Introduction

In the previous chapter, we found out that even if IP is applicable for IoT systems, the complexity of IP-based solutions (e.g., 6LoWPAN) makes one wonder about alternatives that can be more suitable for a global IoT ecosystem. In the recent years, NDN has emerged as new and a promising ICN architecture to efficiently support IoT requirements. However, various approaches are possible to take advantage of NDN in the IoT.

In this chapter we first explain the principles of NDN and its features. After that, we show how NDN is suitable for IoT architectures, by reporting on studies and proposals that we consider as inspiring work for the contributions subsequently presented in this document.

2.2 Named Data Networking

2.2.1 Origins and Overview

The concept of ICN was first introduced by Ted Nelson in 1979 [Xylomenos et al. 2014]. Twenty years later, the Translating Relaying Internetwork Architecture Integrating Active Directories (TRIAD) was proposed as the next generation Internet architecture to avoid DNS lookups. In 2002, Brent Baccala presented an Internet draft presenting the differences between host-oriented and data-oriented networking [Baccala 2002]. In 2006, the Data-Oriented Network Architecture (DONA) [Koponen et al. 2007] project at UC Berkeley proposed the first ICN architecture, followed a few years later (2009) by PARC which announced the CCN architecture and open source implementation CCNx [Jacobson et al. 2009b]. In September 2010, the NDN project [Zhang et al. 2010] was funded by the National Science Foundation (NSF) as one of the four projects under NSF's Future Internet Architecture (FIA) program. The NDN vision is based on the following principles as stated in the project [NDN Website]:

1. Universality: NDN should be a common network protocol for all applications and network environments.
2. Data-Centricity and Data Immutability: NDN should fetch uniquely named, immutable "data packets" requested using "interest packets".
3. Securing Data Directly: Security should be the property of data packets, staying the same whether the packets are in motion or at rest.
4. Hierarchical Naming: Packets should carry hierarchical names to enable demultiplexing and provide structured context.
5. In-Network Name Discovery: Interests should be able use incomplete names to retrieve data packets.
6. Hop-by-Hop Flow Balance: Over each link, one Interest packet should bring back no more than one Data packet.

To give a simple idea of NDN, we can imagine it as the HTTP's request-response model running at the network layer. In the remainder of this document, a "consumer" is an application that issues requests for content while a "producer" is the application that sends the response to satisfy this request. To return to the cow monitoring example, producers are running on sensors deployed on the cows, and a consumer may be the farmer's smartphone that requests and displays collected data.

The main difference with HTTP is that NDN supports the request-response pattern through packets carrying names as the main information, and all the networking operations (e.g., routing, forwarding, etc.) operate on those names, not on binary network addresses.

We should observe that NDN is more than just a case of shifting HTTP to the network layer. Two important differences must be highlighted: (i) In NDN, Data packets are immutable; that is, once a Data has been produced with a certain name it can not be modified. When a new version of the Data is available, the producer must generate a new packet with a new name. (ii) every Data packet is self-secured by carrying a digital signature that binds its name to its content. This signature is generated by the producer at the packet creation time. Upon retrieving a Data, the consumer verifies the signature to ensure that the content corresponds to the requested name and has actually been produced by the right entity. This security approach provides NDN with a content-based security instead of securing communication channels.

2.2.2 Naming and Packets

Another dissimilarity between NDN and IP is that NDN packets (including names) are encoded in the TLV (Type-Length-Value) format [Team]. TLV encoding represents an NDN packet as a collection of sub-TLVs, without a packet header or protocol version. A TLV block consists on a sequence of bytes starting with a predefined number (Type), followed by its Length and its Value.

Two types of packets are defined in NDN to perform communication: Interest and Data. Both packets contain a name and may carry additional information according to the defined fields described below. In the remaining of this document, Interest and Data (with capital letter) refer to the NDN Interest and Data packets respectively. Although Interest and Data packets have default and optional fields respectively (see Figure 2.1), they do not have predefined packet size or field sizes.

2.2.2.1 Names

A content is identified through hierarchical name that contains a sequence of name components [NDN Project Team 2014].

Each packet must contain a *Name* element. Name is represented by a 2-level nested TLV. The outer TLV indicates the complete Name element through the TLV-type (7).

Table 2.1: Name component types

Type	TLV-type	Description
ImplicitSha256DigestComponent	1	Implicit SHA-256 digest
ParametersSha256DigestComponent	2	SHA-256 digest of Interest Parameters
GenericNameComponent	8	Generic name component
KeywordNameComponent	32	Well-known keyword
SegmentNameComponent	33	Segment number
ByteOffsetNameComponent	34	Byte offset
VersionNameComponent	35	Version number
TimestampNameComponent	36	Unix timestamp in microseconds
SequenceNumNameComponent	37	Sequence number

Inner TLVs should be *NameComponent* elements as defined in Table 2.1. *GenericNameComponent* is a generic name component, without any restrictions on the content of the value. Functional name components can express time stamping and/or versioning to distinguish which data is the most recent, segmenting to split large data into smaller packets, and sequencing to handle sequential data collections. In particular, *ImplicitSha256DigestComponent* is an implicit SHA-256 digest component and it is required to contain a value of 32 octets. *ParametersSha256DigestComponent* is a component carrying the SHA-256 digest of Interest parameters (see Interest below) and it is required to contain a value of 32 octets.

For example, the name *"/farm/room/1/cow/21/temp"* may identify the temperature value related to the cow with Id. 21 located in room 1. With hierarchically structured names, the same data type related to another cow in another room can be named *"/farm/room/2/cow/34/temp"*.

Naming schemes are defined by the applications, which provides flexibility in the way the content is named and requested. Consequently, names are opaque to the network. In other words, routers access name components separately for routing and forwarding purposes, but they do not interpret the whole name. This allows application developers and users to design the name-space that suits their needs, without the need to maintain a mapping between network requirements and application configuration.

2.2.2.2 Packets

Applications use units of information to represent the data they handle in a most suitable form. These are commonly designated as Application Data Units (ADUs). For example in the cow monitoring system, ADUs may be the sensor readings.

NDN applications communicate by exchanging Interest and Data packets identifying data names. To send large ADUs (e.g., video streaming) over the network, the first option at application level is to use segmenting and/or sequencing. This approach is simple and does not cause extra computation or additional header in the link-layer. Sensor readings of cow movements over a day can rapidly grow in size. To transmit all the collected data, a producer can split it into segments explicitly identified in the names. A straightforward way to name these segments is to use sequentially incremented numbers, such as *"/farm/room/1/cow/21/mvmnt/1"*, *"/farm/room/1/cow/21/mvmnt/2"*, etc.

Using the name matching properties of NDN, a consumer application that requests cow movement data for the name *"/farm/room/1/cow/21/mvmnt"* will receive a Data packet named *"/farm/room/1/cow/21/mvmnt/1"*. The consumer can then send Interests that specify explicit segment numbers to retrieve all segments of the requested data.

The major difference between how TCP/IP and NDN handle segmentation is that, TCP segment numbering does not necessarily correspond to ADUs boundaries, which can only be known after segment reassembly at the receiver application. In NDN, data names expose the ADU boundaries, therefore segmentation obeys ADU boundaries.

However, splitting content into multiple explicitly-named chunks is not always the best solution. For example, since each chunk is a signed Data packet, segmentation can become computationally expensive for both producers and consumers, particularly when using public key cryptography. Moreover, in some networks with reduced MTUs (e.g. IEEE 802.15.4), and given that Data signature, of at least 32 bytes (to 255 bytes) is mandatory, it is difficult to make every Data packet fits into one frame even with segmentation.

The second option is then to use packet fragmentation. In traditional networks with NDN, a hop-by-hop fragmentation and reassembly is used when a packet is larger than the link MTU. Because routers need the entire Interest and Data packets to perform

NDN operations (i.e. forwarding, matching, caching), each fragmented packet must be immediately reassembled by the next node. This fragmentation mode seems to be the only one possible for NDN; the reasons are widely explained in [Afanasyev et al. 2015].

Several approaches have been already proposed. In the case of low-power wireless technologies, [Shang et al. 2016b] uses a lightweight fragmentation scheme that introduces 3-byte header to each fragment to transmit large packets over IEEE 802.15.4 links. Another approach is used in [Shi and Zhang 2012] and [Mosko and Tschudin 2016], which consists on a new NDN message type to encapsulate message fragments. Obviously, this approach brings more complexity in the NDN architecture and requires more memory consumption and control messages.

However, packet fragmentation usually causes extra computation, larger header size and increases latency, especially with resource-constrained devices. Given this, packet fragmentation/reassembly should be avoided as much as possible.

A popular solution to avoid fragmentation while increasing the amount of data transmitted is known as header compression. This approach is widely used in IP network as described further. Although header compression for NDN is not mature yet, we investigate one possible approach presented in this dissertation.

2.2.2.3 Interest packet

An Interest represents the request issued by a consumer (see Figure 2.1).

CanBePrefix, *MustBeFresh*, *InterestLifetime*, and *ForwardingHint* are optional elements that give more information on Interest matching or forwarding. The presence of *MustBeFresh* indicates that a forwarder can not satisfy the Interest with a Data from its local CS if it is stale (see *FreshnessPeriod* in Data packet). The *ForwardingHint* element contains a list of name delegations. Each delegation implies that the requested Data packet can be retrieved by forwarding the Interest along the delegation path.

The *Nonce* contains a value of four random octets. The combination of Name and Nonce should uniquely identify an Interest packet, and used to detect looping Interests. Nonce is required when an Interest is transmitted over the network links. That is, a forwarder must add a Nonce to the Interest if it is missing.

InterestLifetime indicates the time (in *ms*) remaining before the Interest times out. The timeout is relative to the arrival time of the Interest at the current node. Forwarders may decrease the lifetime of an Interest to account for the time spent in the node before forwarding, although it is not required. The value of InterestLifetime is set by the application, and the default value is 4000 ms.

The *HopLimit* element indicates the number of hops the Interest is allowed to be forwarded. The value is encoded as a 1-byte unsigned integer value in the range $0 - 255$.

An optional *ApplicationParameters* element may be included in the Interest. This element can carry any arbitrary data that parameterizes the request for Data.

If an Interest contains *InterestSignatureInfo* and *InterestSignatureValue*, it is considered a Signed Interest. *Signature* is defined through two consecutive TLV blocks: *InterestSignatureInfo* and *InterestSignatureValue.* To ensure uniqueness of the signed Interest name and to mitigate potential replay attacks, the *InterestSignatureInfo* element can include a *SignatureNonce* element, *SignatureTime* element, and/or *SignatureSeqNum* element.

The signature in the *InterestSignatureValue* element covers all the *NameComponent* elements inside *Name* up to, but not including, *ParametersSha256DigestComponent* component, and the complete TLVs starting from *ApplicationParameters* up until, but not including, *InterestSignatureValue.*

2.2.2.4 Data packet

A Data represents the response sent by the producer (or intermediate cache) that contains the requested content (see Figure 2.1). The Name is required and has the same role as in the Interest.

The *Content* element contains the actual data, and can carry any arbitrary sequence of bytes.

The*FreshnessPeriod* indicates how long (in *ms*) a node that stores the Data in its CS should wait before marking it "*stale*". Consequently, if an Interest contains the *MustBeFresh* element, a node can not return a *stale* Data in response to this Interest. The effect is the same as if that Data does not exist in the CS.

The optional *FinalBlockId* identifies the final block in a sequence of fragments.

Signature element is required. It is defined at the end of the packet and signature computation covers all the elements before Signature. *Signature* is defined as two consecutive TLV blocks: *SignatureInfo* and *SignatureValue*. *SignatureInfo* is included in the signature calculation and fully describes the signature, signature algorithm, and any other relevant information to obtain parent certificate(s). *SignatureValue* is excluded from the signature calculation and represents actual bits of the signature and any other supporting signature material.

Figure 2.2 shows an example TLV representation of an Interest with a Name and Nonce. The Interest is identified by the type value: 0x05, the Name by 0x07, a Name-component by 0x08 and the Nonce by 0x0a.

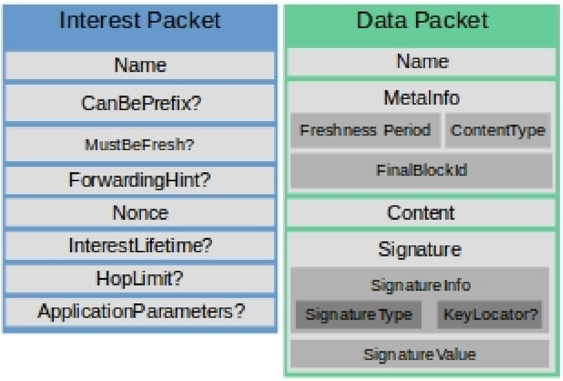


Figure 2.1: Interest and Data fields

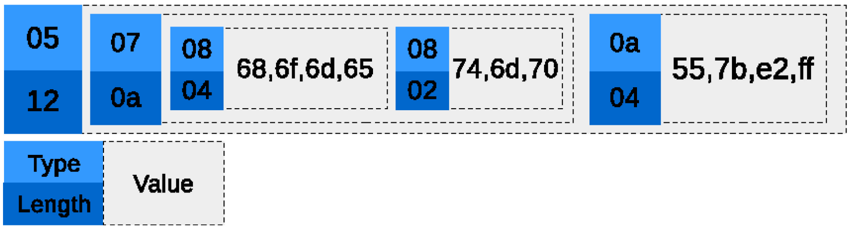


Figure 2.2: Interest TLV encoding example

2.2.3 Communication process

Each NDN node requires three data structures to process packets: FIB (Forwarding Interest Base), PIT (Pending Interest Table) and CS (Content Store). The data structures in a node are represented in Figure 2.3 and described below:

- The PIT maintains an entry for every forwarded Interest until its corresponding Data is received or until the entry lifetime has expired. A typical PIT entry contains the Interest, its incoming interface(s), the interface(s) to which it has been forwarded and a timer for Interest timeout. PIT entries are used to keep trace of Interests in order to forward the Data packet to the consumer(s) according to the exact matching of Interest and Data names. The PIT is also used to filter Interests requesting the same content to avoid redundancy.
- The CS: Since Data packets are self-secured and not related to specific hosts, each Data packet can be reused to satisfy other Interests requesting the same content. This provides NDN with a native in-network caching, and it is managed using the CS. After retrieving a Data packet, an NDN router may store a copy of that packet in the CS before forwarding it to the next hop. Since the CS has a limited size, caching placement and replacement policies such as LRU (Least Recently Used) are used to make the most of the CS.
- The FIB contains information about the reachability of the content. A FIB entry associates a content-name prefix to the interface(s) from which the content can be retrieved. The FIB is populated by routing protocols and is checked every time a node needs to forward an Interest upstream using a longest prefix matching. When a matching is found, the Interest is forwarded to the corresponding face(s).

The typical NDN communication process applied in our cow monitoring scenario is represented in Figure 2.4 and operates as follows:

1. The NDN communication is initiated by the consumer that requests the data. The consumer application in the farmer's smartphone requests data by sending an Interest carrying the name of the data (e.g. */farm/room/1/cow/21/temp*).

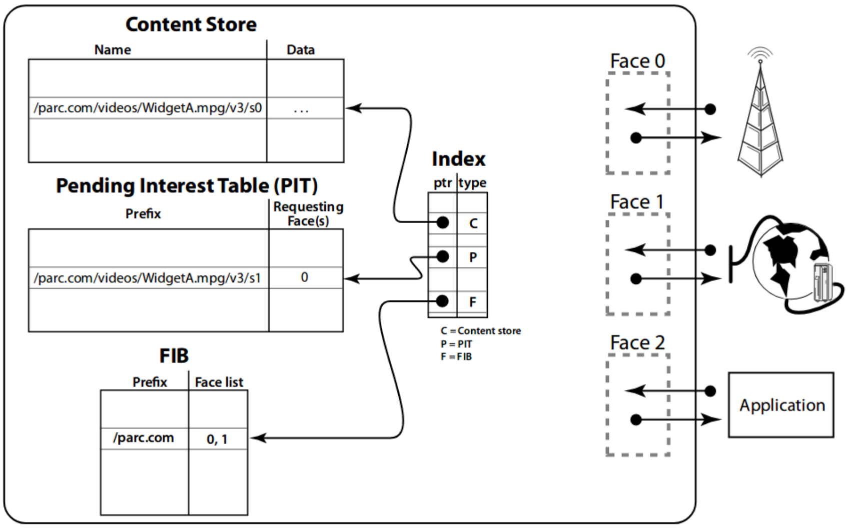


Figure 2.3: NDN node and data structures [Jacobson et al. 2009a]

2. Upon receiving an Interest, a router first checks if matching Data already exists in its CS. If the corresponding Data is found, it is sent back as a response without forwarding the Interest any further. When no matching data is found in the CS, the router checks the PIT to know whether an Interest for the same content is already waiting; if so the new Interest is not forwarded and only the originating interface is added to the existing PIT entry (Interest filtering). The Interest is forwarded only if no corresponding data is found in the CS and no similar Interest is already in the PIT. In this case, the Interest is forwarded according to the longest prefix match (LPM) against the FIB entries. For example, for this Interest name, FIB may find possible LPMs like *"/farm"*, *"/farm/room/1"* and even *"/farm/room/1/cow/21"*, and the longest one is chosen. After that, the router records the Interest in the PIT and forwards it to the corresponding interface. If no matching is found, either the Interest is flooded to all outgoing interfaces or is deleted, according to the forwarding strategy.

3. When the Interest reaches the content producer (i.e., the sensor) or an intermediate cache node, the Data packet containing the requested content is sent back. The Data packet follows the reverse path of the Interest following traces left in the PIT of each router. When a Data packet reaches a router, it is forwarded to the interfaces from

which the corresponding Interests were received. That is, all interested users (e.g., laptop and smartphone) will receive a copy of that Data. After that, the router discards the entry from the PIT, and stores the recent Data packet in its CS. If no matching entry exists in the PIT for the Data packet, (e.g., because Interest lifetime has expired), the Data is dropped.

The processing steps of Interest and Data packets at a node are depicted in Figure 2.5.

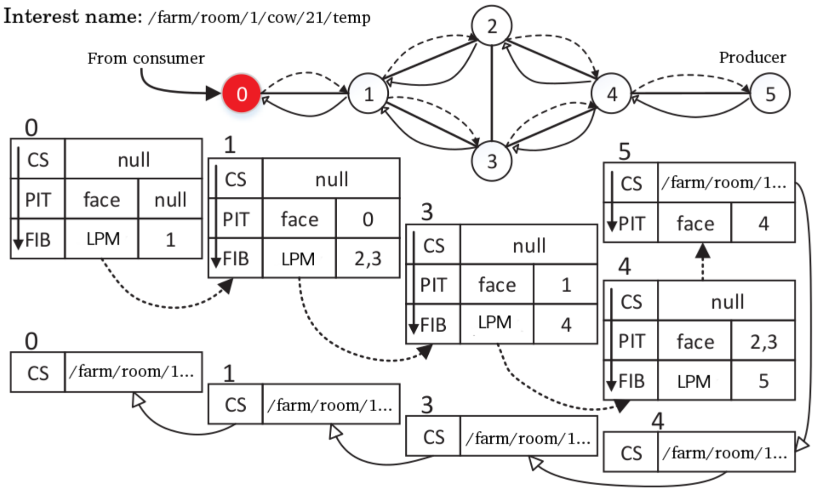


Figure 2.4: NDN communication process illustration

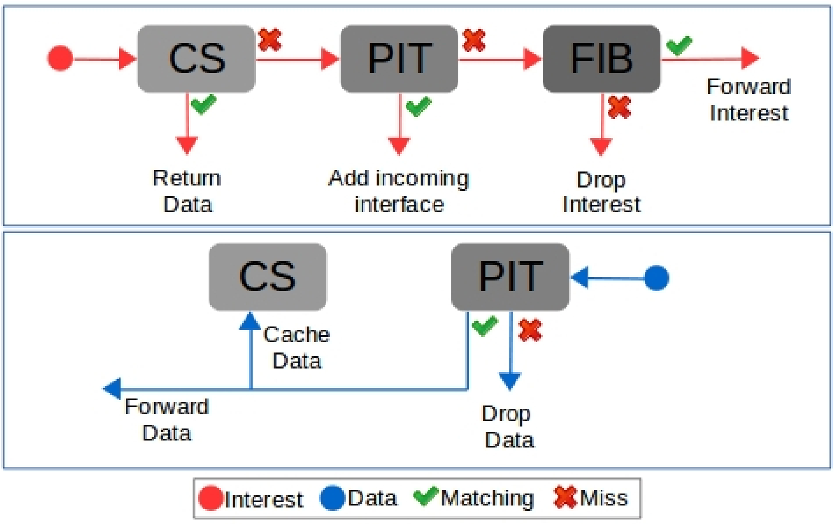


Figure 2.5: Interest and Data processing inside a node

2.2.4 Routing and Forwarding

The NDN communication process can be split into two phases: routing and forwarding. In the routing phase, reachability of the content is propagated and maintained through routing protocols. The forwarding phase exploits routing tables to deliver packets from the source to the destination. However, these two phases are different between NDN and the IP architecture. In IP, only the routing operation is smart in the sense that different routing protocols can be envisioned. The forwarding operation always consists in finding the longest match available in the routing table and sending the packet to the corresponding next hop. In NDN, in addition to the routing operation, which can be smart as in IP, multiple approaches are possible to handle packet forwarding with more or less additional information and with or without caching.

NDN uses routing protocols to propagate information about the reachability of the content. This information consists in content name prefixes. Each NDN router uses routing information to populate the FIB in order to forward Interest packets later on. Basically, routing protocols in NDN propagate content name prefixes in the same way that IP routing protocols propagate address prefixes. Hence, routing algorithms used in the Internet (e.g., link-state, distance vector) can be used for NDN with minimal adaptations such as changing the messages to Interest/Data and adding a support for multipath forwarding. While NDN routing protocols support long-term changes of network topologies and populate/update the FIB, the forwarding process makes performance decisions about the usage of FIB information.

The forwarding strategy tries to choose the best interface(s) to forward packets based on the FIB information and other design options such as flooding or best path, and interface(s) probing. For example, an NDN router monitors the forwarding tables to get forwarding state information, which can be used to calculate packet delivery performance, link failures, network congestion and even suspect behaviours (e.g., DoS attacks, etc.). Wireless networks typically require a slightly different forwarding approach than wired networks, especially in constrained wireless environments. For example, counter-based broadcasting and packet overhearing are used to minimize redundancy and collisions. A complete and detailed study of wireless forwarding in NDN is given in Chapter 5.

2.2.5 Caching and Mobility

One of the main benefits of ICN architectures is in-network caching. The native in-network caching feature is managed in NDN using the CS. CS can be compared to buffer memory in IP routers. However, packets in the CS can be reused to satisfy similar requests, while IP packets cannot be reused for another communication. In addition to accelerating retransmission after a packet loss, this feature allows NDN to achieve high performance data delivery for static content and dynamic content when multicast is needed (e.g., real-time videoconferencing). Many studies have focused on caching efficiency and optimization regarding energy consumption, wireless networks caching and so on.

By accessing content by names rather than host addresses, mobile nodes in NDN do not need to acquire an address after each location change and may continue their communication with minimal disruptions. Consequently, NDN natively supports consumer-side mobility. In the cow monitoring application, the farmer may request content items through a gateway when he is in a room. After that, he moves to another room next to the first one. The Data packet will not be received by the farmer as it will be sent to the old location. Here, the farmer's mobile application has only to re-issue the same Interest to retrieve the Data packet from a closer cache, namely the first common cache/router between the two rooms. In general, this provides a smooth hand-off because if the consumer moves to another location, then the requested content will be cached in intermediate routers.

Regarding producer-side mobility, a native support is not possible. Basically, producer mobility is supported through the following approach: the namespace under which the producer publishes its content is used as an identifier for the mobile producer. Then, a mapping can be set between the identifier and the locator of the producer when it moves. The locator can be the name prefix of the local network and can be found by broadcasting, for example. The Interest aggregation in NDN allows data to be fetched from a mobile producer with minimal overhead, even when multiple consumers are interested in the content.

Generally speaking, some studies have shown promising results in supporting mobility with NDN. In [Etefia et al. 2012], the authors demonstrated that NDN has better performance in a mobile lossy environment compared to the TCP/IP protocol suite. The authors in [Tyson et al. 2012] surveyed several new Internet architecture projects and identified benefits and challenges involved in transforming to information-centric communication in terms of mobility support.

2.2.6 Security

Security in NDN and TCP/IP are fundamentally different due to the fact that NDN names content whereas IP identifies hosts. NDN security is based on packet signature and verification through public-key cryptography. Every entity participating in communication (e.g., an application) uses name(s) and public-private key pair(s). The binding between an application, the name(s) it uses and its key(s) is provided by certificates. In practice, a certificate is a normal Data packet that contains public key information related to a certain name, and certifies that the name and the key belong to the specified user. That is, the certificate must be signed by a superior entity that issued it. Typically, an entity can issue certificates to other entities allowed to produce content under its sub-namespaces. Following this approach, each entity in the system is certified by its superior entity, until we reach the authority of the system. Hence, the authority of each system needs a local trust anchor that proves its identity to allow recursive identity verification of each entity. Access control and confidentiality are also supported through public-key (combined with symmetric key) encryption.

The required elements to produce authenticated and verifiable content are the trust anchor, the certificates, and trust policies. A trust policy is defined by the applications and gives the rules that must be respected to verify the trustworthiness of a packet. This means that, consumers can only accept packets with appropriate names and signed by the appropriate keys. Policies can be expressed as proposed in [Yu et al. 2015]. Trust policies limit the power of each signing key and ensure that each trustworthy packet is signed by a legitimate key, providing data authenticity at a fine granularity.

To illustrate the NDN security principles, sample steps to secure the cow monitoring application are given below.

NDN security example

In our cow monitoring security example, we consider the farmer's laptop (designated as "Farmer") that requests and analyses data produced by cow sensors (designated as "Cow"). For that, all data produced by "Cow" and "Farmer" must be be authenticable. The system must also prevent malicious users from producing fake data.

To control who can access and who can produce data within the farm, the farm manager entity (designated as "Alice") is needed to issue and authenticate certificates locally. Alice allows only "Farmer" to access the private data produced by "Cow". In the following, we will show how the security mechanisms in NDN can be used to achieve this according to the steps given in [Zhang et al. 2018].

Step 1: Security bootstrapping

The first step in securing NDN applications is to ensure that entities (i.e., applications) obtain trust anchors, certificates, and learn trust policies. Since Alice is the local farm manager, the trust anchor for entities within the farm is Alice's certificate. Let Alice's certificate name be *"/farm/aliceFarm/KEY/key001/farm-agent/version"*, where name components before "KEY" form the prefix allocated to Alice, and the components after "KEY" give information about the certificate, such as key number, issuer identity and version.

1. Alice obtains her certificate in another network than the farm network, where the trust anchor is *"/farm/KEY/..."*. That is, Alice obtains its certificate from the authority of the namespace *"/farm"*, which can be a service provider for example.

2. Each consumer needs trust anchors to verify data authenticity. The minimal level of security is to trust the certificate signer that issues certificates to data producers. Trust anchors can be pre-configured or obtained through a secured data exchange. We assume in our example that Alice's certificate has been manually installed in the Cow and Farmer applications.

3. In order to generate authenticable data under the name *"/farm/aliceFarm/cow/temp"*, Cow needs to obtain a certificate for that name. Here, Cow and Farmer apply for a

certificate from Alice. This way, two certificates, *"/farm/aliceFarm/cow/KEY/1/farm-agent/version"* and *"/farm/aliceFarm/farmer/KEY/1/farm-agent/version"* will be issued to Cow and Farmer, respectively.

4. To verify the authenticity of a received Data, Farmer needs trust policies. We assume Farmer has pre-configured trust policies and relies on Alice to update the configuration if needed.

Step 2: Data authenticity and integrity

After security bootstrapping, both Cow and Farmer will trust Alice and each will have trust policies and certificates under *"/farm/aliceFarm"*. Possessing the right certificates, Cow and Farmer can produce Data packets within their corresponding namespaces and sign them using their corresponding private keys.

1. Trust Policies verification: In a trust policy, the data name, the signing key name and the trust anchor name must follow explicit relationship rules as defined in [Yu et al. 2015]. To verify a Data packet, a consumer first assesses the packet's trustworthiness using trust schema. An example of a trust policy rule can be as follows: accept Data packets whose (i) name prefix is *"/farm/aliceFarm"*, (ii) signing key name prefix is *"/farm/aliceFarm/KEY"*, and (iii) certificate chain ends with the trust anchor *"/farm/aliceFarm"*. Accordingly, only packets signed by Alice and strictly under Alice's prefix are accepted.

2. Signature verification: After trust policy verification, the consumer retrieves the certificate of the corresponding producer as identified by the key name in the Data packet. This certificate will recursively point to its signer's certificate and finally arrive at an anchor. The packet is considered to be valid if all fetched certificates, including the anchor, have valid signatures and can satisfy the trust policies.

Step 3: Data confidentiality

In this example, we use a NAC (Named-based Access Control) process to illustrate confidentiality and access control. In NAC, each encryption key name will be explicitly appended to the name of the corresponding Data packet. For instance, a Data packet produced by Cow has the name:

"/farm/alicefarm/cow/data/ENCRYPTED-BY/farm/aliceFarm/E-KEY/cow", where the components after "ENCRYPTED-BY" are the encryption key name. In order to allow only Farmer to access Cow data, the production and encryption process can be as follows:

1. Alice will first generate a key pair ("E-KEY", "D-KEY") for encryption and decryption, respectively. She then produces two Data packets carrying "E-KEY" in plain text and "D-KEY" encrypted by Farmer's public key. The "E-KEY" packet name follows the format *"/farm/aliceFarm/E-KEY/cow"*, while the "D-KEY" packet name follows the format:

   "/farm/aliceFarm/D-KEY/farmer/ENCRYPTED-BY/farm/aliceFarm/farmer".

2. When producing data, Cow first generates a symmetric key for content encryption. Then, it fetches "E-KEY" and encrypts the symmetric key with it. Finally, it packs the encrypted symmetric key into a Data whose name is:

   "/farm/aliceFarm/cow/data/ENCRYPTED-BY/farm/aliceFarm/E-KEY/cow".

3. When Farmer wants to consume this data, it starts by fetching the Data packet. The Data name indicates that the content is encrypted by "E-KEY". To decrypt the content, Farmer fetches the corresponding "D-KEY" (the fetched "D-KEY" is actually encrypted by Farmer's own key). By decrypting the content in the fetched "D-KEY" Data, Farmer obtains "D-KEY" and can decrypt the symmetric key and use it to finally decrypt the content.

To send commands to Cow or other actuators in the farm, Interest packets can also be signed by the controller of the device. When receiving an Interest packet containing a command, a device can authenticate the Interest using the same process as the one used to validate Data packets.

However, the NDN security is not perfect. Even though data authenticity, integrity and access control are efficiently supported as shown above, the main threat to which an NDN network may be exposed is DoS attacks and its variants. Man-in-the-middle attacks

Table 2.2: Possible security attacks in NDN and countermeasures

Possible attacks	**Resilience**	**Countermeasures**
Network sniffing	Partial	Encrypt content and names
Man-in-the-middle	Yes	Connectionless and content encryption
Black hole	Partial	Pull-based symmetric communication
Congestion	Partial	Forwarding-rate limit
Cache monitoring	Partial	-Unpredictable name -Tunneling
Object discovery	Partial	Encrypt content names
Interest flooding	Partial	-Monitoring unsatisfied Interest per interface -Token bucket with per interface fairness -FIB-based Interest flooding
Content/cache poisoning	Partial	Content authentication
Cache pollution	Partial	-Selective caching -Add delay for cached content
Scanning	Yes	Scanning all possible names is impossible

are almost infeasible in NDN because packets are not related to the communication of two identified hosts. Similarly, creating black-holes using prefix hijacking is attenuated by the communication mechanism of NDN which ensures a symmetry between Interest and Data. However, Interest flooding can be considered as the simplest attack that can be performed in an NDN network. The attacker can achieve that by using an existing prefix name to which it appends a sequential number. To address that, many solutions have been proposed, such as limiting the number of forwarded Interests per face, and Interest acceptance according to their satisfaction rate. The same type of attack can be performed in caches to cause cache pollution. This can be handled by setting an expiry delay for cached Data for example.

Table 2.2 summarizes some security attacks possible in NDN and their possible countermeasures adapted from [Saxena et al. 2016].

Table 2.3: NDN vs TCP/IP support of the Internet

	TCP/IP	**NDN**
Addressing	-IP addresses	-Content names
Routing	-IP prefixes -Single-path forwarding -Stateless forwarding -Loop-free handled by routing protocols	-Name prefixes -Multipath forwarding -Stateful forwarding -Interests can not loop (Name and Nonce)
Transport	-Congestion control by protocols (e.g., TCP) -Data acknowledgment by protocols -Data stream multiplexing by protocols	-FIB-rate limit -Data itself -Names used for multiplexing
Error detection	-Link and transport	-Link and network
Caching	-Application level	-In-network caching
Security	-Channel-based with protocols	-Network layer with signed data

2.3 NDN and Internet

Returning to the TCP/IP protocol stack used in the Internet, we present a global comparison between NDN features and TCP/IP features. In Table 2.3, we summarize how NDN and TCP/IP support Internet functionalities. Then, in Figure 2.6 we show the hourglass architecture that characterizes the Internet with both the TCP/IP and NDN protocols.

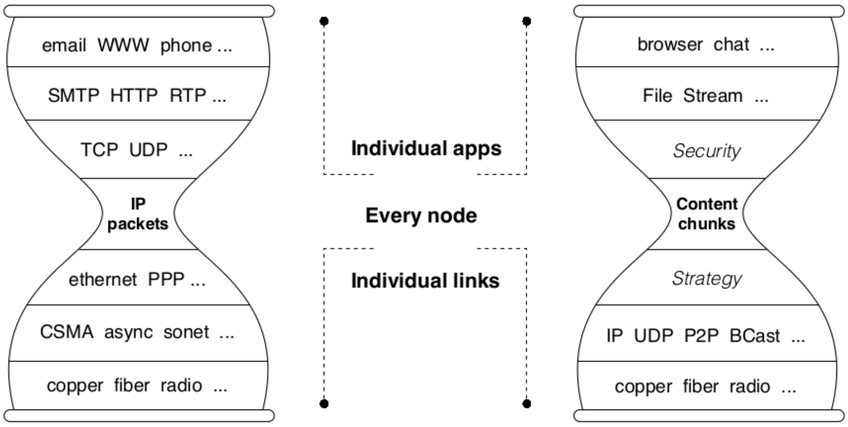


Figure 2.6: Hourglass architecture of NDN and TCP/IP [Zhang et al. 2014]

2.4 NDN meets IoT

As mentioned, NDN was not designed explicitly for the IoT. However, there are many NDN proposals and studies that can be useful for the IoT. In this section, we present some of the studies that either technically improve the NDN support for IoT, or propose

Table 2.4: IoT requirements mapped to ICN features

IoT Requirements	**ICN Features**
Scalability	Naming, in-network caching
Naming and addressing	Naming and name resolution
Mobility	Content naming, receiver-driven mode, location independent names
Security and privacy	Content-based security, Receiver-driven mode
Heterogeneity and inter-operability	Naming, strategy layer
Data availability	In-network caching, conectionless mode
Energy efficiency	In-network caching, naming

designs and visions to enable a viable NDN solution for IoT. Obviously, not all the NDN studies related to IoT are presented here; rather, we focus on those that were inspiring, encouraging, or solutions that can address challenges identified in this document. In addition, useful implementations and tools are reported as related work since our purpose is to realize a realistic IoT deployment with NDN.

2.4.1 Architectures

In [Arshad et al. 2019], a state-of-the-art on how ICN is used in the IoT is presented. The authors first give an overview of the components involved in the IoT and the required features for an IoT network architecture. They also present important challenges and issues in creating an ICN-based IoT. In Table 2.4, we report the proposed mapping between ICN features and IoT requirements.

A high-level NDN architecture is presented in [Amadeo et al. 2014a] to support the IoT features. The designed architecture consists of three layers: Thing, Network and Application layer as reported in Figure 2.7. The network layer is supported by NDN through two components: the Data plane that handles operations related to the packets, which are naming, security, caching and strategy. The second component is the Management and Control plane intended to support device configuration and management operations. Although the proposed architecture is well discussed with a use case, it lacks technical content and there is no implementation to demonstrate the proposed NDN stack.

In [Shang et al. 2016c], the authors also explore how NDN can support the IoT vision through its native features. However, the study distinguishes itself by comparing ongoing IoT implementations and design with NDN to the solutions currently used in IP. The

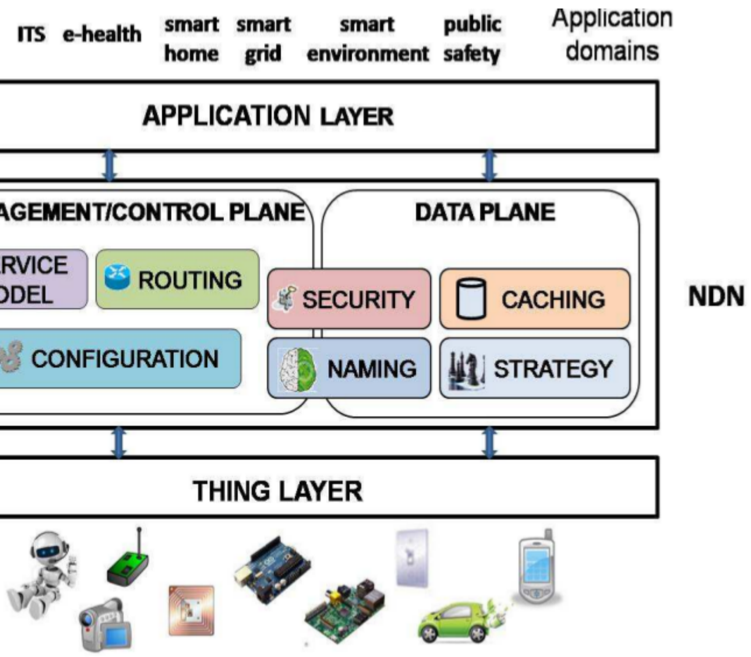


Figure 2.7: Proposed NDN-IoT architecture [Amadeo et al. 2014a]

authors also discuss various scenarios to enable the IoT over NDN. Finally they identify the following challenges for NDN in the IoT: (1) Naming with Multiple Hierarchies, (2) Routing over Infrastructure-less Environments, (3) Implementation for Highly Constrained Devices and, (4) Push-Style Data Collection.

In [Baccelli et al. 2014], the authors explore the ICN-based approach for the IoT through real-world experiments using NDN. The CCN-lite implementation on top of RIOT [Baccelli et al. 2018] is used to that purpose. The experiment is based on a deployment of 60 IoT devices distributed in different rooms, floors, and buildings. Each node is equipped with a radio chip and sensors provide temperature and humidity measurements. The advantages of using NDN are analysed and an experimental comparison with 6LoWPAN/RPL/UDP is provided. Positive results are obtained, which show that NDN can be an alternative to build an IoT architecture. The most interesting result is the comparison between the ROM and RAM sizes of the binaries compiled for NDN and 6LoWPAN/RPL stacks in the RIOT and Contiki platforms. According to those measurements, the ICN/NDN approach can significantly outperform common IoT protocols in terms of ROM size (down to 60% less) and RAM size (down to 80% less). The results

Table 2.5: CCN vs. IP: memory consumption on RIOT platform

Module	**ROM**	**RAM**
RPL+6LoWPAN	53412 bytes	27739 bytes
CCN-Lite	16628 bytes	5112 bytes

Table 2.6: Comparison of NDN, CoAP and MQTT protocols for IoT

	NDN	**CoAP**	**MQTT**
Transport	N/A	UDP	TCP UDP (MQTT-SN)
Pub/Sub	Possible [Carzaniga et al. 2011]	Possible (CoAP-Observe)	Yes
Push	Possible [Amadeo et al. 2014b]	Yes	Yes
Pull	Yes	Yes	No
Flow Control	Yes	No	Possible
Reliability	Yes	Confirmable mode	With QoS

are summarized in Table 2.5. We should note that the difference is too significant to be related to programming tricks. Rather, it mainly results from the ICN paradigm, which requires less mechanisms and no translations between names and addresses. The study also reports that even a basic NDN forwarding technique (detailed in Chapter 5) generates less overhead than RPL/UDP. However, we still need to know if the data availability and communication reliability can be complete with RPL. Nevertheless, some challenges concerning NDN solutions for IoT are also presented and discussed such as the need for packet fragmentation, header compression, and adapted caching mechanisms. This work is one of the most encouraging and inspiring for the contributions proposed in this thesis.

Regarding performance comparison between NDN and IP solutions in the IoT, the authors in [Gündogan et al. 2018a] provide a comparative measurement study between NDN and different variants of CoAP and MQTT. First, they provide a feature comparison between the three solutions, which we report and enhance in Table 2.6 with other NDN mechanisms for the IoT.

Second, extensive experimental evaluations are conducted in the study. These evaluations measure memory consumption, network utilization by control and data traffic such as protocol overhead and packet retransmissions. Communication performance is also measured through data loss, throughput at application level, and round-trip delay between issuing a request and getting a data. The IoT deployments considered in the study consist of typical class-2 devices with IEEE 802.15.4 radios, arranged in single-hop or multi-hop networks. Different traffic patterns were tested, as present in IoT applications: (i) scheduled periodic sensor readings, (ii) unscheduled and uncoordinated data updates, and (iii) on demand notifications or alerting.

The single-hop topology network consists of approximately 70 nodes, which are within the same radio range. Two arbitrary nodes are chosen to run all single-hop experiments: one is a content producer, and the other acts as a consumer. The multi-hop topology network consists of approximately 350 nodes spread evenly in a building. 50 low-end IoT device and one gateway/broker are arbitrarily chosen to run all multi-hop experiments. All low-end devices operate as content producers.

Based on the results obtained, the authors draw up the following conclusions. In single-hop topologies, the three approaches present approximately the same behaviour. However, lightweight adaptations such as MQTT-SN and CoAP Observe operate faster, and consume less energy. In multi-hop scenarios, NDN achieves better flow balancing and efficient data delivery with few packet retransmissions. Moreover, such complex scenarios quickly degrade CoAP and MQTT performance.

The NDN-ACE framework [Shang et al. 2015] describes an NDN-based control access protocol using signed Interests. Its design principle consists in allowing resource-constrained devices to use symmetric cryptography to authenticate the actuation commands. Then, key distribution and management is delegated to powerful stations such as gateways and servers, designated as Authorization Servers (AS). For example, an actuator generates a root symmetric key and shares it with the AS which can derive access keys for each client and each service according to the privileges. When an actuator receives a command Interest, it recomputes the access key and verifies the command. As NDN-ACE does not maintain secured connections, it reduces the overhead in comparison to the CoAP+DTLS solution. Moreover, a proof-of-concept implementation demonstrates the feasibility of the NDN-ACE approach. However, the proposed solutions handle only the case of sending commands such as in the lighting control case, so a more general mechanism is suitable.

V-NDN [Grassi et al. 2014] is a design proposed to enable NDN in vehicular networks

(VANETs) to provide a unified architecture. To address VANET challenges, the authors take advantage of the NDN model that decouples data naming from hosts' addresses. Consequently, a car can utilize any available interfaces to fetch data from any other nodes as soon as physical connectivity is possible. A prototype of V-NDN with around 10 cars has been implemented and tested in the UCLA Vehicular Testbed. A large-scale evaluation has been conducted through simulations.

In [Dauphin et al. 2017], NDN has been applied for networking in IoT robots. The Robot Operating System (ROS) is used to create distributed software modules that communicate with one another in a publish/subscribe fashion using named and typed data. Hence, the authors use NDN as a network primitive to support this communication model which seems to be flexible enough for this type of communication.

To push ICN to industrial IoT systems, an ICN-to-MQTT gateway has been designed in [Gündogan et al. 2017] to enable a publish-subscribe mechanism for NDN. The gateway translates NDN names to MQTT topics and a demo is implemented with RIOT and CCN-lite.

2.4.2 Forwarding

In the literature, various studies investigated the information-centric paradigm use on wireless networks such as MANETs/VANETs [Amadeo et al. 2014c; Grassi et al. 2015], and WSNs [Ren et al. 2013; Amadeo et al. 2013; Hail et al. 2015]. Some of these solutions consider ICN/NDN as an overlay on top of the IP layer while other proposals use an ICN architecture directly on top of the link layer.

In [Amadeo et al. 2014c], authors investigate the applicability of CCN in wireless networks under various constraints such as mobility and limited resources. They study the main features of wireless ad hoc networks (see Table 2.7), the applicability of CCN principles to wireless networks, the strengths and the main research challenges for CCN deployments. They identified and summarized the motivation for CCN-based wireless networks as follows:

- Node mobility is easily supported in CCN with minimal or no additional mechanism,

Table 2.7: Main features of wireless ad hoc networks

Features	**MANETs**	**VANETs**	**WSNs**
Mobility	Medium	High	MEdium to static
Battery constraints	Medium to low	No constraints	High
Storage capabilities	Medium to low	Very high	Low
Main technology	IEEE 802.11a/b/g/n	IEEE 802.11p	IEEE 802.15.4

as described above for NDN.

- CCN/NDN does not require any node identity or location knowledge to retrieve content. This can greatly facilitate the support of mobile application scenarios where only content names matter.

- Most current applications (e.g., VANETs, traffic, weather, and parking information) consist of information addressed to more than one recipient. Data can be disseminated from servers (e.g., news, weather information) or it can be shared in a group of consumers (e.g., road traffic and security) or it can be generated by single users to a group of interested recipients (e.g., social networks). All these communication models are efficiently supported through multicast and broadcast data delivery. With Interest filtering and Data caching, CCN/NDN architectures take advantage of wireless communication supports and natively support these emerging communication models.

- With a connection-less and consumer-driven model, CCN/NDN can cope well with intermittent connectivity and dynamic topologies in wireless ad hoc environments, while ensuring satisfactory data delivery and security.

The authors also discussed naming, routing/forwarding, caching, security and transport challenges to address in CCN/NDN solutions for wireless ad hoc networks. A summary is given in Table 2.8.

The main observation that came out from this survey is that most of the related literature supports CCN/NDN as a clean-slate solution (i.e., directly over the Layer 2). According to the authors, an overlay of CCN/NDN on IP should be avoided in ad hoc networks for two main reasons: (i) the end-to-end route set-up and maintenance between

Table 2.8: CCN for wireless networking: main benefits

CCN features	**Main benefits**
Naming	-Low-cost network configuration -Theoretically unbound namespace
Security	-Content-based security -No need to secure channels
Routing and Forwarding	-Lightweight node setup and maintenance -Easy multicast/multipath -Broadcast friendly
Caching	-Coping with intermittent connectivity and error prone channels -Shortening latency
Transport	Connection-less communication

overlay nodes induce high control overhead; (ii) the overlay design forces point-to-point communications, without exploiting broadcast and in-network caching.

Regarding broadcast communication with NDN, the applicability of the broadcast-based self-learning to NDN has been studied in [Shi et al. 2017]. Basically, self-learning is useful in local ad hoc networks to find packet delivery paths. A forwarder node performs this process by broadcasting the first Interest and observing where the Data packets come from. The node can then create the corresponding FIB entry so that future Interests will be forwarded in unicast or more accurate broadcast. The authors studied two main issues in using broadcast-based self-learning: (i) how a forwarder can figure out which is the prefix in the name of the returned Data packet, in order to create the FIB entry. (ii) how nodes can know whether a learned prefix-name is legitimate or malicious (i.e., sent by an attacker to perform a spoofing attack). To address these issues, solutions are proposed and summarized below with their applicability to an IoT deployment. To work out how to extract the name-prefix from the Data name, three solutions are proposed:

- Derive the prefix from the name of the Data packet by removing the last k components, where k is predefined by the application naming scheme. This solution is simple and does not require complex operations or communication overhead. We believe that it can be sufficient to support local communications (e.g., sensing) in most IoT deployments.

- Aggregate prefixes in the FIB as new prefixes are added. For example, if *"/A/B/C"*,

"/A/B/D" and *"/A/B/E"* correspond to the same next hop, they are aggregated into an *"/A/B"* entry pointing to that next hop. However, computational overhead is required to perform prefix aggregation, which can be infeasible for constrained devices.

- Producer announces its prefix explicitly. The producer can attach a prefix announcement on a Data packet sent in response to a flooded Interest. The prefix announcement is a Data packet containing the prefix and is signed by the producer. However, this solution requires longer link-layer frames, which makes it hardly applicable in low-power wireless networks such as IEEE 802.15.4.

To ensure that a prefix announcement is trustworthy, we can apply the trust model used before to provide data authenticity. Every switch/router in the network is configured with a trust model for authenticating prefix announcements. Upon receiving a prefix announcement, a forwarder can verify that: (i) the announcement has a valid signature matching the public key in the announcement signer certificate, (ii) that the announced name prefix matches the allowed prefix encoded as part of the certificate name, and (iii) that the certificate is issued by the network's certificate authority. To prevent replay attacks (i.e., reuse authentic announcements with malicious Data packets), the authors propose that every prefix announcement carries the trust model for Data packets under their announced prefix.

An important observation to make is that self-learning can be used to associate prefixes to other types of information than link-layer addresses. For example, a forwarding strategy can use self-learning broadcast to associate name prefixes to a real value (e.g., cost, round-trip time) to know if a node can forward an Interest or not.

2.4.3 Link layer

Given that current IoT wireless devices can filter a packet only by MAC address, the side effect of broadcast is that all NDN packets are processed by the CPU. This causes more load on resource constrained devices and obviously generates network overhead. Fortunately, solutions do exist to reduce the effect of broadcast transmissions while keeping its advantages, as detailed in Chapter 5.

To find an alternative to broadcast, the authors in [Kietzmann et al. 2017] investigate mapping solutions between NDN names and MAC addresses. In their study, the authors propose a mapping of names to MAC addresses to efficiently handle NDN packets, and explore different mapping schemes as follows:

1. Interest Broadcast, Data Broadcast (IBDB). All nodes will send an Interest by broadcast when a matching name prefix is found in the FIB, and send a Data by broadcast when a corresponding PIT entry is found.

2. Interest Broadcast, Data Unicast (IBDU). Similar to mapping 1, all nodes of the broadcast domain will create a PIT entry after re-broadcasting an Interest. However, each node will keep the MAC source address of the Interest, so when a Data packet is received it will be sent to the unicast source address.

3. Interest Unicast, Data Broadcast (IUDB). The FIB in this case associates each prefix name to a unicast address (next hop). Hence, when a matching name prefix is found in the FIB the Interest is sent only to the unicast address. However, Data packets are always retransmitted by broadcast.

4. Interest Unicast, Data Unicast (IUDU). Interest as well as data packets are sent to a unicast MAC address using name to address bindings in both FIB and PIT.

Note that, mappings 3 and 4 sometimes need to broadcast Interests when no information is available in the FIB. According to the authors, IBDB creates Data redundancy and provides several path possibilities, but requires more resources and generates high overhead. IBDU has Interest forwarding redundancy but reduces Data duplication as it uses unicast. IUDU consumes the lowest resources and has the lowest overhead among the four mappings. However, path redundancy and caching capabilities are reduced to the minimum due to the unicast transmissions. IUDB brings little benefit to NDN, as the unicast Interest forwarding does not exploit the redundancy of Data.

Experiments have been conducted with different network sizes and different content chunk sizes. In summary, the results attest that unicast can improve the battery lifetime of

Table 2.9: NDN and link-layer interaction approaches

Approach	Description	Standard	Reference
Adaptation layer	Additional layer between link-layer and NDN Support ACK, retransmission, etc.	802.11	[Shi and Zhang 2012]
Unicast mapping	Mapping between NDN names and MAC addresses	802.15.4	[Kietzmann et al. 2017]
Hardware-based	Name-based filtering at NIC	Ethernet	[Shi et al. 2016]
Broadcast reduction	Delayed retransmissions and packet overhearing	802.11	[Wang et al. 2012]

devices by keeping CPU-wakeups and processing overhead at a minimum, and can benefit from MAC layer ACK and retransmission. However, additional memory may be required to maintain the name-to-MAC mappings. Furthermore, unicast does not follow the NDN vision, which is designed to take advantage of the broadcast channels and data redundancy.

Other approaches have been proposed to support NDN-MAC interaction. NDNLP [Shi and Zhang 2012] is a specific link-layer for NDN designed between the NDN layer and the link layer. It supports packet fragmentation/reassembly and acknowledgment/retransmission. In [Grassi et al. 2015], the same approach is used to design a link adaptation layer for vehicular networks. A different approach consists in reducing risks of collision and packet redundancy while using broadcast (i.e., IBDB). To do so, delayed retransmissions and packet overhearing are used, but the overhead is still high and not acceptable in IoT environments as simulations show in Chapter 5. Another approach proposes to modify the device driver of the NIC to support frame filtering based on the names rather than MAC addresses. Good performances are achieved but the solution implies re-engineering a part of the hardware, which makes it unusable with current popular IoT devices. We have summarized these approaches in Table 2.9.

2.4.4 Mathematical models

Most of the models on ICN/NDN are conducted exclusively around caching, such as cache deployment, cache decision and cache replacement. Other studies are devoted to ICN/NDN transport and routing performance, often with comparison to TCP/IP, such as in [G. et al. 2013].

However, very few analytical models for ICN/NDN consider modeling networks of caches and the interaction between caching and transport; those models consider traditional wired networks [Guo-qing et al. 2013; Yongmao et al. 2017; A et al. 2013]. In the same context, some studies have been presented in the context of Web caching with ICN under LRU replacement policy. One of these studies [G. et al. 2011] is adapted in our model presented in Chapter 4 to compute cache miss rate probabilities.

To the best of our knowledge, no model has been formulated on NDN in wireless networks with cache consideration, whether for constrained or traditional wireless networks.

2.4.5 Comparing NDN and IP

Ideally, the improvement that can be provided by NDN in the IoT (and Internet in general) should be demonstrated by direct comparisons between NDN and IP protocols (e.g., IP, TCP/UDP, HTTP, CoAP, MQTT). Regarding the studies mentioned before that propose direct experiments to compare NDN and IP, we argue that such comparisons are difficult to carry for several reasons.

First, NDN and IP solutions are based on different paradigms which makes it difficult to fit their respective communication models in one fair scenario. For example, a great attention should be given to URL and name length as they highly impact processing and communication performance. However, names and URLs are not processed at the same level in NDN and IP. Moreover, the size ratio between a request and a response is not the same in NDN and IP. Another example related to networking paradigm difference is the security. Indeed, if the security overhead is not considered in the comparison, it should be ignored in NDN, which is not fair as it is one of NDN design principles that requires more processing time. If the security aspect is considered, it is difficult to accurately estimate the overhead of each approach, given that IP protocols use secured sessions whereas NDN uses packet signature. Moreover, key and certificate distribution is an important part of NDN communication, thus it should be often considered as network overhead.

Second, NDN natively benefits from features that clearly increase its performance, such as caching that improves data delivery efficiency. On the other side, the IP stack can support almost any feature at the application layer such as caching and object-based security as mentioned in Chapter 1. Consequently, in-network caching considerably increases NDN performance, but one may argue that caching can also be implemented with IP, which then provides IP with similar performance. However, the difference resides in how caching (or another feature) is supported in NDN and IP, and how much complexity is required to support caching (or another feature). We believe that, this kind of comparison is mainly related to common sens and empirical estimation of the required effort. Another point of comparison is implementation complexity and required memory in IoT environments. All the studies that carry such comparisons, as mentioned before, report that NDN implementations are much lighter than IP implementations. However, there is no guarantee that future improvements of NDN, such as security, will keep providing this advantage for NDN. Moreover, NDN is currently an academic project, which means that it does not benefit yet from optimized implementations that industry usually provides as it is the case for IP protocols for example.

Generally speaking, the most common comparison provided in the literature is related to the features such as security and caching, and how they are supported by each of NDN and IP.

2.4.6 Projects

2.4.6.1 Platforms/Deployments

- NDN Forwarding Daemon (NFD) [NDN a] is a software module that implements the NDN forwarding mechanisms and its features. It evolves together with the NDN protocol and allows NDN to be experimented in the real world. The first release was made by the NDN team project but contributions from the broad community are currently included.

- NDNoT project [NDN b] is a toolkit that allows the development of smart home networks. It runs on Raspberry Pi and supports sensing/actuating functions via GPIO. It provides functionalities such as: (i) adding and removing devices and services, (ii) managing control access, and (iii) collecting data and sending commands. The platform proposes an on-boarding mechanism to authenticate and add new devices to the home network. The hierarchical structure of NDN names is exploited and signed Interests are proposed to authenticate commands. The toolkit includes a sample application that can switch a connected television on and off depending on room occupancy.

- NDN-BMS [Shang et al. 2014] is an NDN secured Building Monitoring System deployed at UCLA, and uses its existing building monitoring system. This system supports collecting data from sensors through gateways and publishing it with a secured control access based on NDN names and security mechanisms. It uses a naming scheme that expresses data's physical types (voltage, light, etc.) according to their geographical provenance (building, floor, room, etc.) and its temporal aspect (date, time) giving an easy and human-friendly way to retrieve the desired data. The same idea can be used as a naming scheme in our cow monitoring scenario: the prefix *"/farm/area/1/room/1/cow/22"* covers the data generated by cow 22 located in room 1 in area 1. A Data packet can append more components to express more information about the content such as *".../temp/201902070850"* for the temperature measured at *8:50* a.m. on February 7, 2019. NDN-BMS security is based on data encryption and access control through public key distribution and privilege management. However, device/gateway communications use legacy protocols while only data publishing and access control are NDN-based.

- The NDN protocol stack has been ported to the RIOT platform with NDN-RIOT [Shang et al. 2016b]. RIOT is a lightweight operating system designed for constrained IoT devices. It includes drivers for Ethernet and IEEE 802.15.4 network interfaces. It also supports IoT-related network protocols such as IPv6, UDP, 6LoWPAN, RPL and CoAP. The NDN-RIOT initial implementation provides a basic support of the IEEE802.15.4 using broadcast communication with a simple packet fragmentation and reassembly mechanism. The implementation provides a high-level application interface with data security support and shows the feasibility of porting NDN on constrained devices.

2.4.6.2 Libraries/Frameworks

- NDN-CCL [NDN c] is a set of libraries for developing NDN applications with several languages. The languages currently supported are C++ (NDN-CPP), Python (PyNDN2), Java (jNDN), JavaScript (NDN-JS) and .NET (NDN-DOT-NET). The libraries basically provide implementation for NDN entities and concepts such as the Name, Interest, Data, Face, etc.

- A collection of networking tools is available for NDN [NDN d]. These tools provide monitoring and debugging for NDN networks and essentially resemble IP-based networking tools. The collection is called ndn-tool and includes the following:

  - peek: transmit a single packet between a consumer and a producer
  - chunks: segmented file transfer between a consumer and producer
  - ping: test reachability between two nodes
  - dump: analyze traffic on wire
  - dissect: inspect TLV structure of NDN packet format
  - dissect-wireshark: Wireshark extension to inspect TLV structure of NDN packets
  - pib: a service to manage the public information of keys and publish certificates

- A lightweight version of the NDN-CPP library has been developed to support resource-constrained platforms such as Arduino. This library eliminates dynamic data structures of PIT, CS and FIB to provide NDN applications that fit Arduino constrained resources devices.

2.4.6.3 Simulation/Emulation

- The NDN research testbed [NDN e] is a shared resource created for research purposes, that includes software routers at several participating institutions, application host nodes, and other devices. An emulated NDN testbed that runs in the Open Network Laboratory (ONL) is available to evaluate NDN applications. It runs on real servers and uses the same software as the testbed (i.e., NFD and NLSR). This allows testing, debugging, and evaluating applications quickly and without any risk of disturbing the NDN testbed.

- ndnSIM [Afanasyev et al. 2012] is the official NDN simulator. It is based on ns-3 and implements the NDN model as a network layer protocol which can run either on top of link layer, network-layer or transport-layer protocols. The implementation started in 2011 and the first release has been available since June 2012. ndnSIM is widely used by the NDN research community.

- Mini-NDN [NDN f] is a lightweight networking emulation tool that enables testing, experimentation, and research on the NDN platform. Based on Mini-CCNx, which is a fork of Mininet, Mini-NDN uses the NDN libraries, NFD, NLSR, and tools released by the NDN project to emulate an NDN network on a single system.

2.5 Conclusion

Despite the various studies related to NDN for IoT, no clear ICN/NDN development path currently exists that could be used to show NDN's superiority over IP. In practice, fundamental differences such as caching and naming data make it difficult to provide fair direct comparisons between NDN and IP. Therefore, we aim to provide a multi-level environment that helps to study the benefits of NDN in current IoT deployments.

Rather than discussing global NDN challenges for the future Internet that may include the IoT only as a part, we choose to study the feasibility of an IoT deployment with NDN in a typical scenario with popular IoT equipment (e.g., Arduino) and applications. Therefore, the first step is to design and deploy a realistic NDN-based architecture which highlights the main challenges that should to be addressed by NDN. In the next chapter, we propose such an architecture, and specify its components and the main challenges it raises.

Chapter 3

A Realistic NDN Architecture for the IoT

3.1 Introduction

It appears through the previous chapters that NDN can be more suitable to build IoT systems than IP. Its communication model does not require issuing and managing device addresses, and operates directly on the application's named content. It secures content regardless of transport protocols, sessions or channels. This provides reusable secured Data packets and gives NDN a native support of caching, broadcast and multicast. Moreover, NDN does not have a predefined packet format or minimum MTU requirements, and its simplicity produces implementations with smaller code size in the devices as shown previously.

However, the practical deployment of NDN must be defined to take advantage of these features in current IoT solutions. Indeed, the integration of NDN in the existing Internet infrastructure is vital and it will impact many networking and application aspects. Currently, a global NDN deployment is not feasible due to the difference between IP and NDN paradigms. In practice, to deploy the NDN protocol in IP networks, hosts and routers need at least to support name-based routing, packet processing, and implement some forwarding strategies. Moreover, short-term solutions require the coexistence of NDN and IP in the same global network. For that, we must ensure that NDN and IP devices do not interfere with each other, while guaranteeing that such deployment will lead to increasing benefits for applications.

Bearing this in mind, the integration of NDN as it is envisioned in this dissertation strives to be realistic and incremental. This means that, we propose an NDN-based design for IoT that can co-exist with the IP infrastructure and current IoT equipment. In this way, we aim to make NDN easily accessible by enabling it on the "thing" side of the IoT. In other words, we provide low-end IoT devices with a Layer-3 data-centric identity which we believe is more natural than the current IP identity. This requires the integration of NDN with current IP-based infrastructure through a realistic architecture.

This chapter draws the picture of a real-world NDN integration in the IoT. We start by identifying and discussing possible NDN integration approaches according to related work. Then, a realistic deployment approach with pragmatic NDN-specific operations is proposed to enable NDN in constrained devices. Finally, we identify the main features to support in such an architecture with some potential solutions.

3.2 NDN Integration Approaches

At the network layer, an NDN deployment requires network entities to support name-based routing, packet processing, and implement some forwarding strategies and security procedures. In addition, more storage is needed for caching and stateful forwarding.

Considering the global IP network infrastructure, NDN can be deployed as an overlay over IP, it can replace IP as a native network protocol over the link layer (e.g., NDN over Ethernet), or IP and NDN can coexist in the same network. The first approach, the overlay, is easy to deploy and creates a uniform content-centric layer. The NDN testbed [NDN e] is an example of such approach. However, this solution creates complexity and overhead for the underlying network protocol, and IP-based applications must switch to NDN in order to use the network. Moreover, the overlay approach considers NDN as a transport/application protocol for IP, and thus does not provide a coexistence between the two network protocols (i.e., IP and NDN). More importantly, implementing both NDN and IP stacks is not feasible with IoT constrained devices that can barely support the current IP stack. The second approach, deploying NDN as a native network protocol, works only for environments that do not need to communicate with global IP networks, such as isolated vehicular networks or local networks. The third and last approach is to make IP and NDN coexist within the global network. This approach may either use NDN at the core and keep IP at the edge of the network (NDN-core), or deploy NDN at the edge and keep IP networking at the core (NDN-edge).

With an NDN-core approach, IP applications do not need to be changed at all, but a global deployment of NDN as a native network protocol is currently not feasible, as previously mentioned. As an exceptional example, the POINT project [Xylomenos et al. 2018] had to work with ISPs to deploy a real-world prototype in which an ICN architecture is used at the core of the network. The prototype then introduces ICN in the core network without changing the rest (i.e., the edge) of the Internet.

With an NDN-edge approach, the core network keeps running IP, while applications and devices run native NDN. This solution is easy to deploy and does not require deep changes in the infrastructure. Moreover, it provides a progressive integration of NDN.

In both NDN-core and NDN-edge, the coexistence of IP and NDN can be achieved by using peripheral nodes such as gateways to translate between NDN names and IP protocol stack information. For example, Cisco's hICN [Muscariello et al. 2018] encodes names as IPv6 addresses to allow hICN packets to be processed by both ICN-based and IP-based routers, and Zhang et. al. [Wu et al. 2017] proposed a dual-stack scheme for NDN switches and IP switches to coexist in local area networks.

At application level, when NDN and IP stacks have to coexist together, NDN requires completely different mechanisms than IP-based applications, and vice-versa. As the purpose is to make the most of NDN while providing reasonable solutions, we identify two possible translation approaches between NDN and IP as described in [Liang et al. 2018]:

- The first solution is to provide a translation between TCP/IP or UDP/IP and NDN. The advantage of this approach is that it supports various application protocols with the same transport-level translation. However, as network and transport layers in the IP stack have limited expressiveness, some NDN features will not be exploited. For example, translating a TCP packet into an NDN packet may use information from

TCP/IP headers to generate NDN names. That is, the name will be associated to the specific TCP connection. This provides benefits such as caching within the same TCP connection (e.g., efficient retransmission of lost packets), but cannot support caching across different TCP connections (e.g., multicast to different consumers). Moreover, data-centric security of NDN will be limited as names are still related to hosts and connections.

- The other approach consists in translating between application-level protocols such as HTTP to NDN. Application-level information is much more expressive and data-oriented than network and transport information. Thus, NDN names generated from HTTP headers will be more meaningful regardless of hosts and connections. This allows this approach to make much more of NDN benefits than the previous one.

Furthermore, to avoid translation, a hybrid deployment can be adopted combining NDN-edge with NDN-overlay approaches to achieve the maximum possible integration of NDN. This combination is realistic since low-end IoT devices implement only NDN and core network equipment has enough resources to support NDN over IP even with some additional overhead. The NDN-802.15.4 architecture we propose is based on this hybrid solution and is discussed below. Integration approaches discussed above are summarized in Figure 3.1.

3.3 Proposed NDN-802.15.4 architecture

This section describes the realistic NDN architecture we envision for the IoT. After studying the integration possibilities, we chose the NDN-edge approach combined with an NDN over UDP/IP in the core network. Our motivation for that is explained below, followed by an overview of the possible wireless technologies in the IoT and our choice. Then, the architecture, its components and specific mechanisms are described.

3.3.1 Adopted Integration Approach

When applied to the IoT, the NDN-edge integration corresponds to the deployment of NDN in low-end IoT. In other words, NDN is used where the content is produced and

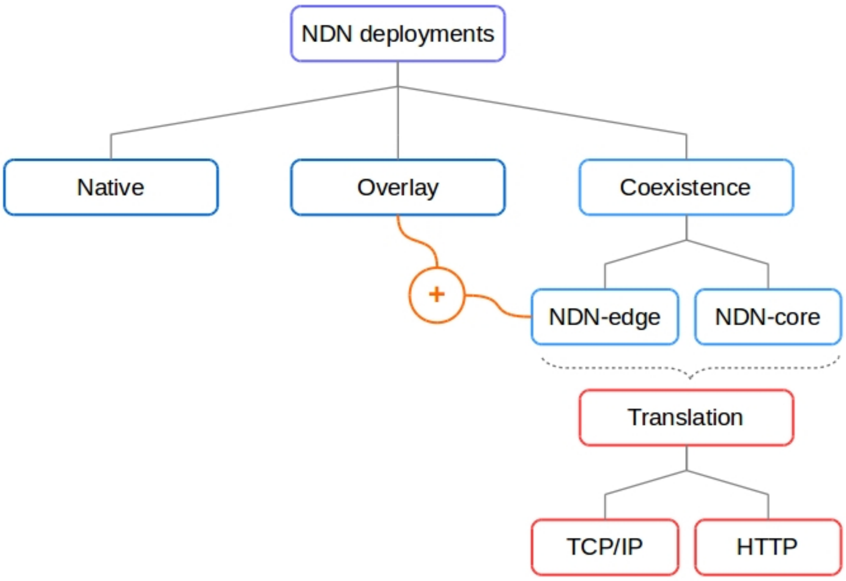


Figure 3.1: NDN integration approaches

consumed. On the one hand, IoT devices run native NDN applications over a wireless link-layer technology. On the other hand, using NDN over IP-based transport protocols such as UDP allows applications on computers and smartphones to communicate with IoT devices via NDN. In addition to being completely feasible in the current Internet infrastructure, this approach takes advantage of all NDN features such as naming content, data-centric security, and caching. Moreover, as IoT design is in its early stages, particularly at low-end IoT, this approach is a reasonable starting point to create NDN-capable devices together with NDN native applications without wasting time. In addition, integrating NDN from the edge of the network supports a progressive and incremental integration. Experience gained from local deployments will lead to a stronger NDN architecture and various possibilities can be envisioned for the long term.

Furthermore, most IoT applications rely on the Internet to reach cloud servers. However, there are scenarios when Internet connectivity is not available but local network connectivity exists, such as in smart agriculture deployments. Typically, NDN applications can discover names and exchange data between two locally connected devices without going through the Cloud.

Table 3.1: Most common wireless technologies in the IoT

Technology	**Frequency**	**Data Rate**	**Range**	**Power Usage**	**Cost**
2G/3G	Cellular Bands	10 Mbps	Several Miles	High	High
Bluetooth/BLE	2.4 Ghz	1, 2, 3 Mbps	300 Feet	Low	Low
IEEE 802.15.4	subGhz, 2.4 Ghz	40, 250 Kbps	> 100 Square Miles	Low	Low
LoRa	subGhz	< 50 Kbps	1-3 Miles	Low	Medium
LTE Cat 0/1	Cellular Bands	1-10 Mbps	Several Miles	Medium	High
NB-IoT	Cellular Bands	0.1-1 Mbps	Several Miles	Medium	High
SigFox	subGhz	< 1 Kbps	Several Miles	Low	Medium
Wi-Fi	subGhz, 2.4/5 Ghz	0.1-54 Mbps	< 300 Feet	Medium	Low

3.3.2 Wireless Technology

The IEEE 802 Standard is a set of networking standards for both wired and wireless networks. The most well-known wireless specifications include 802.11 [IEEE 2016] (e.g., WiFi), 802.15.4 [IEEE 2011] (e.g., ZigBee) and 802.15.1 [IEEE 2005] (e.g., Bluetooth). Due to its satisfactory bandwidth and admissible cost, WiFi nicely meets LAN requirements and is widely used in businesses and homes. However, IoT local networks focus on other aspects such as low power consumption, large numbers of nodes and long range communication. Although communication solutions for IoT do not generally require a large bandwidth, they need an efficient power management plan, a low cost of production and must support a large number of (mobile) nodes in a simple way. To support that, many physical and link-layer specifications exist. For example, Bluetooth Low Energy (BLE) and ZigBee are designed for wireless personal area networks (WPANs) and allow satisfactory data rates with low-power consumption and reasonable complexity. Other communication specifications are available for specialized networks, such as WAVE [IEEE 2019] for VANETs and 3G/4G for very long distances. More recently, new wireless technologies explicitly designed for the IoT have appeared, such as Sigfox [Sigfox] and LoRa [LoRa Alliance]. However, these technologies are still expensive for customers in comparison to ZigBee and BLE. To summarize, Figure 3.1 gives a comparison of the relevant wireless technologies involved in IoT deployments, according to important evaluation criteria such as power usage and cost.

Among these wireless technologies, two popular ones appear to offer a satisfactory compromise between range, power consumption and cost. For this reason, they are currently dominating IoT systems: Bluetooth Low Energy (BLE) and IEEE 802.15.4. Both are low-power low-rate technologies. They operate in the 2.4 GHz ISM spectrum, but have their own modulation scheme, bit rate, channel map and channel spacing, and upper layers. In the following, we provide a description of these technologies, followed by a brief performance comparison adapted from [Narendra et al. 2016].

BLE. This technology uses frequency hopping over 37 channels for bidirectional communication and 3 for unidirectional advertising, with a bitrate of 1 Mbps. In Bluetooth 4.0, the link-layer MTU is 27 bytes, increased to 251 bytes in Bluetooth 4.2. Frames are protected with a 24-bit CRC.

BLE networks form a star topology, with a master orchestrating bidirectional communication with one or several slaves. Nodes have different link-layer states including advertising, scanning and connection. An advertising device (typically a low-power node) periodically broadcasts packets over channels 37, 38, 39 which are spread over the 2.4 GHz ISM spectrum so they do not overlap the most common WiFi channels 1, 6 and 11. A scanning device (e.g. a smartphone) listens on these advertising channels, waiting for advertisement packets. Upon receiving an advertisement, the scanning device may initiate a connection with a ‘connection request’ packet. The advertising device becomes the slave and the scanning device the master. To manage connections, BLE uses a Connection Interval parameter, which is the interval between connection events. It ranges from 7.5 ms to 4 s, and may change after connection has been established. Another parameter is Slave Latency which defines how many connection events a slave device can skip in a row. Skipping connection events is used by slave devices to save energy.

IEEE 802.15.4. This technology uses 27 non-overlapping channels, including 16 in the 2.4 GHz and 11 in the sub-GHz bands. The 2.4 GHz band has a bitrate of 250 kbps. The MTU is typically 127 bytes, and frames are protected with a 16-bit CRC.

IEEE 802.15.4 networks support star, cluster tree and mesh topologies. IEEE 802.15.4 is widely used in the research area and features many different MAC layers such as Carrier Sense Multiple Access (CSMA), and Time Slotted Channel Hopping (TSCH) [Watteyne et al. 2015]. Some are part of the standard, others research prototypes.

With CSMA, nodes keep their radio always on, operate on a single channel, and access the medium through a contention algorithm, CSMA, in a slotted or unslotted mode. In unicast transmissions, link-layer acknowledgments are used to confirm reception and enable retransmissions. The CSMA is one of the MAC layers defined in IEEE 802.15.4-2011.

With TSCH, all the nodes are globally synchronized and form a mesh network. The communication is slotted and each slot is long enough for the transmission of a frame and its acknowledgment (typically 10 or 15 ms). Slots are grouped into one or several slot-frames, which repeat over time and form a schedule. At every slot, the nodes know exactly whether they are supposed to sleep, transmit or receive, and deterministically select a channel to use from a pseudo-random hopping sequence. This improves interference resilience and supports link dynamics. As IEEE 802.15.4 includes only physical and MAC layer, many upper layers are possible to run above IEEE 802.15.4 such as ZigBee and 6LoWPAN. The IETF Working Group 6TiSCH [Thubert 2019] is proposing an architecture for 6LoWPAN/TSCH networks.

Comparison. Unlike IEEE 802.15.4, which is restricted to the physical and MAC layers, BLE is a full protocol stack. It is thus potentially more complex to manage.

To compare the performances of BLE and IEEE 802.15.4, we report on experiments conducted in [Narendra et al. 2016] focusing on latency, data rate, reliability, and energy consumption. In all the experiments two nodes are communicating; a master and a slave for BLE, a coordinator and simple node for IEEE 802.15.4. In all the experiments, the nodes use a transmission power of 0 dBm. The reported metrics are the following:

- Latency: the time taken in the application layer to get data in a request-response communication.
- Data Rate: the amount of link-layer payload per unit of time.
- Energy: time spent with the radio turned on, referred to as Radio Duty Cycle (RDC).
- Reliability: the number of packets received over the number of packets sent at the link-layer, referred to as Packet Reception Ratio (PRR).

The first experiment is a request-response communication to measure the latency for fetching data from another node. Nodes are at a distance of 10 cm from each other and the request-response cycle is repeated 1000 times for each test.

According to the measures, the lowest latency is achieved with BLE with a connection interval of 7.5 ms, but this requires a high duty cycle of above 20%. The second lowest latency is achieved by 802.15.4-CSMA, but with a duty cycle of 100%, which increases energy consumption. Both BLE and 802.15.4-CSMA achieved a latency under 50 ms. 802.15.4-TSCH and BLE with a connection interval of 125 ms achieved an interesting latency-energy balance, with a duty cycle between 0.6 and 1.3%, and a latency between 100 and 200 ms. The lowest energy consumption is achieved by BLE with a slave latency of 65, at the cost of a very high latency of 750 ms.

The second experiment is a bulk-transmission process to compare the maximum data rate with and without WiFi interference. Nodes are placed at a distance of 1m and a WiFi router is placed 2.5m away from the two nodes to generate interference. The maximum payload allowed by each link-layer is used; 27 for BLE and 110 for IEEE 802.15.4. Each run of the experiment lasts one minute in order to measure a stable mean data rate. The metrics measured in this experiment are data rate, energy, and reliability.

The results show that 802.15.4-CSMA achieves the highest data rate with 155 kbps, but with a higher energy consumption as it keeps the radio turned on almost all the time. BLE achieved a PRR of 99.9%, due to channel hopping and short frames. The same behaviour is observed with and without WiFi interference. IEEE 802.15.4 has a higher data rate and consumes more energy. TSCH and BLE, through channel hopping, are less affected by interference than the single-channel CSMA. With the help of link-layer retransmissions, both BLE and TSCH achieved 100% reliability.

Overall, both IEEE 802.15.4 and BLE are suitable for IoT deployments. They can both achieve latency-energy trade-offs. Furthermore, Siekkinen et al. [Siekkinen et al. 2012] focused on the relation between throughput and energy, and found that BLE had a constant energy utility, while IEEE 802.15.4 became more energy-efficient as throughput increased. However, IEEE 802.15.4 defines only the physical and MAC layers, and natively supports a mesh network topology. Thus, it allows more flexibility than BLE while providing satisfactory performance.

Consequently, we adopt the IEEE 802.15.4 technology in our architecture design. In the remainder of this chapter, the distinction between link-layer protocols used in IEEE 802.15.4 is not necessary as our design assumes the IEEE 802.15.4 features independently of the underlying MAC protocol. However, in the next chapters, when a specific link-layer has to be considered, we assume the CSMA link-layer which is the most common.

Some studies have been dedicated to NDN communication over Bluetooth. In [Attam and Moiseenkoy 2013], the authors propose to fit NDN model into constrained Bluetooth stack. They design a proxy layer that provides NDN connectivity over Bluetooth. At the time of writing this manuscript, NDN over Bluetooth Low Energy (BLE) is still under investigation. Recently, the authors in [Petersen et al. 2019a] demonstrate an NDN-over-BLE implementation on a multihop network. They design a mapping of NDN to BLE primitives inspired from the concepts of IPv6-over-BLE. In a related study [Petersen et al. 2019b], the authors compare Bluetooth mesh networking and ICN, conceptually and through real-world experiments. The objective is to investigate how much Bluetooth mesh has in common with ICN principles. The authors identify major differences, such as the fact that Bluetooth mesh uses flooding without caching mechanisms, which is a native ICN principle to add in future versions. Another interesting conclusion from reported experiments is that, according to the authors, NDN can utilize network resources more efficiently than Bluetooth mesh stack.

3.3.3 Communication Architecture

The first benefit of integrating NDN with IEEE 802.15.4 is to make IoT end-devices an integral part of the NDN network, whether they are producers or consumers. Therefore, sending Interest and Data packets over IEEE 802.15.4 frames is not the only operation required. An efficient integration should also consider packet transformation at the intersection between the wireless local network and the backbone, to accommodate constrained devices while making the best use of their resources. Bearing this in mind, we designed the NDN-802.15.4 architecture detailed below.

We consider a local wireless mesh network formed by a set of various IoT devices (e.g., sensors). This local network is connected to the Internet via a gateway. Typically, the gateway forwards Interests issued from user applications, whereupon nodes can reply with Data packets that they produced or stored in their cache. The wireless link between devices, including the gateway, is based on IEEE 802.15.4.

We assume that each device is a source of data and thus is a content producer, while the gateway forwards Interests issued from applications and thus acts as a local consumer. Each device generates data under a specific prefix name obtained together with the required security materials (e.g., signature keys, certificates) through a pairing process. Each device has a different content name but a common prefix is shared between devices and the gateway within the local network.

As we envision a realistic integration, we adopt a typical IoT architecture in which native NDN is integrated in the edge (i.e. gateway-device) and NDN over UDP/IP in the backbone. A Wireless Local Area Network (WLAN) is formed by a gateway and a set of devices. Each WLAN is accessible via the gateway under a Common Prefix (CP). To sum up, architecture's components are the following:

1. End-devices (EDs): Wireless nodes running the NDN protocol and communicating through an IEEE 802.15.4 transceiver.

2. Local Manager (LM): LM's role is to manage device identities, access control, etc. Thus, it can handle a pairing process for EDs, an authentication server, etc. LM is a software component typically included in the gateway.

3. WLAN: A gateway, including the LM, and a set of EDs form a wireless local network accessible via a common prefix (CP).

Figure 3.2 illustrates an example of our architecture deployed in the cow monitoring scenario described in Chapter 1.

This architecture resembles most usual IoT deployments; with the same equipment, technologies and entities. Only networking is different as it is based on NDN. To provide a better support of NDN over IEEE 802.15.4, we include two mechanisms to control packets size. These mechanisms do not represent an additional layer or middleware for NDN as it is common in IP. Rather, they exploit the flexibility of NDN to better suit the constraints

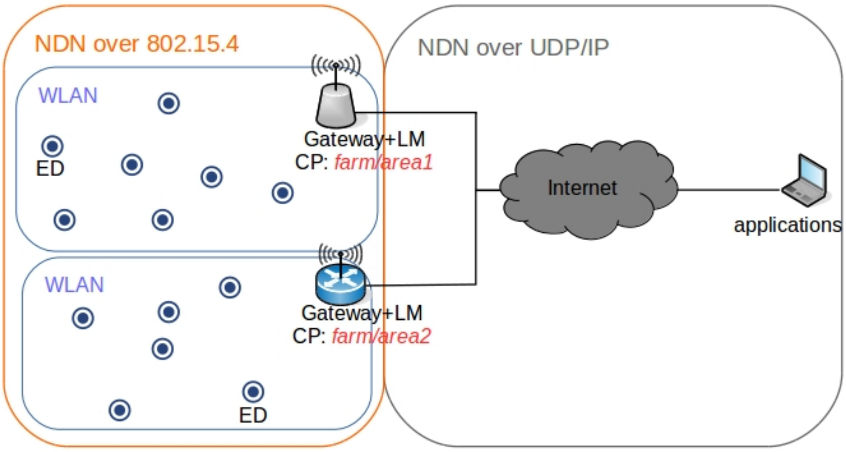


Figure 3.2: NDN-802.15.4 architecture

of IoT devices, such as small MTU, limited memory and limited CPU. The proposed operations are implemented in the EDs and the gateway. However, EDs do not necessarily include the same operations as the gateway; while the latter can support more complex operations, the former only implements the basic ones.

These mechanisms are presented below and for comparison, Figure 3.3 depicts the NDN protocol stack with IEEE 802.15.4 integration, a simplified OSI model and the 6LoWPAN stack.

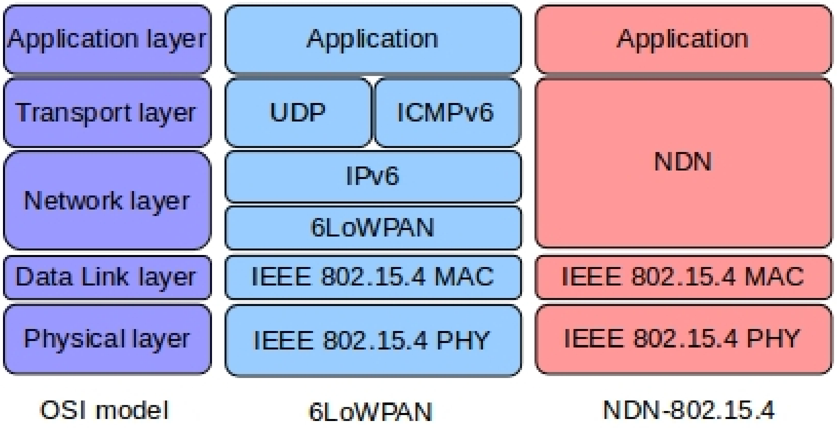


Figure 3.3: NDN-802.15.4, OSI model and 6LoWPAN stack

3.3.4 Integration Mechanisms

In the following, we detail the two operations designed to manage NDN communication over IEEE 802.15.4: Name-Payload-Field size control, and stateless packet compression scheme.

3.3.4.1 Name-Payload-Field size control

So far, propositions to support NDN in constrained environments have been based on predefined restrictions. For example, a lightweight version of CCN has been designed for WSNs in [Ren et al. 2013], in which restrictions are imposed on name length and packet fields to control packet sizes. However, excluding some fields and limiting name length arbitrarily or intuitively is not suitable to cover all use cases. An application may require more expressive names but sends small amounts of data, while another might require accurate data information, achieved through packet fields, but uses short names and small content. To illustrate that, two application scenarios can be given: (1) A sensor network that needs to name small-sized data according to the location, the timestamp, the data type and the version, requires maximum name length at the expense of the payload. (2) A livestock monitoring system that needs to identify each animal individually with all its related information may name data according to an ID and timestamp, but a larger payload is required for the content.

As we can observe, name length in NDN directly impacts Interest length, Data length and thus payload size. Moreover, name length impact becomes more critical in Data packets as they must carry signatures and content, thus they are generally much bigger than Interest packets.

To efficiently manage the small frames of IEEE 802.15.4, it is useful to control Data packet structures since no predefined size and format exist. We believe that it is important to understand the proportion in length between name, payload and fields in a Data packet. Therefore, we propose a Name-Payload-Field balancing function to control the size of each part.

The following notation is adopted to describe the function:

- F: payload size of the frame considered (e.g. $F = 100$ bytes)
- d: length of the Data packet with the defined fields and all necessary Type-Length bytes. Here, the fields include all Data packet fields, except Name and Content since they represent the name of the Data and its payload respectively.
- p: payload length of the Data packet
- s: length of the signature included in the Data packet
- $f(p)$: size allowed for the Name according to the payload

Here, a Payload-Name relationship, $f(p)$, can be defined by subtracting from the frame length the sum of the Data packet parts; which are the payload, the signature and the rest of the Data packet fields. This can be intuitively evaluated as:

$$f(p) = F - (p + d + s) \tag{3.1}$$

However, d changes dynamically according to the number of fields defined in the Data packet and their types. Moreover, the relationship between defined fields and data size (d) is not constant since each application is free to define the fields it needs, and they do not have the same length.

To find a relation between the structure of the Data packet and its size d, we want to derive a function that gives an estimation of a Data packet size according to the number of fields it includes, independently of their types. For that, we study possible field combinations and their corresponding sizes according to the following approach:

1. We consider only the fields that can directly contain a value, a Name or a Name-Component field. This gives the set :

   $\Omega = ContentType, FreshnessPeriod, FinalBlockId, SignatureType, KeyLocator.$

2. Since each field from om can be defined only once in a packet, the number of combinations C is as follows: $C = \binom{N}{k}$, where $N = |\Omega|$ and $k = \overline{0, 5}$.

3. For every possible field combination, we calculate the size of the corresponding Data packet including all necessary bytes, except Name and Content fields. For the combinations that have the same number of fields, their average size is calculated.

Since the focus is on small-sized frames (i.e. $\leq$ 100 Bytes), we expect reasonable field lengths so that packets can fit in IEEE 802.15.4 frames. For security-related fields (i.e. SignatureInfo and SignatureValue), we consider a symmetric cryptography with HMAC authentication. Although Type-Length bytes of KeyLocator field are included in the Data structure size, its actual value is not included. Since KeyLocator contains a Name, it is up to the application to compute its length and subtract it from Name length. Figure 3.4a (square marker) gives the average Data size d according to the number of fields included.

To verify the accuracy of this approach, we calculate the Pearson correlation coefficient r:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{3.2}$$

Where $\bar{x}$ and $\bar{y}$ are the means of x and y values respectively.

The correlation coefficient between field number x and data size y is $r = 0.9964$. That is, an accurate linear regression relation can be calculated as follows:

$$d(x) = \alpha + \beta x \tag{3.3}$$

Where $\beta = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$ and $\alpha = \bar{y} - \beta\bar{x}$.

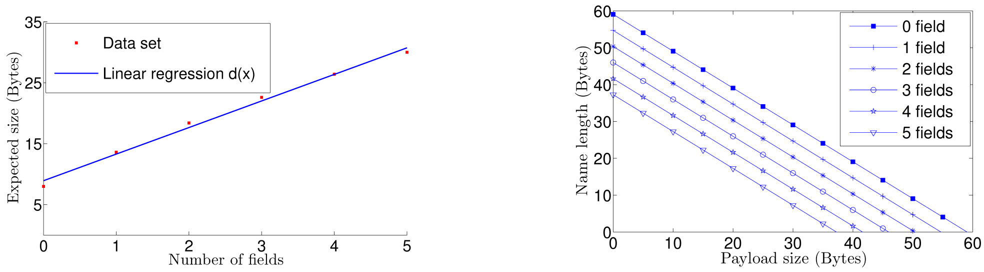
Calculated α and β are 4.36 and 8.93 respectively. Figure 3.4a (plain line) gives the linear regression relation $d(x)$ between the number of fields x and data size d.

With this linear regression, the Payload-Name relationship defined in Equation 3.1 can be improved to Payload-Name-Field relationship, expressed as follows:

$$f(p) = F - (p + (8.93 + 4.36m) + s) \tag{3.4}$$

where, m is the number of fields included in the Data packet.

Figure 3.4b gives the payload-name relation according to the number of fields defined in the Data packet, considering an HMAC signature (i.e. = 32 Bytes).



(a) Expected Data packet size according to the number of fields

(b) Allowed Name length according to the payload and the number of fields

Figure 3.4: Name-Payload-Fields estimations

As a result, the $f(p)$ function estimates the size allowed for the Data packet name according to the payload length and the structure of the packet. However, it is not intended to be used to calculate the sizes of Data packets on creation. Rather, $f(p)$ should be used as a dynamic guider for EDs and/or the gateway to set the appropriate naming scheme for the content, while balancing with payload size and packet fields. Moreover, the proposed function is general and assumes all existing fields, but other field assumptions can be envisioned to find a more accurate Name-Payload-Field balancing function.

3.3.4.2 Stateless packet compression

Interest packets issued by applications can rapidly become large, due to long names and optional fields. The corresponding Data will be even larger due to content and signature. However, we observe that requested content can be retrieved from EDs without sending all Interest fields. Missing fields can be computed based on predefined configurations, the packet specification, and previously shared information. Following the same principle, Data packets produced by EDs can be sent with missing fields that will be completed by the gateway. To exploit this feature, we designed a simple packet compression scheme to reduce overhead in the WLAN and avoid packet fragmentation.

As mentioned in Chapter 1, IP stack already uses header compression whose main benefit is to reduce packet sizes sent over WLANs and thus avoid fragmentation. Moreover, sending small packets reduces the probability of packet loss caused by bit errors on wireless links. To give an idea of the packet compression approach, we refer to the header compression scheme used in IP, particularly 6LoWPAN.

The IP protocol stack relies on packet headers to describe and manage communications through multiple hops over the network. Typically, headers carry source and destination addresses, port numbers, protocol versions, sequence numbers and other flags. However, most of this information does not change over time, or changes in specific patterns such as by incrementation. Those fields that keep the same value or change in a predictable way, do not need to be sent in each packet, or they can be represented with a smaller number of bits than their actual value. In short, the underlying principle of header compression is to reduce not only the redundancy of the information contained in a packet header, but also the redundancy between several consecutive headers. Thus, information that does not change is sent only at the beginning of the session or at a slow pace. For the changing fields, a prediction or dependency mechanism makes it possible to further reduce the amount of information transmitted.

Header compression mechanisms use context to store some header values and redundant information in the data stream. The compressor and decompressor have a copy of the context with the information of the last header. Context is refreshed each time a value in the header changes.

In addition to the context, header compression mechanisms usually classify the fields of the headers. The analysis for the classification is based on how the values of these fields change during the connection. The classification is done according to different criteria, and it is used for the definition of compressed packet formats. The criteria for the fields classification are different in each compression mechanism. For example, the Van Jacobson mechanism [Jackobson 1990] simply divides the fields by having a CONSTANT or VARIABLE value. For another example, more advanced classification can be adopted as follows:

- INFERRED: are fields with a value that can be known by examining the package and are never sent
- RANDOM: are fields with a variable value and are sent in each package
- NO_CHANGE: are the fields with a fixed value and are sent at the beginning of the transmission

- DELTA: are the fields with a variable value but which is also predictable as the sequence number and are sent encoded in all compressed packets.

Based on the same principle as for IP header compression, we adopt a packet fields classification according to which packets are compressed. The field classification is summarized in Table 3.2 and the description of the classes is the following:

- Static: fields shared by EDs and the gateway/LM in the local wireless network. For example, CP is a part of content Name shared by all WLAN nodes. Such fields are never sent between the gateway and EDs, either in Interest or in Data packets.

- Inferred: fields that are not exactly the same for the gateway/LM and all the EDs, but they can be calculated using WLAN shared configuration and conventions (e.g. trust conventions). For instance, we assume that common information has been shared between the gateway and EDs through a pairing process. When such information exists, related fields are not transmitted.

- Default value: fields with a default value defined in the NDN specification. That is, these fields are not transmitted when they have the default value, but are transmitted otherwise. For example, ContentType in a Data packet is not transmitted when the packet contains application data.

- Variable: fields that can not be inferred and are not common to WLAN entities. Thus, they must always be transmitted.

- Unsupported: fields that EDs and/or the WLAN do not support because their processing is too complex for constrained devices. They are never transmitted. This class is intended to support Interest/Data packet field restrictions for future NDN over IEEE 802.15.4 forwarding mechanisms. When no explicit restriction is defined, these fields are transmitted.

Unlike in IP, our compression scheme is stateless. That is, nodes do not maintain a context to process packets since the information required for compression and decompression

Table 3.2: Packet fields classification

Class	**Fields**	**Treatment**
STATIC	(part of) *Name*, *NameComponent*	Not transmitted
INFERRED	*SignatureType*, *KeyLocator*	Not transmitted
DEFAULT_VALUE	*ContentType*, *FreshnessPeriod*, *HopLimit* *InterestLifetime*, *MustBeFresh*, *CanBePrefix*	Not transmitted when default value
VARIABLE	*Nonce*, *Content*, *SignatureValue*, *Parameters*	Always transmitted
UNSUPPORTED	Rest of the fields (e.g. *ForwardingHint*).	Not transmitted

is shared through a secured pairing process. The reasons for choosing a stateless compression are multiple. First, as the information compressed in NDN packets mainly consists in names (i.e., characters), even a simple compression scheme will significantly reduce packet sizes. That means, a context is not required to achieve an efficient compression ratio. Second, stateless compression does not require additional memory and overhead to operate, and is simple to implement in constrained devices. Third, from an implementation point of view, a stateless compression process consists in omitting/updating certain bytes when sending the packet. Thus, the sender does not need to maintain the two states of the packet (i.e. compressed and decompressed) as in IP. Similarly, the receiver decompresses the packet by adding/updating certain bytes.

In addition, we should note that the proposed packet compression does not always require a decompression. Indeed, the gateway and EDs perform different (de)compression operations. When an ED receives a compressed Interest, it does not need to calculate all the missing fields to generate the Data or to forward the Interest. Furthermore, if a decompression is required, each field can be extracted separately with TLV encoding. When a Data packet reaches the gateway, it is decompressed by adding/updating the necessary bytes, and then forwarded to the backbone.

Regarding security, data authenticity is not compromised by packet compression. When each ED signs its Data, the original (i.e decompressed) Data packet is signed before the compression. At the gateway, the decompression process adds the exact bytes needed to make the signature verification correct. In this way, if the gateway is partially compromised, the decompressed packet will contain errors, and this will be detected by the consumer upon Data authentication. When the Data signature is delegated to the gateway, the Data packet is signed by the gateway after decompression.

At the time of presenting this packet compression scheme, ICNLoWPAN [Gündoğran et al. 2018] has been published with a header compression support for NDN. It consists of a stateless compression scheme that exploits the TLV encoding to create shorter field representation. A stateful compression scheme is also designed. As for IP header compression, the stateful scheme relies on shared contexts that are either distributed and maintained in the whole network, or are generated and maintained on-demand for a particular Interest-Data path. In short, shared contexts use IDs to replace frequently appearing information such as name prefixes, suffixes, and meta information (e.g., Interest lifetime).

In comparison, our compression scheme can be considered as a hybrid approach between the stateful and stateless mechanisms proposed in ICNLoWPAN. On the one hand, we rely on onboarding process and security configuration to avoid the transmission of some shared information. On the other hand, we avoid sending frequently appearing information, but using field classification instead of a shared context.

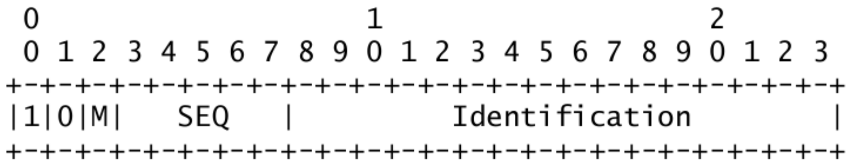
3.4 Additional Features

In this section, we discuss features that are not currently implemented in our architecture, but are either included in the design or intended to be in the future.

3.4.1 Packet Fragmentation

As frequently mentioned, MTUs in low-rate wireless networks are around 100 bytes. Although IoT applications generally do not have to send a large amount of data in one packet, it is not easy to guarantee that NDN packets, including signatures, fit into small MTUs.

Indeed, the mandatory Data signature is frequently achieved with public key encryption, making it difficult to fit a Data packet with payload and signature into one IEEE 802.15.4 frame. Although RSA signatures are too complex for constrained devices, lighter cryptographic algorithms (e.g. ECDSA) can be envisaged. They also require less computational power and are more suitable for constrained devices. Using the proposed packet compression, we expect fragmentation to be required only when public key cryptography is used. In this case, a lightweight link-layer fragmentation mechanism is proposed in [Shang et al. 2016b]. It uses a 3-byte fragmentation header represented in Figure 3.5. In the header, the first bit is set to 1 to indicate that the packet is fragmented. An unfragmented packet will start with the type code for Interest (5) or Data (6) packet, whose highest-order bit is always 0. The More-Fragment (MF) bit indicates whether the current packet is the last fragment. The sequence number (SEQ) and identification fields provide ordering of the fragments, which are used by the receiver to reassemble the original NDN packet.



```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|M|   SEQ   |         Identification        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3.5: Packet fragmentation header in NDN-RIOT [Shang et al. 2016b]

To support the cohabitation of NDN and IPv6 in IEEE 802.15.4 networks, the authors in [Gündogan et al. 2018b] propose to reuse the 6LoWPAN dispatching framework to benefit from the protocol-independent link fragmentation. In their design, a fragmented NDN packet includes 4-byte fragmentation dispatch header identifying the original packet and its size. Subsequent fragments contain 1 additional byte that indicates the fragment offset. Fragments are reassembled on the next hop and passed to the NDN module as regular packets.

3.4.2 Push Traffic

In pull communication mode, an ED waits for an incoming Interest to respond with a Data packet. This is the native communication mode of NDN; it supports many common use cases like monitoring sensors and controlling actuators. However, this mode typically requires EDs to always be on-line, which may cause faster battery exhaustion. Moreover, an NDN architecture for IoT must allow devices to instantly transmit data such as alarms or status changes.

To handle such scenarios, the NDN communication mechanism can be adapted to support a push-based data dissemination. In push mode, an ED publishes its data automatically when it is available, thereby saving itself from always having to wait and listen for incoming interests.

To support push traffic efficiently in NDN, the authors in [Amadeo et al. 2014b] propose three schemes: Interest Notification, Unsolicited Data, and Virtual Interest Polling (VIP). When applied to our architecture, the first solution suggests that EDs include their data in the Interest name. The second solution is to allow the gateway tp accept Data packets from EDs without sending an Interest for it. The last approach consists in using an Interest that it is maintained in the PIT for a longer time period than usual, in such a way that it will be satisfied by a Data packet from ED when it is produced. Note that VIP assumes a negotiation phase between EDs and the gateway to set the maximum time between two Data productions.

However, Interest notification can carry only small amounts of data. In addition, it does not feature the data-centric security of NDN and the produced content can not be cached. In the Unsolicited Data approach, EDs produce Data packets that are not routable according to the NDN scheme since there is no Interest for them. Thus, Data packets can only be retransmitted by broadcast. Moreover, accepting unsolicited Data may be dangerous for the network and the gateway as it opens a breach for DoS attacks.

For our architecture, we decide to retain only the third solution. Despite additional resources required at routers in between a consumer-producer path, this scheme does not raise security and overhead issues. It can be used in our scenario where the consumer is typically hosted in a powerful node such as the gateway. The VIP scheme works as follows (see Figure 3.6(a)). In the configuration phase, the ED and the gateway exchange information about the maximum interval between successive Data generation, let this parameter be τ.

Then, the gateway may send a long-lived Interest (ll-INT) and waits for a virtual timeout interval (vTO), slightly higher than τ, to receive the Data. Here, vTO is relatively long and corresponds to the ll-INT lifetime. If the content is received before vTO expires, the gateway simply refreshes the ll-INT lifetime and does nothing. Otherwise, if the vTO timer expires without receiving any Data, the gateway transmits a standard Interest with the usual and relatively short timeout (TO). If a Data packet is not received within TO, a new Interest is retransmitted by the gateway. Otherwise, at the successful reception of the Data packet, a vTO timer is started afresh.

We should note that VIP is useful when EDs produce data periodically, and may be even more useful if the data are produced quite frequently. In this case, keeping an Interest in the PIT for a long period of time may consume fewer resources than issuing a new Interest each time new data is available.

For the case when a push communication is not frequently needed, another mechanism that we can use is the Advertise Interest (AdvInt). AdvInt works as follows (see Figure 3.6(b)). When the ED needs to push data, it sends an Interest to advertise it to applications. The Interest name is composed of the prefix name allowed to the device, followed by the advertised Data name. The advertise Interest can be routed normally with NDN until it reaches an interested consumer, such as the gateway. The gateway can then fetch the advertised Data name and send an Interest to retrieve it. This mechanism creates an overhead as it requires the ED to send one additional Interest. However, it can be acceptable if it is not used frequently.

3.4.3 Caching and Energy Management

Caching is an important feature to ensure data availability and reduce energy consumption. In the following, we report on some work related to caching in NDN constrained wireless networks such as WSNs. These solutions can be envisioned to improve our architecture.

In [Hahm et al. 2017], the authors propose a side-protocol for NDN, called CoCa, to enable distributed cooperative caching of IoT content. Extensive real-world large-scale experiments have been performed on IoT networks with up to 240 nodes, and on an emulator with up to 1000 nodes. Results show that with NDN+CoCa, devices can achieve a 90% reduction in energy consumption compared to the state-of-the-art while maintaining recent IoT content availability above 90%.

More importantly, the authors argue that cooperative distributed caching in general could also be implemented on top of IP. However, NDN uses caching capabilities built in the network layer. In comparison, an IP stack does not provide built-in caching capabilities,

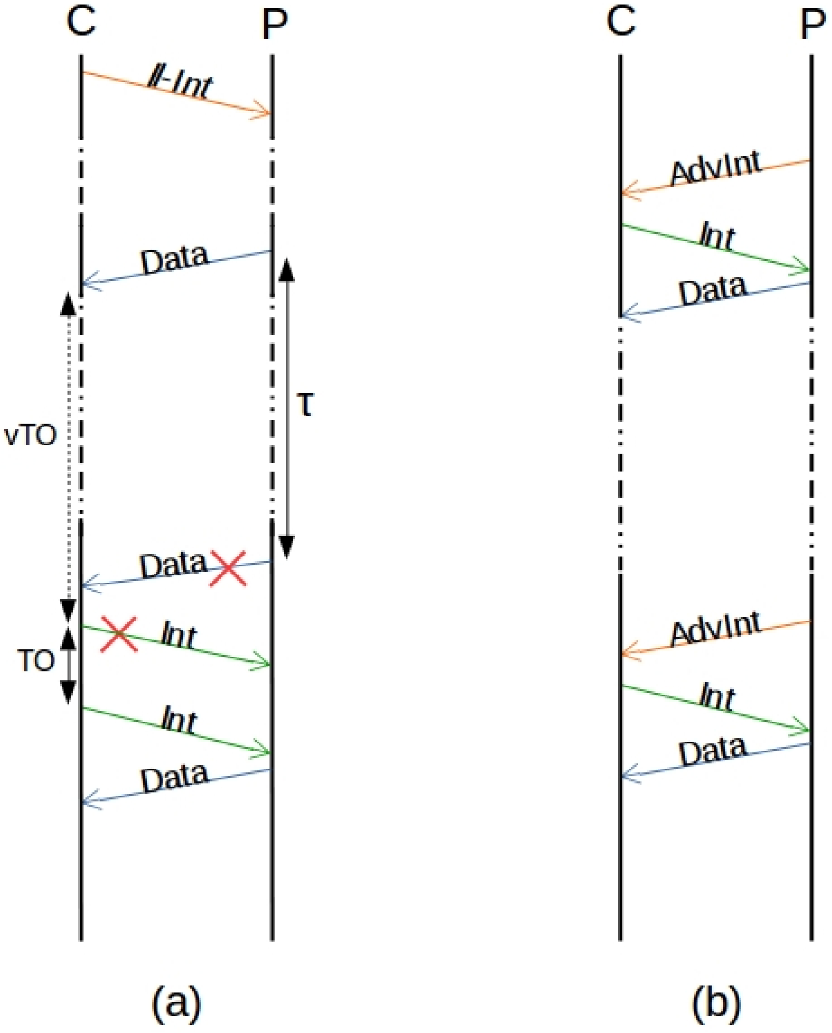


Figure 3.6: Push-mode mechanism illustration: (a) Virtual Polling Interest, (b) Advertising Interest

but still uses a large part of the RAM on low-end IoT devices (e.g. on IoT devices with 16kB of RAM or less). Hence, on low-end IoT devices, NDN can dedicate significantly more RAM to content caching compared to similar solutions using IP. Results indicate that even a small amount of additional cache size (e.g., 5 kB) can significantly increase the content availability. Thus, solutions such as NDN+CoCa can provide a significant advantage over a similar approach with IP in terms of energy efficiency.

In [Hail et al. 2015], the authors propose a probabilistic CAching STrategy for the INternet of thinGs (pCASTING) to improve caching efficiency in NDN wireless IoT networks. It relies on the freshness of data, the energy level and storage capacity of constrained devices to dynamically adapt the caching probability of each node. Simulation results show that the approach outperforms traditional NDN caching mechanisms in terms of data retrieval and network energy efficiency.

Sleep mode of IoT devices introduces the problem of data availability. To cope up with that, a Dataset Synchronization protocol for WSN has recently been proposed in [Xu et al. 2018]. The protocol divides sensors into groups, each has a shared dataset; through dataset synchronization within each group. The protocol ensures that the group's latest dataset is always accessible from its active sensors. Reliable dataset synchronization, NDN's Interest aggregation and Data caching are exploited to optimize energy consumption. Simulation results show that a data availability close to 100% can be achieved under various network conditions with negligible overhead.

3.5 Conclusion

In this chapter, the practical deployment of NDN in IoT systems has been investigated. An integration of NDN from the edge of the network has been identified as a reasonable approach after studying all the possibilities. Based on this choice, a realistic NDN-802.15.4 architecture, including NDN-specific mechanisms, has been designed and discussed. This is considered as a first step towards enabling the ICN/NDN paradigm in the IoT. Through the NDN-802.15.4 architecture, we aim to shape a viable NDN-IoT duo with the following objectives:

1. Investigate how the integration of the NDN paradigm with low-rate low-power wireless technologies can be designed in comparison with IP integration (i.e., 6LoWPAN).
2. Show the flexibility of enabling NDN in IoT devices by exploring mechanisms that cannot be envisaged with IP such as Name-Payload-Field balancing.
3. Enable the vision of NDN in real-life IoT applications to take advantage of NDN features.

However, providing mechanisms to support NDN packets over IEEE 802.15.4 is only the first stage of the NDN-802.15.4 integration that we aim to achieve. A major networking aspect should be examined, which will inevitably impact the way NDN communication is handled in low-rate wireless environments. This important concern is the packet forwarding in IEEE 802.15.4 mesh networks. Indeed, in addition of reducing packet size, a lightweight forwarding strategy for wireless mesh networks is needed to complete the NDN-802.15.4 integration.

As mentioned before, using NDN directly on the link-layer layer is a wise choice in wireless networks. This raises various questions on how to design forwarding strategies. First, we need to figure out whether unicast MAC addresses must be mapped to NDN names, or a broadcast forwarding is more efficient. Second, while a forwarding strategy is supported at the NDN network layer, it may impact the underlying link-layer components such as the CSMA algorithm.

To answer these questions, testbeds can be used as real-world evaluation tools, but are not sufficient as they are limited by the number of nodes, scenario possibilities and measurement accuracy. To allow large-scale and quick evaluation of wireless forwarding strategies, a ready-to-use simulator for wireless NDN in IoT is needed. To further enhance the accuracy and understand the impact of the parameters, analytical models are also needed. In the next chapter, we describe how we support the evaluation of NDN-802.15.4 designs through three tools that make up our evaluation environment: the testbed, the simulation framework and the mathematical model.

Chapter 4

Evaluation Tools

4.1 Introduction

As pointed out in the previous chapter, an important feature to support in our NDN-802.15.4 architecture is the wireless forwarding over IEEE 802.15.4. However, we should note that NDN wireless forwarding, particularly in constrained networks, is slightly different from the usual NDN forwarding used in wired networks. The main reason is that using one wireless radio, nodes have no possibility to choose among different interfaces to forward packets; all packets are transmitted through the same network interface. Moreover, packet forwarding must deal with redundancy if broadcast is used, or deal with MAC addresses when unicast is used. Obviously, the main challenge for a wireless forwarding strategy is to reduce network overhead and useless retransmissions while keeping efficient data availability and dissemination.

To be able to design and test wireless forwarding strategies, we set up a research environment that includes three complementary tools: a testbed, a simulation framework based on the OMNeT++ simulator, and a mathematical model. First, the testbed reflects the NDN-802.15.4 architecture presented in the previous chapter with a small number of fixed nodes. It allows us to measure small-scale operations such as one-hop round-trip time, memory usage and packet compression delay.

Although an open-access testbed such as IoT-Lab [Adjih et al. 2015] provides the needed features to test and evaluate IoT solutions, we decided to design a new testbed for our evaluation. The main reason is that we want the testbed to evolve to a prototype for cow health monitoring system with NDN. Therefore, we use the testbed in an incremental way to integrate new features, instead of supporting only the evaluated aspect as it is usually the case in a general purpose testbed.

Second, to overcome the limitations of a testbed in evaluating complex forwarding scenarios, such as node mobility, we developed the NDN-OMNeT framework. NDN-OMNeT also reflects the gateway-to-ED and ED-to-ED communications of our architecture, but allows evaluation with many more nodes, mobile or not, with various network topologies, and different link-layer solutions such as IEEE 802.11 and IEEE 802.15.4. The main purpose of this framework is to quickly compare and evaluate wireless forwarding strategies as our implementation includes several state-of-the-art solutions.

Using the ndnSIM simulator [Afanasyev et al. 2012] mentioned in Chapter 2, broadcast-based wireless forwarding strategies used for evaluation are not natively supported, and need to be implemented. Therefore, we decided to implement these forwarding strategies along with NDN behaviour in another simulator, OMNeT++ which does not support NDN. We believe this will help to bring NDN to an additional simulation tool, which is widely used in IoT, VANETs, and WSNs community.

Finally, we model the basic NDN forwarding strategy proposed in the literature and introduced later in this chapter, CF (Controlled Flooding). The mathematical model is used to study the efficiency of NDN in a simple wireless network topology under different content popularity. The three tools are presented below in their respective sections, and used in the next chapter.

4.2 Testbed

The NDN-802.15.4 architecture has been initially designed and tested in a cow monitoring system, as previously mentioned. However, since it is not easy to deploy experimental features in a farm production environment, particularly with livestock, the deployment has been reused as a testbed in our lab. The testbed includes a WLAN with one gateway and four EDs. To achieve that, we designed a typical NDN-802.15.4 gateway, a set of NDN-802.15.4 constrained devices, and a monitoring application. Together with the IEEE

802.15.4 technology, we chose typical hardware to use in our architecture. The hardware equipment considered, followed by the design of each aforementioned entity are described below.

4.2.1 Hardware Technologies

As the focus is on realistic deployments, we consider current IoT equipment available for IoT users. First, Arduino single-board microcontrollers are a typical example of constrained devices to create EDs; with a low-power, slow-speed CPU and a few kilobytes of RAM and Flash. When deploying IoT applications in environments such as agricultural fields, it is common to use sensors and actuators running on such constrained equipment. These devices are intended to support NDN stack implementation and run a simple NDN producer application that basically creates, names and signs Data packets, and processes Interests received from the gateway. To communicate with the gateway and with other EDs, a wireless radio module is connected to the Arduino chip. The wireless radio is chosen according to the underlying wireless technology, which is IEEE 802.15.4. Second, the gateway typically has more CPU and memory resources than EDs. It also has more power as it is commonly plugged in to a constant source of energy. The class of equipment we use as gateways is represented by Raspberry-Pi single-board computers. Raspberry-Pi hardware is widely used for prototyping and making IoT low-cost applications for testing and developing Proofs-of-Concept and embedded systems.

Finally, wireless communication within the WLAN is based on the IEEE 802.15.4 technology using XBee modules. XBee is the brand name of ZigBee compatible radio modules from Digi International. ZigBee is a suite of high-level communication protocols based on the IEEE 802.15.4 specification. XBee modules can be used to deploy wireless mesh networks using small, low-cost and low-power digital radios. XBee radio modules are very popular in embedded applications, wireless sensor networks, monitoring and IoT applications. We should note that we use XBee modules as IEEE 802.15.4 radios, without any ZigBee feature or related configuration. Each of the hardware technologies considered has different product types available on the market, Table 4.1 reports on the equipment particularly used to prototype our architecture.

Table 4.1: Hardware Technologies Considered

Name	Usage	Features/resources
Arduino Uno	Equipment used to create EDs that support NDN over IEEE 802.15.4. EDs include sensors and/or actuators to collect data (e.g., temperature) and/or execute actions (e.g., lock a door).	32 KBytes Flash Memory 2 KBytes SRAM 16 MHz CPU SoC: ATmega328P Size: 68 mm X 53 mm Price: $24.95
Arduino Due		512 KBytes Flash memory 96 KBytes SRAM 84 MHz CPU SoC: ARM Cortex M3 Size: 101 mm x 53 mm Price: $32.13
Arduino Mega		256 KBytes Flash memory 8 KBytes of SRAM 16 MHz CPU SoC: ATmega2560 Size : 101 mm x 53 mm Price: $38.50
Raspberry Pi 2 Model B	Equipment used to create the gateway. It uses NDN over IEEE 802.15.4 to communicate with EDs, and uses NDN over UDP/IP/Ethernet to communicate with applications over the Internet. Thus, it includes at least two network interfaces: IEEE 802.15.4 radio and Ethernet/Wifi.	1GB RAM 900MHz quad-core ARM Cortex-A7 CPU Linux OS Size: 85 mm x 56 mm Price: $39.95
XBee transceiver	Equipment used to provide EDs and gateways with connectivity over low-power low-rate wireless technology. We use the IEEE 802.15.4 standard supported by this equipment. Each WLAN device is equipped with one XBee transceiver, connected via a serial communication.	250 Kbps data rate Frequency Band 2.4 GHz Range: 90 m indoor Size: 24 mm x 27 mm Price: $39

4.2.2 Gateway Design

The gateway is implemented using a Raspberry Pi [Wikipedia] with an XBee radio [Digi International Inc. 2008] for wireless communication. To provide Raspberry Pi with NDN-over-802.15.4 communication, we implemented a process that runs next to the NDN module (i.e., NFD). The role of this process is to intercept Interests with a certain prefix name and to send them through the IEEE 802.15.4 radio. Figure 4.1 depicts the software and hardware components involved in the Raspberry Pi after integrating NDN-over-802.15.4. Details of the software structure and forwarding are provided below.

4.2.2.1 Software Structure

The NDN-802.15.4 process manages sending Interests and getting Data, and receiving Interests and sending Data. That is, the gateway is able either to receive Interests from EDs, forward them to the backbone and get the Data back, or to forward Interests from the backbone to EDs and get Data packets back. In our design, the IEEE 802.15.4 technology

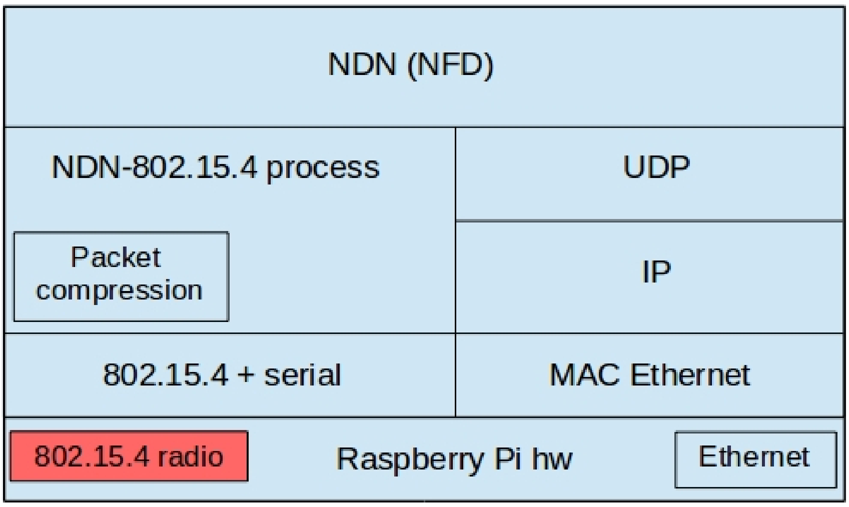


Figure 4.1: Architecture of the gateway

is a link-layer for NDN. It uses the broadcast address 0xFF to send packets over IEEE 802.15.4.

Figure 4.2 gives a simplified version of the NDN-802.15.4 process running on the gateway.

4.2.2.2 Routing and Forwarding

In our testbed, the NDN-802.15.4 process intercepts Interests with a certain prefix p to send them over the wireless link. For example, p may be *"/farm"*. This prefix has to be set to forward all Interests with names starting with p to the IEEE 802.15.4 WLAN. To achieve that, the NDN-802.15.4 process registers a Face to NFD, which creates a FIB entry to bind prefix p to the local Face of the NDN-802.15.4 process. Then, NFD acts in a normal way: it forwards Interests with prefix p to the NDN-802.15.4 process, and when the corresponding Data comes back it sends it to the appropriate application or next hop according to the PIT. That is, in the gateway NFD uses NDN routing information to forward Interests from the backbone to the WLAN and Data from the WLAN to the backbone.

As an illustration, Table 4.2 gives the typical FIB at the gateway. When Interests are issued from the WLAN (i.e., EDs are consumers), the gateway forwards those Interests to

```
/* callback for incoming Interest from the backbone */
function onInterest(interest) :
    frameBuffer = encodeAndCompress(interest)
    ieee802154.broadcast(frameBuffer)

/* callback for incoming Data from the backbone */
function onData(data) :
    frameBuffer = encodeAndCompress(data)
    ieee802154.broadcast(frameBuffer)

function main() :
    /* to receive Interests from the backbone */
    backboneFace = Face()
    prefix = "/farm"
    backboneFace.registerPrefix(prefix, onInterest)

    /* connect to the 802.15.4 radio */
    ieee802154 = Ieee802154()

    while True:
        /* process incoming packets from the backbone */
        backboneFace.processEvents()

        /* process incoming packets from the WLAN */
            frame = ieee802154.wait_read_frame(0.01)
            if frame :
                    if frame.isData() :
                        data = decodeAndDecompress(frame)
                        backboneFace.put(data)
                    else if frame.isInterest() :
                        interest = decodeAndDecompress(frame)
                        backboneFace.issue(interest)
```

Figure 4.2: NDN-802.15.4 process operations

Table 4.2: Typical FIB at the gateway

Prefix	**Face**	**Cost**
/farm	NDN-802.15.4 process	0 (local)

the corresponding next hop in the wired network. Once the Data is received, the gateway sends it over the IEEE 802.15.4 link towards the ED that requested it.

4.2.2.3 Implementation

The packet compression scheme in our architecture is implemented in the NDN-802.15.4 process. Moreover, to process IEEE 802.15.4 frames according to the specification, we implemented a simple library that interacts with the XBee module via the serial port.

The NDN-802.15.4 process is implemented in Python using the PyNDN2 library. It can run on any Linux-based distribution with NFD installed such as Ubuntu, Debian and Raspbian. The library that handles IEEE 802.15.4 frames is also implemented in Python.

4.2.3 End-device Design

End-devices are implemented using Arduino boards with XBee radios for wireless communication. A lightweight version of the NDN protocol stack is available on Arduino thanks to the ndn-cpp Lite library. This library supports encoding and decoding TLV packets, and includes cryptographic algorithms such as HMAC and ECSDA. To implement EDs as described in our architecture, we extended ndn-cpp Lite with a simple IEEE 802.15.4 communication library that makes it possible to handle XBee modules to send and receive NDN packets over IEEE 802.15.4 frames, in the same way as the gateway does.

In practice, to access XBee modules, our extension library uses a serial communication library included in the Arduino environment. To support forwarding strategies, we also implemented light versions of the NDN structures; FIB, PIT, and CS, as they are not included in the version of ndn-cpp Lite that we used.

As mentioned above, EDs typically run producer applications. Thus, having the ndn-cpp Lite library and the IEEE 802.15.4 extension, a simple producer application can be written as an Arduino sketch.

Figure 4.3 depicts the ED architecture including software modules and libraries. Figure 4.4 is a picture of an ED actually used in the testbed, here with an Arduino DUE and an accelerometer. Figures 4.5 and 4.6 contain the typical producer application sketch in which the ED receives an Interest from the IEEE 802.15.4 radio, processes it, and replies with the corresponding Data.

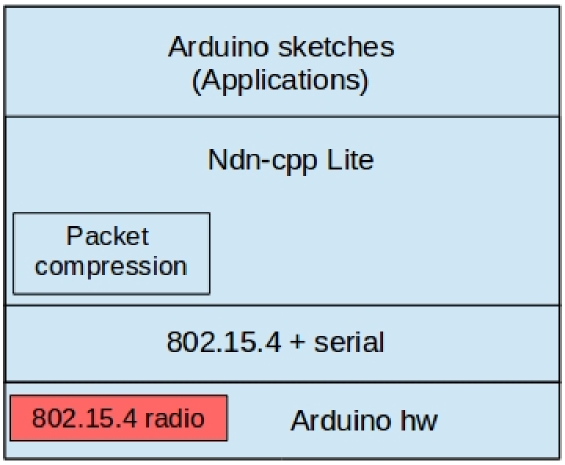


Figure 4.3: Architecture of the ED

4.2.4 Applications

To complete the testbed, we implemented a typical NDN consumer application that collects and displays data produced by EDs. The application runs on a traditional computer or laptop and periodically issues Interests that are forwarded by the gateway to the WLAN. Depending on the purpose of the monitoring, variants of the application exist and can display data on a map, on a dashboard, or simply store it. Applications are developed in Python using the NDN library PyNDN2.

4.2.5 Deployment and Evaluation

The NDN-802.15.4 testbed has been deployed in a smart agriculture scenario that reflects the cow health monitoring example presented throughout this document. We recall that cow health monitoring systems use sensors (e.g. movement, temperature, etc.) installed on cows to collect individual cow data. The data are then analyzed to detect

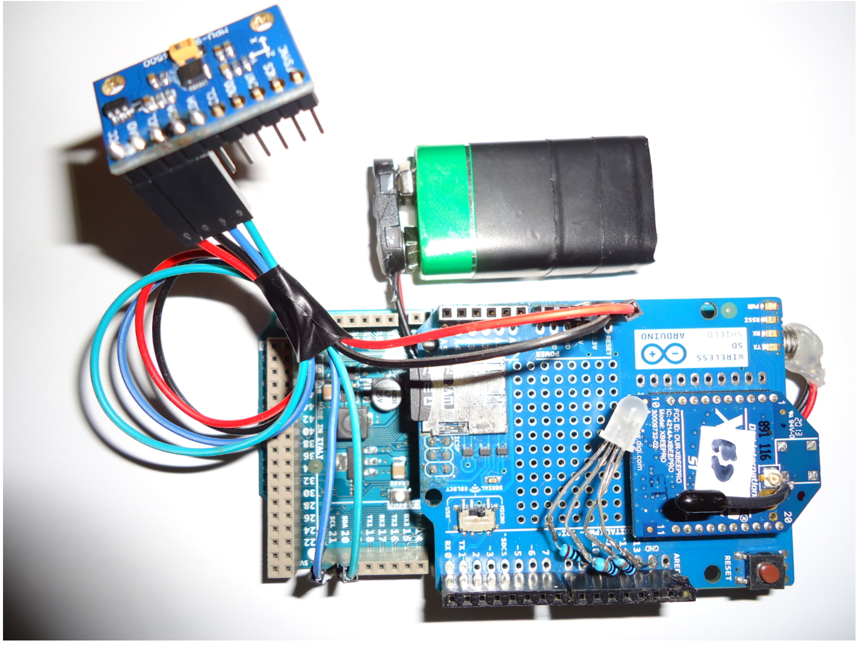


Figure 4.4: Picture of an ED

whether a cow is sick, or to forecast activities such as heat periods to make breeding decisions.

The deployment consists in one WLAN with one gateway and four EDs. The equipment used consists in a Raspberry Pi as the gateway/LM and Arduino boards as EDs. A consumer application runs on a laptop connected to the gateway on the local network (LAN) and periodically sends Interests to collect data and visualize data.

The WLAN is identified by the common prefix CP = */cowHelath/farm/area/1*. Each ED serves content under a name */cowHelath/farm/area/1/cow/<cowID>/temp*, formed by the CP, a cow ID, and data type which is temperature in our case. Here, <cowID> is a 1-byte number that identifies each cow.

A 32-byte HMAC signature is used to secure Data packets. The secret keys are directly hard-coded in EDs and the monitoring application since key management is beyond the scope of this work. However, the security support descibed in [Zhang, Yu, Zhang, Newberry, Mastorakis, Li, Afanasyev, and Zhang 2018] and presented in Chapter 2 can be deployed as well.

An Interest packet issued by the consumer (i.e., uncompressed) carries the Name, a

```
#include "node.h"        /* NDN-802.15.4 library */
#include "MPU9250.h"         /* Accelerometer library */
#include "ndn-cpp/lite/data-lite.hpp"
#include "ndn-cpp/lite/interest-lite.hpp"
#include "ndn-cpp/lite/util/crypto-lite.hpp"
#include "ndn-cpp/lite/encoding/tlv-0_2-wire-format-lite.hpp"
MPU9250 myIMU;            /* Accelerometer object */
using namespace ndn;        /* to use ndn-cpp objects */
Node node = Node();        /* NDN producer node */
/* signature key */
const uint8_t keyBytes[] = {...};
/* prefix for producer */
struct ndn_NameComponent prefixComponents[3];
NameLite  prefix( prefixComponents, sizeof(prefixComponents) /
    sizeof(prefixComponents[0]));

void setup() {
  Serial.begin(57600);
  node.begin(Serial);  /* init serial connection to 802.15.4 radio */
  prefix.append("2");
  prefix.append("mvmnt");
  node.setPrefix(prefix, &onInterest); /* set producer prefix */
  delay(500);
  /* Accelerometer init and calibration */
  /* ... */
  randomSeed(analogRead(0));
}
void loop()
{
  node.processEvents();
  /* read and process accelerometer data */
  /* ... */
}
```

Figure 4.5: Producer application code running on an ED - Main program

```
/* callback to process Interests */
void onInterest(const InterestLite& interest)
{
  digitalWrite(greenLed, HIGH);
  ndn_Error error;
  uint64_t cowID = 0;
  /* get cow ID */
  if ( (error = interest.getName().get(2).toSegment(cowID)) )
  {
    return;
  }
  /* some control on Interest name */
  /* ... */
  /* If OK: Create and send Data */
  ndn_NameComponent dataNameComponents[3];
  DataLite data(dataNameComponents, sizeof(dataNameComponents) /
    sizeof(dataNameComponents[0]), 0, 0);
  data.setName(interest.getName());
  data.setContent(BlobLite((const uint8_t*) contentBuffer, strlen(contentBuffer)));
  data.getSignature().setType(ndn_SignatureType_HmacWithSha256Signature);
  /* First encoding */
  uint8_t encoding[100];
  DynamicUInt8ArrayLite output(encoding , sizeof(encoding), 0);
  size_t encodingLength, dSignedPortionBeginOffset, dSignedPortionEndOffset;
  if ( (error = Tlv0_2WireFormatLite::encodeData(data, &dSignedPortionBeginOffset,
  &dSignedPortionEndOffset, output, &encodingLength))){
    return;
  }
  /* Data signature */
  uint8_t signatureValue[ndn_SHA256_DIGEST_SIZE];
  CryptoLite::computeHmacWithSha256(keyBytes, sizeof(keyBytes),
            encoding + signedPortionBeginOffset,
  signedPortionEndOffset - signedPortionBeginOffset, signatureValue);
  data.getSignature().setSignature(BlobLite(signatureValue, ndn_SHA256_DIGEST_SIZE));
  /* Encode again to include the signature.*/
  if ((error = Tlv0_2WireFormatLite::encodeData(data, &signedPortionBeginOffset,
  &signedPortionEndOffset, output, &encodingLength))){
    return;
  }
  /* Compress and send Data */
  node.compressAndPutData(data);
}
```

Figure 4.6: Producer application code running on an ED - Interest processing

Table 4.3: Testbed deployment parameters

Parameter	**Value**
Interest size (uncompressed)	44 bytes
Data size (uncompressed)	105 bytes
Data signature size (HMAC)	32 bytes
Payload size (Content)	20 bytes
Number of EDs	4

Nonce, a MustBeFresh indicator to accept only fresh data, and has a lifetime with the default (i.e., 4 seconds).

In the uncompressed Data packet created by the ED, the Name contains an additional component that represents the timestamp of the data collection. The Data name obtained is then */cowHelath/farm/area/1/cow/<cowID>/temp/<timestamp>*. Each Data packet has a 4-byte content that contains the measured temperature. In MetaInfo field, a 1-byte ContentType indicates that the packet carries raw content, followed by a 2-byte FreshnessPeriod field. Finally, the Signature TLV component contains a 1-byte SignatureType and a 16-byte KeyLocator.

Testbed parameters are summarized in Table 4.3. A typical Interest-Data exchange operates as follows. The Interest issued by the application is forwarded to the gateway over UDP/IP. Then, the gateway compresses it and sends it over the IEEE 802.15.4 link. After the wireless forwarding process, the corresponding ED responds to the Interest by a signed Data with the corresponding content. Finally, the gateway decompresses the Data packet and forwards it toward the consumer application.

Various types of evaluation are reported in the following: (i) a features comparison between our NDN-802.15.4 architecture and 6LoWPAN, with a discussion on the feasibility in terms of implementation and security, (ii) a theoretical evaluation of the gain achieved with the packet compression scheme, (iii) results on code size and communication delays measured in our deployment.

Table 4.4: NDN-802.15.4 and 6LoWPAN features comparison

Feature	NDN-802.15.4	6LoWPAN
Fragmentation	Yes	Yes
Packet structure	Flexible packet format	Fixed packet format
Compression	Stateless packet compression	Stateful header compression
Mobility (consumer side)	Simple adaptations	Additional protocols NEMO, AdapterMIPv6, etc.
Security	Native data-centric security	MAC and TLS security

4.2.5.1 Features and Feasibility

The main objective of the implemented schemes is to show that traditional operations such as header compression can be designed more simply in NDN while gaining interesting improvements. Therefore, the purpose is not to design a strong and complete compression mechanism, which requires intensive investigation of NDN application behaviour, which is not yet well known.

As mentioned before, the packet compression does not always require decompression. Using the data-centric security of NDN, the data related to each cow is signed directly when it is collected by the corresponding ED. Hence, every cow has a unique identity (not an address) in the network system. This identity is securely bound to its data at network level.

As an empirical evaluation, we report in Table 4.4 an overall comparison between NDN-802.15.4 and 6LoWPAN features.

4.2.5.2 Theoretical Performance

Figure 4.7 depicts a comparison between initial (i.e., uncompressed) Interest and Data packets and their compressed versions. The Data packet represented here does not include the signature, which is about *32* bytes in both initial and compressed Data. Considering the packet structures in the deployment described above, the size of the Interest is reduced from *57* bytes to *24* bytes. Similarly, the Data packet size is reduced from *93* bytes to *34* bytes.

Note that the most part of the compression gain is achieved by reducing name size in the packets. That is, when CP becomes longer such as in large scale deployments, the compression gain will increase. Moreover, this compression gain comes at the cost of only few microseconds of delay, as no complex processing or context storage is required.

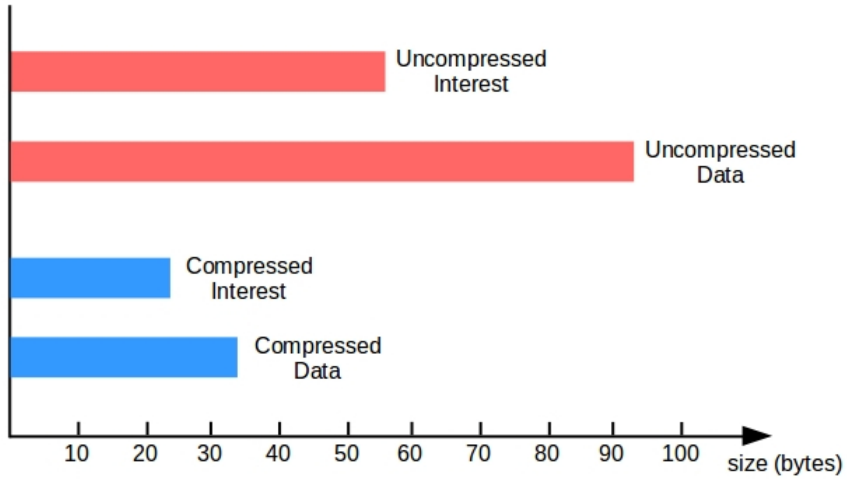


Figure 4.7: Compression improvement

4.2.5.3 Prototype Measurements

Table 4.5 reports on memory and processing time required by the NDN stack implementation in EDs considering both Arduino UNO (*16 Mhz* MCU) and DUE (*84 Mhz* MCU).

Concerning memory space required by the implementation, only 28% of flash memory and 50% of RAM are needed in the Arduino UNO. The implementation on the Arduino DUE occupies 6% of the total memory. The evaluated implementation includes the three NDN data structures, the communication over IEEE 802.15.4 including packet compression scheme and the R-LF strategy described in Chapter 5. Although these values increase when adding NDN packet definition and security algorithms, the leeway is still large for such components.

As an empirical comparison, some open source implementations of the IPv6 stack over IEEE 802.15.4 on Arduino (Mega) take about 12% storage and 45% RAM, while our NDN stack occupies about 3% storage and 12% RAM on an Arduino Mega board.

Table 4.6 reports on the one-hop Round Trip Time (RTT) measured at the gateway

Table 4.5: Memory and processing measurements

Operation	Arduino UNO	Arduino DUE
Interest forwarding	$145\mu s$	$55\mu s$
Data forwarding	$50\mu s$	$10\mu s$
Required memory	28% (Flash) 50% (RAM)	6% (Total)

Table 4.6: Communication measurements at the gateway

Operation	Measure
Round-Trip Time	$72ms$
Compression Delay	$6\mu s$

to send an Interest and get the corresponding Data using Arduino DUE devices, and the Compression Delay (CD) added to the communication due to packet compression.

In comparison, the measured RTT is below 6LoWPAN performance usually reported (e.g *9* to *25 ms*) [Brendan, David, Desmond, Ricardo, Meriel, and Ciaran 2009; Gardasevic, Mijovic, Stajkic, and Buratti 2015]. However, 6LoWPAN packets are not signed and additional layers are required to support data naming, while the measured RTT includes Data creation and signature, and Interest-Data (de)compression. Moreover, we recall that our implementation is a prototype that can be improved.

The compression delay (CD) does not exceed $6\mu s$ as it only consists on adding/skipping bytes while transmitting a packet.

NDN-based solutions for IoT need accurate evaluation to convince industrialists, investors and IP-enthusiasts. Although some aspects of IoT solutions can be evaluated in a testbed, real-world deployments are commonly based on realistic needs, and sometimes do not support certain features or "outlandish" scenarios. In such situations, network simulators become essential. To handle those situations, we designed an NDN simulation tool for IoT introduced in the next section.

4.3 NDN-OMNeT Simulation Framework

Beside wireless forwarding, use cases in which simulations can be useful are manifold; ranging from NDN for public safety and tactical networks [Gibson et al. 2017] to NDN for WSN [Amadeo et al. 2013]. However, networking and system are tightly related in IoT, and a pure network simulator does not allow us to model interactions inside an IoT device for example. Therefore, experimental NDN designs for IoT require the best of both worlds: a tool capable of simulating network protocols and system-level interactions. Moreover, a graphical visualisation of the simulated networks can be helpful to illustrate scenarios. For these reasons, we chose to implement the NDN behaviour on the OMNeT++ simulator [OMNet++]. In addition to being widely popular in IoT and WSN research community, OMNeT++ can be used to simulate wired and wireless networks as well as on-chip networks, and so on. OMNeT++ is not a native network simulator. However, ready-to-use domain-specific functionalities are provided by frameworks such as INET [INET], which contains models for the IP protocol stack, link-layer protocols, mobility, etc. In this section, we describe our NDN-OMNeT framework for IoT. This framework is based on INET and is designed to meet the following objectives:

1. Evaluate how a design affects NDN internal structures in a given topology/scenario.
2. Provide a good visualization of the NDN communication process for testing and teaching purposes.
3. Provide an easy-to-use framework to quickly experiment wireless forwarding strategies without additional implementation.

4.3.1 Framework Design

The NDN core module is designed as a network layer (NdnL3) that implements the network layer interface of INET (INetworkLayer). Within an NDN host, NdnL3 is connected to the upper layer; expected to be the application layer, and the lower layer consisting of wired/wireless network interfaces. However, the NDN network layer can run on top of (or beside) IP with minimal adaptations. Following the modular approach of OMNeT++, we designed the NDN entities as independent modules included in the NdnL3 module (see Figure 4.8). OMNeT++ modules use message passing through connected Gates to communicate. The notion of Gate provides a native abstraction of the Face concept, making the NdnL3 module interact with upper (i.e. application) and lower (i.e. link) layers in a transparent way.

In the following, we present the main components of NDN-OMNeT and provide a description of the packet representation it uses.

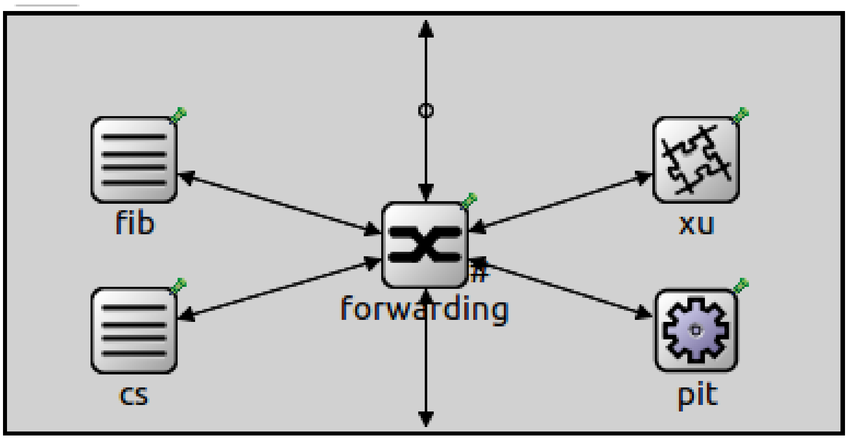


Figure 4.8: NDN L3 module and its entities

4.3.2 Host and Application Modules

NDN hosts. To represent devices such as EDs in the NDN-802.15.4 architecture, a base NDN wireless host (NdnWirelessHostBase) is implemented. It includes the typical wireless host components and the NDN layer (NdnL3). This module is used directly as a relay node since it does not include any application (see Figure 4.9). By extending NdnWirelessHostBase, a typical IoT end-device is created (NdnWirelessHost) with consumer and/or producer applications.

Applications. Two base applications are implemented. A producer (ProducerAppBase) with the following parameters: (i) prefix; under which the content is produced. (ii) dataLength; if not provided, the TLV length is computed and used. (iii) startTime. (iv) stopTime. (v) dataFreshness.

A consumer (ConsumerAppBase) with the following parameters: (i) startTime. (ii)

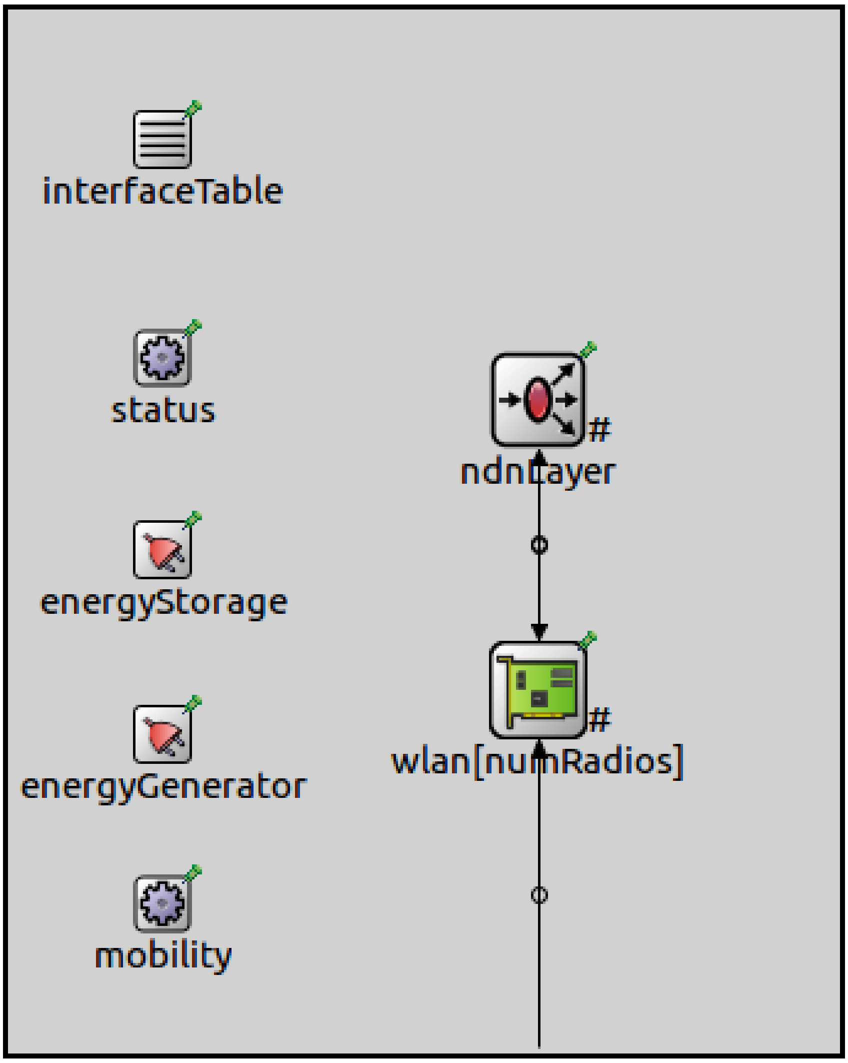


Figure 4.9: NDN simple wireless host (e.g., router)

stopTime. (iii) prefix; for which Interests are issued, with additional name components (e.g. sequential number) (iv) interestLength; if not provided, the TLV length of the Interest is computed and used. (v) sendInterval; time to wait between two issued Interests. (vi) numInterests; number of Interest to issue. (vii) interestReTx; maximum number of Interest retransmissions. (viii) interestLifetime; value of the Interest lifetime field.

Figure 4.10 shows a typical NDN end-device with applications (NdnWirelessHost).

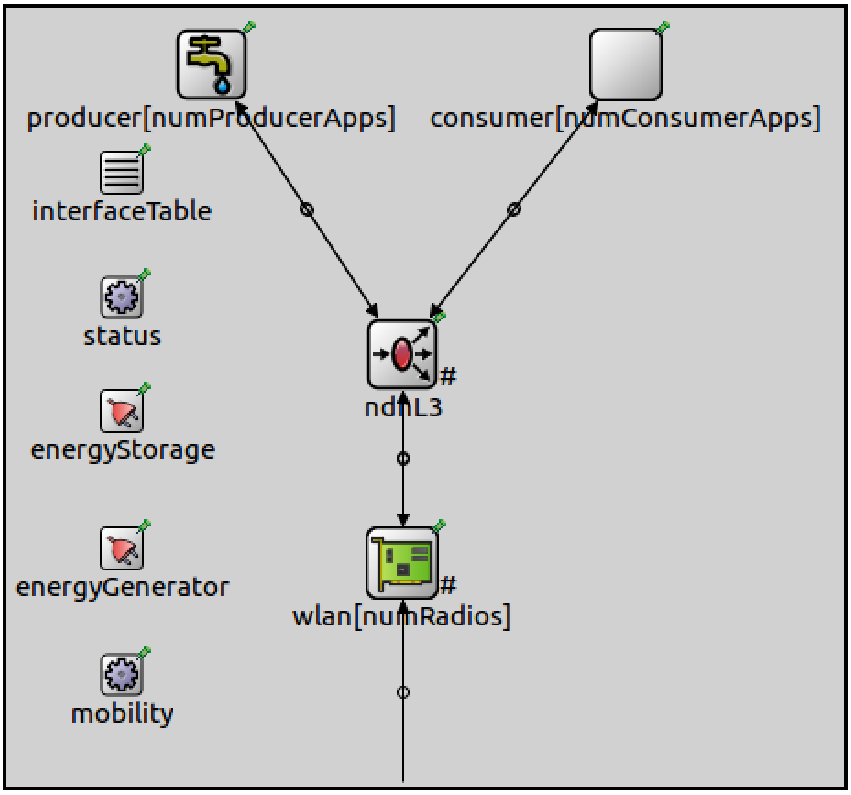


Figure 4.10: NDN wireless host with applications (producer and/or consumer)

4.3.3 NDN Layer Modules

Pending Interest Table. IPit is an abstraction of the PIT. It includes a typical entry which stores the following information: (i) A copy of the forwarded Interest, (ii) Incoming Face(s) of the Interest, (iii) Face(s) to which the Interest is forwarded, (iv) Expire time of the entry and, (v) Source MAC. IPit supports the following functions: (i) Lookup, (ii) Create, (iii) Remove, (iv) Update, (v) Print.

On Interest timeout, IPit can emit a signal with a notification that includes the Interest, its incoming Face(s) and source MAC address(es).

A base implementation (PitBase) of IPit is provided with the following parameters: (i) interestLifetime. If no value is provided, the lifetime value of the Interest is used instead. (ii) maxSize. When the PIT is full, no Interest can be forwarded.

Forwarding Information Base. IFib is an abstraction of the FIB. It includes a typical entry which keeps the following information: (i) Name prefix, (ii) Face(s), (iii) Destination MAC address, (iv) Expire time of the entry (optional). IFib provides the following functions: (i) Lookup, (ii) Create, (iii) Remove, (iv) Update, (v) Register prefix; used to create an entry to the local application and, (vi) Print.

When a prefix expires, IFib implementations can emit a signal with a notification that includes the prefix and its corresponding Face(s) and MAC address(es).

A base implementation of IFib (FibBase) is provided with the following user parameters: (i) entryLifetime and, (ii) maxSize.

Content Store. ICs is an abstraction of the CS. It includes a typical entry which stores the following information: (i) A copy of the Data packet and, (ii) A stale flag to manage the freshness of the Data. ICs provides the following functions: (i) Lookup, (ii) Add, (iii) Remove, (iv) Update freshness, (v) Print.

A base implementation of ICs (CsBase) is provided. It supports FIFO and LRU replacement policies and the cache size can be defined (maxSize).

Experimental unit. Some internal NDN processes that can be imagined to improve efficiency cannot be assimilated to the forwarding strategy, and are not related to native NDN entities (i.e. PIT, FIB, CS). Packet compression described before and Fuzzy logic NDN forwarding [Mastorakis et al. 2018] can be cited as examples of such processes. To provide a clean and flexible way to implement such experimental design without disturbing the NDN base implementation, an eXperimental Unit module (XU) is included. This module is designed to evaluate future AI-based operations such as semantics extraction from names, intelligent routing, etc.

IXu is an abstraction of the eXperimental Unit. The following signals can be emitted by IXu implementations to notify on processing progress: (i) Packet received, (ii) Packet processing begin, (iii) Packet processing end, (iv) Packet processing error and, (v) Packet processing success.

Forwarding. The forwarding module is the main component of our design and it is connected to PIT, FIB, CS and XU. The forwarding process is abstracted in the IForwarding interface which provides the following functions: (i) processLLInterest; process Interest coming from lower layer. (ii) processLLData; process Data coming from lower layer. (iii) processHLInterest; process Interest coming from higher layer (i.e. application layer). (iv) processHLData; process Data coming from higher layer. (v) forwardInterestToRemote; forward Interest to remote Face (e.g. radio). (vi) forwardDataToRemote; forward Data to remote Face. (vii) forwardInterestToLocal ; forward Interest to local application. (viii) forwardDataToLocal ; forward Data to local application. (ix) map-ToMAC ; encapsulate Interest or Data considering the given unicast or broadcast MAC address. (x) onInterest-Timeout; when receiving an Interest timeout signal. (xi) onPrefixExpired; when receiving a prefix expired signal.

The communication with PIT, FIB, CS and XU can be either through message passing or direct module access as provided by OMNeT++. However, the message passing communication allows us to model the processing time of each module separately and independently. The forwarding module implementation is intended to subscribe to PIT, FIB and XU signals in order to handle NdnL3 events.

The current framework implementation includes a base forwarding strategy (Forwarding-Base). It supports the following parameters: (i) ndnMacMapping; code of the NDN-to-MAC mapping to use. (ii) cacheUnsolicited ; whether to cache unsolicited Data packets or not. (iii) forwarding; whether to forward Interests (router) or not (end-device).

4.3.4 Messages and Packets

To represent NDN packets, we use the OMNeT++ message representation in order to provide an easy way to create packets and access their fields. However, for packet-related evaluations, a set of tool functions are provided to generate the TLV representation and to compute the actual size of a packet from the OMNeT++ packet representation. For evaluation purposes, Interest and Data have a common superclass (NdnPacket) which includes non-NDN fields. These fields are of two types: (i) inherited from OMNeT++ Packet class and, (ii) additional fields that include a sequence number, a type (i.e. Interest or Data), a hop count and other fields that are useful for collecting statistics.

4.3.5 Framework Use

The NDN-OMNeT framework is intended to be used to evaluate NDN for low-end IoT solutions, by focusing on both local wireless networks and system state of constrained devices. Although for some evaluations, a simulation cannot serve as a suitable substitute for a real-world testbed, the proposed framework can quickly provide accurate results for different aspects of the solution. Currently, the framework includes some forwarding strategies needed later in this dissertation: Blind Flooding described in the next section, RONR and R-LF described in the next Chapter. Moreover, it can be used for example to evaluate aspects such as hybrid NDN-IP gateways, and the impact of NDN name-to-MAC mapping approaches.

We should note that the current implementation does not support all NDN features. For example, the forwarding process implemented is based only on names, whereas hop limit and lifetime should also be considered.

Having the NDN-OMNeT framework for low-end IoT simulations, we can formulate analytical models and evaluate their accuracy by simulation. Thereafter, models may be used for more accurate analysis of the behaviour. The next section introduces such a model.

4.4 Analytical Model

One purpose of the following model is to highlight the importance of link-layer adaptation for NDN, particularly in constrained wireless networks. For that reason, we model a simple broadcast-based forwarding approach considering a binary-tree topology and content popularity. We describe below the forwarding strategy considered, the assumptions behind the model and its formulation. Then, the model will be evaluated and exploited in the next chapter to understand the relevance of a link-layer adaptation in NDN wireless networks.

4.4.1 Forwarding Strategy Considered

To retain the benefits of flooding/broadcast while reducing overhead and redundancy, a simple mechanism can be used to reduce unnecessary transmissions. We refer to this simple forwarding strategy as Controlled Flooding (CF). Nodes in CF exploit broadcast communications to overhear packets and possibly avoid forwarding some packets. To do so, every node defers a packet transmission with a random delay during which it keeps listening on the shared wireless medium. While waiting, if the node overhears a packet (i.e., Interest or Data) with the same name, it cancels its retransmission.

In practice, Interest and Data transmissions are deferred for Δ_I and Δ_D periods of time respectively. Both Δ_I and Δ_D are computed based on an interval, defer window (dw), from which an integer value is randomly chosen to generate the waiting delays as follows [Amadeo et al. 2015]:

$$\Delta_D = rand[0, dw - 1] \times DeferSlotTime \tag{4.1}$$

$$\Delta_I = rand[dw, 2dw] \times DeferSlotTime \tag{4.2}$$

where $DeferSlotTime$ is a short period of time.

Here, Δ_I and Δ_D are selected in disjoint intervals with $\Delta_I > \Delta_D$ to give higher priority to Data packet transmissions and avoid useless Interest broadcasts. During the Δ_I waiting time, a potential forwarder listens to the channel: if it overhears the same Interest or the requested Data, it cancels its own transmission.

4.4.2 Assumptions and Notation

We consider an IoT deployment with consumer applications requesting content produced by wireless devices. Each IoT device is provided with a single IEEE 802.15.4 inter-

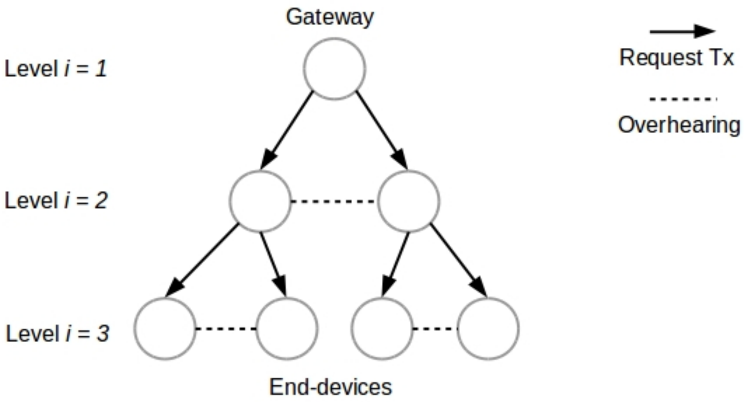


Figure 4.11: Tree topology example with $N = 3$

face and consumers request data through a gateway.

The model assumes ideal physical-layer conditions. The network topology is considered as a full binary tree of depth N, in which the root and the leaves represent the gateway and the end-devices respectively, and the other nodes represent the relay-nodes (see Figure 4.11). The binary-tree topology is assumed to simplify the study, but no host addresses are required to build the topology. Nodes are fixed, sibling nodes can overhear each other, but for the sake of simplicity we assume that no packet is transmitted between them. Consequently, only one path is possible between the gateway and each content producer. Relay-nodes (including the gateway) at the same level have caches of the same size. This means that the cache size is larger in the nodes closer to the gateway, and the gateway has the largest cache size.

The modeled metrics are the following:

- Cost-per-request (CPR). The number of packets transmitted in the network to retrieve some content requested from the gateway.

- Round-trip time per request (RPR). The mean delay time (in *ms*) measured by the gateway from sending an Interest to receiving a matching Data.

Table 4.7: Model variables

Notation	Meaning
$p_k, p_k(i)$	Cache miss probability for class k content at the gateway, at level $i > 1$.
p_t	Probability that two sibling nodes re-transmit the same Interest given that a cache miss has occurred in both nodes.
N	Tree depth.
K	Number of popularity classes.
M	Number of total content items ($m = M/K$ in each class k).
x	Cache size in number of chunks.
$\lambda, \lambda(i)$	Content request rate at the gateway, at level $i > 1$.
λ_k	Content request rate at the gateway for class k.
σ	Average content size in number of chunks.
$q_k, q_k(i)$	Content popularity distribution of requests at the gateway, at level i.
dw	Defer window.
r_I	Time needed to send an Interest over one hop (excluding waiting delays).
r_D	Time needed to send a Data over one hop (excluding waiting delays).
τ	Defer slot-time.
p_f	Probability that the link-layer avoids a collision given that two nodes transmitted an Interest.

4.4.3 Content Popularity

Each relay-node in the network has a cache managed with the LRU replacement policy. We consider a set of M content items equally divided into K classes, each one containing $m = M/K$ content items. Each class represents a different popularity to be requested with probability q_k, $k = 1, 2, ..., K$.

The content requested in our scenario can be considered as Web content which usually follows Zipf distribution [Breslau et al. 1999]. Hence, to model the popularity of content classes we use a Zipf distribution, $q_k = c/k^\alpha$ with $\alpha > 1$ and $c = 1/\sum_{k=1}^{K} 1/k^\alpha$. As content in the modeled application consists of a small amount of data, we consider that each content item is transmitted in one Data packet.

We assume that content items are produced uniquely and uniformly by end-devices located at the leaves of the tree. In other words, each end-device produces the same number of content items of each class, and an item can not be produced by two different devices. The gateway issues the requests (i.e., Interests) originated by consumer applications. The request arrival process is modeled through a Markov Modulated Rate Process (MMRP) of intensity λ. Requests for content in class k are generated according to a Poisson process of intensity $\lambda_k = \lambda q_k$, and the requested content within the class is uniformly chosen among the m different content items in the given class. That is, a given item in class k is requested with probability q_k/m. The notations and their meanings are summarized in Table 4.7.

4.4.4 Model Formulation

We start by defining p_k, $p_k(i)$ and p_t. According to the topological assumptions, a node can cache only content retrieved from its sub-tree. Since content items are uniquely and uniformly produced by end-devices, there is no data duplication among caches at the same level. This allows us to consider all the caches at each level i simply as one cache.

Given this, and with the content request arrival process described previously, at the first level (which corresponds to the gateway in our topology), the stationary miss popularity for chunks of class k, p_k is defined and proven in [G., M., L., and D. 2011] as follows:

$$p_k \equiv p_k(1) \approx exp^{-\frac{\lambda}{m} q_k g x^{\alpha}} \tag{4.3}$$

for relatively large x, where $1/g = \lambda c \sigma^{\alpha} m^{\alpha-1} \Gamma(1-\frac{1}{\alpha})^{\alpha}$

Considering a binary tree with N levels and an MMRP content request process with rate $\lambda(i)$, under the popularity distribution given above, the miss probability at level $i \in [2, N)$ is also defined and proven in [G. et al. 2011] as follows:

$$\log p_k(i) = \log p_k(1) \prod_{l=1}^{i-1} p_k(l) \tag{4.4}$$

For more details on Equations 4.3 and 4.4, including proof and discussions, readers may refer to [G. et al. 2011].

When a cache miss occurs at two sibling nodes, they will both try to forward the Interest after a random delay, as defined in Section 4.4.1. Given the random delays computed in Equation 4.1, the same Interest may be forwarded by both nodes if they choose random numbers with a difference smaller than $s = r_I/\tau$. Hence, the probability that two sibling nodes transmit the same Interest is equivalent to the probability that two random numbers chosen from the interval of length $S = dw + 1$ have a difference smaller than s. This can be formulated as follows:

$$p_t = 1 - \left(\left(\frac{S-2s}{S} \right)^2 + 2 \sum_{i=0}^{s-1} \left(\frac{S-(i+s)}{S^2} \right) \right) \tag{4.5}$$

Here, we can define the CPR for retrieving a class k content item as follows:

$$CPR_k = \sum_{i=1}^{N} \left((1 - p_k(i)) \prod_{j=1}^{i-1} p_k(j) \right) \times \left(2(i-1) + \prod_{l=2}^{i-1} p_t C_k(l) \right) \tag{4.6}$$

where:

$$C_k(l) = 1 + p_f \left((N - l - 1) + \prod_{n=l+1}^{N-1} p_t C_k(n) \right) \tag{4.7}$$

Note that Equation 4.6 models the CPR only for the requests that have been satisfied. That is, $p_k(N) = 0$.

Equation 4.6 is obtained based on the following approach. As the content can be found at any level from 1 to N, the cost is defined as a weighted sum of the transmitted packets associated to each level i. Then, the weights correspond to the cache hit probability $(1 - p_k(i))$ at level i given that a cache miss occurred at all the previous levels (i.e., 1 to $i - 1$).

For every possible level i, the number of packets is composed of two parts: $2(i - 1)$ corresponds to the number of packets transmitted along the path from the gateway to the level-i device, plus the number of packets transmitted if the sibling of each previous node (from level 2 to $i - 1$) has also transmitted the Interest, which has a probability p_t of occurring for each pair of siblings.

Here, Equation 4.7 assumes that when the brother of a node (at level l) transmits an Interest, the cost can be recursively computed using the same approach as Equation 4.6 in its sub-tree (i.e. from level $l + 1$ to N). The only difference is that, on this side of the network, we directly consider the path from level l to the leaf level N, since the requested content has already been found elsewhere (according to Equation 4.6) and there is no data duplication. However, in each sub-tree, the first Interest is always transmitted, but the number of transmissions recursively computed is subject to the probability that no collision occurs between the first sibling nodes of the sub-tree (i.e., p_f).

Following the same approach, we define the mean RPR for a class k content item as follows:

$$RPR_k = \sum_{i=1}^{N} \left((1 - p_k(i)) \prod_{j=1}^{i-1} p_k(i) \right) R_i \tag{4.8}$$

where $R_i = (i-1)(r_I + r_D + \delta_I + \delta_D)$.

Similarly to CPR, as the content can be found at any level from 1 to N, the mean RPR is a weighted sum of the total time R_i required to send the Interest and get the Data associated to each level i. The weights correspond to the cache hit probability $(1 - p_k(i))$ at level i given that a cache miss occurred at all the previous levels (i.e., 1 to $i-1$).

Here, R_i is obtained by multiplying the number of hops $(i-1)$ for level i by the total delay needed to send an Interest and get Data; which includes waiting delays $(\delta_I + \delta_D)$ and time-on-air $(r_I + r_D)$.

When two sibling nodes delay their transmissions, the node with the shortest delay will transmit the packet first. Furthermore, the round-trip delay measured by the consumer (e.g., gateway) will be affected by the shortest waiting delay computed at each level. Hence, the global estimation of δ_I and δ_D is not the half way between the lowest and the highest values (e.g., $dw/2\tau$). To approximate the values of δ_I and δ_D, we consider the mean of the lowest half of $[0, dw-1]$ and $[dw, 2dw]$ intervals respectively. This gives us $\delta_I = ((3dw)/4)\tau$ and $\delta_D = ((dw-1)/4)\tau$.

4.5 Conclusion

Generally speaking, the tools presented in this chapter are necessary to test and evaluate NDN designs in low-end IoT. First, the testbed shows the feasibility of the proposed NDN-802.15.4 architecture and enables NDN in real-world IoT applications. It is also useful in providing some preliminary measurements and empirical comparison with IP-based solutions. For example, the deployment may show the lightweight and simplicity of NDN implementations for IoT. Moreover, such a deployment is ready to host most IoT Proofs-of-Concept to demonstrate NDN-based solutions proposed by startups, for instance. Second, the simulation framework enables NDN in the OMNeT++ simulator, which is very popular in the IoT community. Positive feedback from the OMNeT++ community has been already collected regarding the NDN-OMNeT framework. Nevertheless, this tool will allow us to quickly evaluate our forwarding approaches in wireless networks and compare them to existing propositions, under complex scenarios and various parameters. This will be illustrated through the extensive use of simulations in the next chapter. Third, the mathematical model we formulate is intended for analyzing the NDN behaviour in wireless mesh networks. As we show in the next chapter, one objective of the model is to study whether caching can attenuate the transmission overhead generated by broadcast communications. Another objective is to study the relevance of investigating the link layer in NDN wireless forwarding strategies. Although the current model is simple as it considers only binary-tree topology networks with a simple forwarding strategy, it is the first one that models NDN in wireless mesh networks.

In short, the combination of these tools will help us to study, understand, take appropriate design choices and then evaluate our approaches with different levels of accuracy, as is shown in the next chapter. Moreover, to tackle the problem of lightweight wireless forwarding with NDN, in the next chapter we bring together all the concepts and tools described throughout the previous chapters.

Chapter 5

NDN Wireless Forwarding in Low-end IoT

5.1 Introduction

To deploy NDN in IoT devices, one of the main features to support is NDN forwarding in a low-rate wireless mesh network, such as IEEE 802.15.4 networks. NDN wireless forwarding strategies are generally based on a broadcast-and-learn mechanism. This approach uses a phase in which Interests are broadcast until the content is found, then subsequent requests are forwarded more accurately based on the information learned. Therefore, the use of broadcast is necessary in NDN wireless networks. Moreover, evaluation results reported later in this chapter suggest that broadcast is the most compliant pattern to handle content dissemination, mobility and caching.

Lightweight forwarding strategies for wireless NDN environments are rare in the literature. This chapter introduces our solutions for lightweight forwarding in constrained wireless mesh networks in general, and IEEE 802.15.4 in particular. To that end, in Section 5.2 we first study the main forwarding approaches for NDN existing in related literature. Second, as broadcast-based strategies are simple and efficient in disseminating data, and are compliant with the native communication pattern of ICN/NDN, we have to find out how we can use broadcast while reducing overhead, memory and processing requirements. For that, in Section 5.3 we study the impact of the broadcast pattern in constrained wireless networks in terms of data redundancy, number of packets transmitted, data availability and decision accuracy. Our study considers two scenarios: (i) a simple wireless network with a binary-tree topology and fixed nodes and, (ii) a complex network with a grid topology and mobile nodes. This study ends with a set of design guidelines for a lightweight broadcast-based wireless forwarding strategy.

Finally, with all the necessary considerations, in Sections 5.4 and 5.5 we propose two solutions designed at two different levels. At the link-layer level, we propose Named-Data CSMA (ND-CSMA), an adaptation of the CSMA algorithm used in the IEEE 802.15.4 specification to handle NDN communications in a simple topology. At the network-layer level, we propose a Reinforcement-based Lightweight Forwarding (R-LF) strategy, a lightweight forwarding mechanism based on reinforcement learning to support complex networking scenarios.

The idea behind both approaches is to make broadcast as accurate as unicast, in terms of forwarding decisions and the number of transmitted frames. Therefore, we focus on designing lightweight trade-off techniques that can maintain satisfactory performance in different communication scenarios and network configurations.

5.2 NDN Forwarding in Wireless Networks

As mentioned before, NDN forwarding approaches in wireless networks are based on a broadcast-and-learn mechanism. In practice, an Interest flooding phase is used to discover and learn about content sources (i.e. producers or caches), and subsequent Interests are forwarded more accurately based on the information learned. That is, the flooding phase is a sequence of Interest broadcasts performed by relay nodes to discover the source of a content item.

Although NDN wireless forwarding has been mostly investigated in MANETs, mechanisms proposed so far represent an interesting starting point for any wireless forwarding strategy with NDN.

According to [Amadeo et al. 2015], forwarding approaches for NDN in ad-hoc networks can be based on either blind forwarding or aware forwarding. Blind forwarding consists in simple schemes used to limit the impact of the broadcast/flooding in the network. Aware forwarding are schemes that use additional information and processing about the content source(s) and/or the neighborhood to allow more accurate and more efficient forwarding decisions. Typically, more packets have to be exchanged between nodes to collect forwarding information. However in constrained networks, aware forwarding solutions are not systematically better that blind forwarding mechanisms. Indeed, often the performance offered by aware forwarding comes at the cost of additional overhead, memory and complex processing. In other words, a blind forwarding mechanism can be more appropriate in some scenarios. For this reason, we explore both approaches in our proposals.

At this point of the study, it is useful to distinguish between two types of mobility introduced by the content-centric aspect of IoT applications and emerging Internet applications. The physical mobility of nodes and the logical mobility of data are two of the main causes of dynamics in multi-hop wireless networks. While physical mobility results from moving hosts, logical mobility can be related to the case when chunks of the same content item are produced by more than one host, for example.

In this section, we introduce the main approaches proposed in the related literature to handle NDN forwarding in wireless ad hoc networks. These solutions cover both blind and aware forwarding. Traditional wireless forwarding approaches for IP networks are not considered here for several reasons. First, they are based on IP addressing, and rely on point-to-point communications. This does not allow IP-based protocols for ad-hoc networks to deal with logical mobility directly, and their performance depends on the level of physical mobility in the network. Second, typical IP routing solutions for wireless networks are proactive. For example, RPL creates a logical topology and updates routes using host addresses. This approach is completely different from the NDN wireless forwarding and thus is difficult to adapt to NDN or to compare with.

Nevertheless, in order to have an IP-based protocol as a reference in our study, we consider the the Ad-hoc On-Demand Distance Vector protocol (AODV) [Das et al. 2003]. Therefore, we start by introducing this well-known routing protocol for ad hoc networks. Although it may be considered something of an intruder in this section, it is based on a mechanism that creates routes on-demand and thus resembles the NDN receiver-driven model. Thus, it is worth describing it and using it later in comparative evaluations.

5.2.1 AODV: An Intruder With a Similar Model

In the wireless IP world, the closest approach to NDN forwarding is AODV, a routing protocol for mobile ad hoc networks. This protocol communicates via UDP/IP to discover and maintain unicast routes between nodes within the wireless network. The AODV protocol establishes routes on the fly when they are needed and maintains them as long as they are being used. The routing approach used by AODV consumes low energy, low memory and does not require large computing power, which makes it easy to deploy on small mobile devices. AODV is designed for networks with tens to thousands of mobile nodes with relatively high mobility, while reducing network overhead to improve scalability and performance. AODV uses three types of messages: (i) Route Requests (RREQs), (ii) Route Replies (RREPs) and, (iii) Route Errors (RERRs). Typically, RREPs and RERRs are not blindly forwarded as they are destined for a particular host, but RREQs have to be flooded throughout the network to reach the destination. The range of RREQs flooding is controlled by the TTL in the IP header.

Figure 5.1 illustrates an example of AODV route discovery. The dashed arrows represent a one-hop communication between two neighbor nodes. To reach a destination node located at a distance of several hops, a source node has to find a route. For that, the source node broadcasts an RREQ message which carries information such as the source, the destination, the TTL and a Sequence Number that uniquely identifies the message. In the example, Node A wishes to communicate with Node J. To discover a route to Node J, Node A sends out an RREQ message. The RREQ is heard by Node A's neighbors which are Nodes B, C and D. When Node A's neighbors receive the RREQ message, they have two choices; if they know a route to Node J they can send a unicast RREP message back to Node A, otherwise they will rebroadcast the RREQ to their neighbors. The message is re-broadcasted hop-by-hop until it reaches Node J or until its TTL is over. In the example, a route to Node J is found by Node D, which finally replies to Node A's RREQ by an RREP. Nodes B and C on the other hand do not find a route to Node J, but in another case they may find a route and reply to Node A.

To avoid packet-loops and handle route freshness, nodes use sequence numbers in messages and each node records the sequence number of all the nodes it talks to. Every time a node sends out any type of message it increases its own sequence number. That is, a higher sequence number means a fresher route. In the example, Node B may forward another RREP to Node A. If Node A notices that the route in the RREP is more recent than the route in its table, it may replace the route it currently has with the route in Node B's RREP.

To cope up with network dynamics resulting from mobility, AODV uses RERR messages. When a node receives an RERR, it removes all the routes impacted by the error. RERRs can be used in different cases. For example, if a node receives a unicast packet that it is supposed to forward but it does not have a route to the destination. This situation indicates that there are nodes that have wrong route information about that destination. For another example, a node detects that it cannot communicate with one of its neighbors. When this happens, it looks at the route table for routes that use that neighbor as a next hop and marks them as invalid, and sends an RERR to invalidate those routes.

Although the communication scheme of AODV is close to that of NDN, it is still based on host addresses and operates on explicit route between nodes. Therefore, AODV is largely affected by the physical mobility of the nodes. To handle nodes mobility, AODV has to track and update the state of some of the links in the network, which causes more communication and processing. Moreover, AODV does not support logical mobility of the data.

5.2.2 CF: The Basic NDN Forwarding

Blind Flooding all packets on a broadcast medium for every content request (i.e., Interest/Data) may cause high overhead and packet redundancy. Controlled Flooding (CF) is a simple improvement of Blind Flooding (BF) with a packet suppression technique. To keep the benefits of flooding while reducing the overhead, nodes exploit broadcast communications to overhear packets and possibly avoid forwarding some packets. To do so, every node defers a packet transmission with a random delay during which it keeps listening on the shared wireless medium. While waiting, if the node overhears a packet (i.e., Interest or Data) with the same name, it cancels its retransmission.

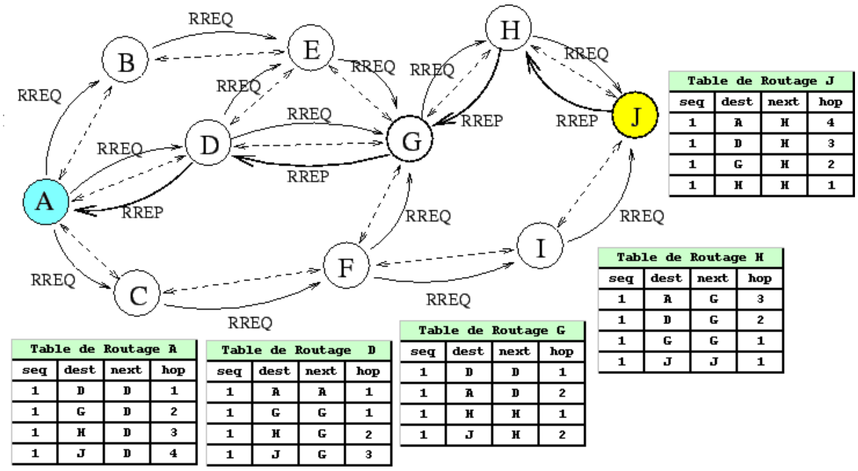


Table de Routage J			
seq	dest	next	hop
1	A	H	4
1	D	H	3
1	G	H	2
1	H	H	1

Table de Routage H			
seq	dest	next	hop
1	A	G	3
1	D	G	2
1	G	G	1
1	J	J	1

Table de Routage A			
seq	dest	next	hop
1	D	D	1
1	G	D	2
1	H	D	3
1	J	D	4

Table de Routage D			
seq	dest	next	hop
1	A	A	1
1	G	G	1
1	H	G	2
1	J	G	3

Table de Routage G			
seq	dest	next	hop
1	D	D	1
1	A	D	2
1	H	H	1
1	J	H	2

Figure 5.1: AODV route discovery example [Wikipedia b]

In practice, Interest and Data transmissions are deferred for Δ_I and Δ_D periods of time respectively. Both Δ_I and Δ_D are computed based on an interval, defer window (dw), from which an integer value is randomly chosen to generate the waiting delays, as defined previously in 4.1 and 4.2 respectively.

Δ_I and Δ_D are selected in disjoint intervals with $\Delta_I > \Delta_D$ to give higher priority to Data packet transmissions and avoid useless Interest broadcasts. During the Δ_I waiting time, a potential forwarder listens to the channel: if it overhears the same Interest or the requested Data, it cancels its own transmission. An example of CF applied in a binary-tree topology network is given in Figure 5.2.

Note that the CF strategy is mainly used in MANETs with IEEE 802.11 technologies as it requires quite a high bandwidth to achieve good performance. However, CF can be envisioned over IEEE 802.15.4 when extremely low latency is not required.

5.2.3 RONR: An Improvement With Unicast

One forwarding strategy proposed for constrained IoT networks is the Reactive Optimistic Name-based Routing (RONR) [Baccelli et al. 2014]. It has been deployed in IEEE 802.15.4 networks with the objective of reducing overhead compared to BF and CF, while keeping a satisfactory data retrieval rate and a minimum forwarding state in nodes.

RONR resembles AODV; after retrieving the first content by Interest broadcast, a node

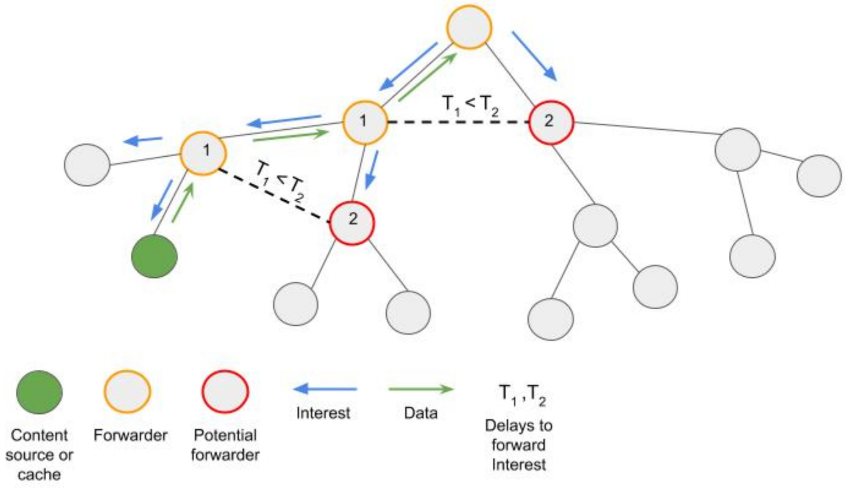


Figure 5.2: CF example in a binary-tree network

keeps a temporary FIB entry in order to avoid flooding subsequent Interests. The FIB entry binds the content name prefix to the content source MAC address. The nodes also use the Interest source MAC address to forward back the Data packet. This is equivalent to the IUDU mapping introduced in Chapter 2. When the FIB entry for the requested content does not exist, or it is deleted, the flooding phase is performed again to discover another content source.

For example, in our NDN-802.15.4 testbed, after flooding an Interest for content */farm/cow/21/temperature*, nodes on the reverse path traversed by the Data packet will create a temporary FIB entry for */farm/cow/21*. Thus, subsequent Interests for content */farm/cow/21/movement* can be forwarded via unicast using the established path, instead of flooding.

Using a mapping between unicast MAC addresses and NDN name prefixes, RONR significantly reduces network overhead. Measurements collected in a real-world IoT deployment show that RONR can decrease the number of radio transmissions by about 50% compared to BF. More importantly, the number of broadcast transmissions is reduced because only the first Interest packet of a content item is flooded, while subsequent Interests are unicast. Consequently, RONR supports larger networks than BF and also matches devices energy requirements better. Moreover, RONR does not require any control traffic and occupies only some temporary FIB entries, in addition to PIT entries common to all

NDN forwarding mechanisms.

However, RONR suffers from important limitations. First, using addresses makes the communications point-to-point, which is not compliant with NDN according to some ICN-enthusiasts. Second, it does not take advantage of the broadcast naturally offered by wireless links and natively supported by NDN. That is, RONR has no means to select the shortest path to retrieve contents. Third, as its name indicates, RONR is optimistic because it assumes that the node which contains a content chunk contains the whole content item. However, this is not always the case, particularly with caching. Moreover, one can easily see that mapping names to unicast addresses can not support node mobility and simultaneous data flows in the network.

5.2.4 LFBL: A Better Use of Delayed Transmissions

The Listen First Broadcast Later (LFBL) [Michael et al. 2010] technique adopts the same listen-before-transmit approach used in CF, but with some improvements. First, instead of systematically forwarding an Interest, each relay node decides whether it is an eligible forwarder or not based on the distance to the data source. The distance can be propagated through the number of hops traversed by packets (see Figure 5.3). Second, if a node is an eligible forwarder according to the distance comparison, it delays the Interest transmission for a period that is proportional to its eligibility; instead of a random delay. In other words, a potential forwarder waits and listens if a more optimal node forwards the packet first. The closer is the forwarder to shorted is the waiting delay. If a potential forwarder does not hear the same Interest transmission during the waiting delay, it forwards the packet itself.

LFBL does not require any explicit knowledge about the network topology. To evaluate the distance, each content source in the network is uniquely identified and each node inserts its distance to the content source before sending an Interest.

A node that receives an Interest has to first determine if it is an eligible forwarder. This decision is based on whether the node is closer to the destination than the previous node that sends the Interest. If the node is eligible, it has to decide how long to wait before forwarding the Interest. During the waiting delay, the forwarder listens to know whether

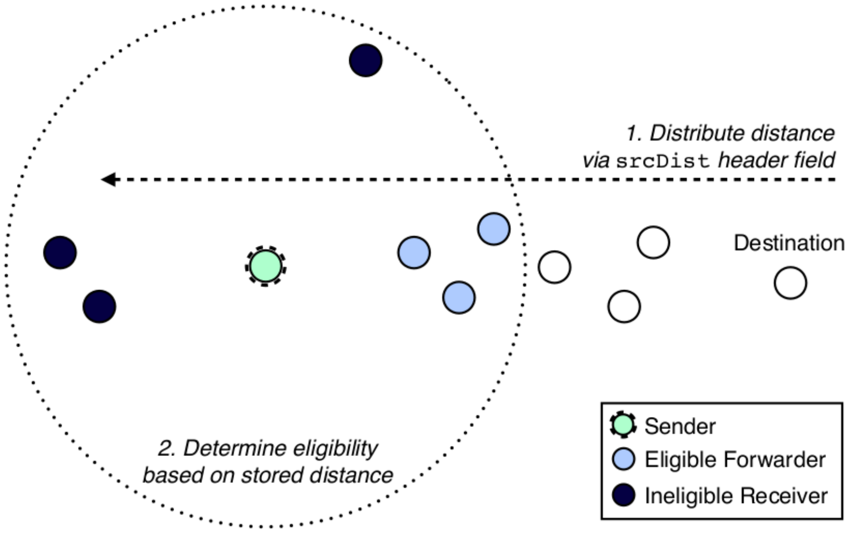


Figure 5.3: Representation of eligible forwarders [Michael et al. 2010]

another node performs the forwarding task. The waiting delay is computed in such a way that the closer is the node to the source, the shorter is the waiting time. As in CF, the waiting delay also contributes to avoid the risk of collisions; by reducing the chance that two eligible forwarders will forward the Interest simultaneously and cause a collision.

LFBL has been evaluated by simulation considering a IEEE 802.11 network. Evaluations are based on networks of 100 nodes randomly placed in an area of 1500 by 1500 meters generating a bidirectional flow composed of a requester and a responder.

The following metrics were considered: round-trip delay, delivery ratio, overhead, and total data transferred. The round-trip delay is the amount of time elapsed from when a request is sent by a requester until it receives a response. The delivery ratio is the total number of packets received (by any node, requester or responder) divided by the number of packets sent. Overhead corresponds to the portion of transmissions used for something other than the successful delivery of application-layer data. Total data transferred is the sum of all bytes received by all requesters over the entire duration of the simulation.

The results obtained show that LFBL can outperform AODV for all the metrics. Particularly, under high dynamics, LFBL can deliver up to five times more packets than AODV with comparable overhead. That is, LFBL performs extremely well in highly dynamic environments, independently of node mobility and data location. Moreover, the eligibility-based forwarding allows LFBL to generate reduced overhead by avoiding useless transmissions and propagating packets fast. However, LFBL nodes maintain distance tables, which requires additional memory in the NDN process, and endpoint identification is used for source and destination nodes, which slightly evokes the host-based communication model.

5.2.5 NAIF: A Different Approach

Instead of delayed transmissions, the Neighborhood-Aware Interest Forwarding (NAIF) [Yu et al. 2013] uses a forwarding rate adjustment to refine forwarding decisions. Given a relay node in the network, its forwarding rate is the ratio of Interests it may forward among the number of Interests it receives. Without explicit communication, each node exploits broadcast overhearing to collect forwarding statistics by monitoring Interests and Data packets forwarded by itself and one-hop neighbors. These forwarding statistics are collected for each name prefix during an update interval, and are as follows: (i) s_{int}, the number of distinct Interest packets sent, (ii) r_{data}, the number of distinct Data packets received, (iii) c, the number of distinct Interest packets cleared, according to the PIT, (iv) r_{int}, the number of distinct non-cached Interest packets received. A non-cached Interest is defined as an Interest requesting a Data that is not cached at the given node. The statistics are used to periodically update the retrieval rate and the forwarding rate for each name prefix. After each update, the forwarding statistics are reset.

The NAIF forwarding decision is based on two forwarding parameters computed locally at each relay node: (i) The data retrieval rate R, which is the ratio of the number of Data packets successfully retrieved to the number of Interests sent. (ii) The forwarding rate F, which is the fraction of incoming Interests that a given node can forward.

The data retrieval rate is used to measure the effectiveness of a node in retrieving Data packets for a name prefix. At the end of an update interval of duration t, the data retrieval rate is computed:

$$R_t = \frac{c}{S_{int}} \tag{5.1}$$

Here, c corresponds to the number of satisfied Interests and cleared from the PIT. The forwarding rate is defined as the fraction of Interest packets sent by a node. At the end of an update interval t, the forwarding rate during t is:

$$F_t = \frac{s_{int}}{r_{int}} \tag{5.2}$$

The purpose of the forwarding rate adjustment is to share the workload among neighboring nodes. At a given node, when the neighborhood is efficiently retrieving Data packets for a given name prefix, the node may reduce its own forwarding rate. When a node drops an Interest and relies on its neighbors to send it, this Interest is considered missed if the retrieval of the corresponding Data is unsuccessful. Assuming the number of missed Interests, δ is known, the forwarding rate is adjusted by:

$$\tilde{F}_t = \begin{cases} \min(s_{int} - 1/r_{int}, \tilde{F}_{t-1} - \alpha) & if\ \delta_{t-\beta} = ... = \delta_{t-1} = \delta_t = 0 \\ (s_{int} + \delta_t)/r_{int} & otherwise \end{cases} \tag{5.3}$$

If there are no missed Interests in the previous intervals, the node gradually reduces its forwarding rate. Otherwise, the node increases its forwarding rate based on δ which each node locally estimates. If a node received an Interest and dropped it, and the node does not overhear the corresponding Data later, the Interest may have been missed. Therefore, the number of missed Interest is obtained by:

$$\delta = (r_{int} - s_{int}) - (r_{data} - c) \tag{5.4}$$

Then, the NAIF algorithm is applied only to Interest forwarding at relay nodes and consists of two phases. The first phase eliminates the ineligible forwarders based on their

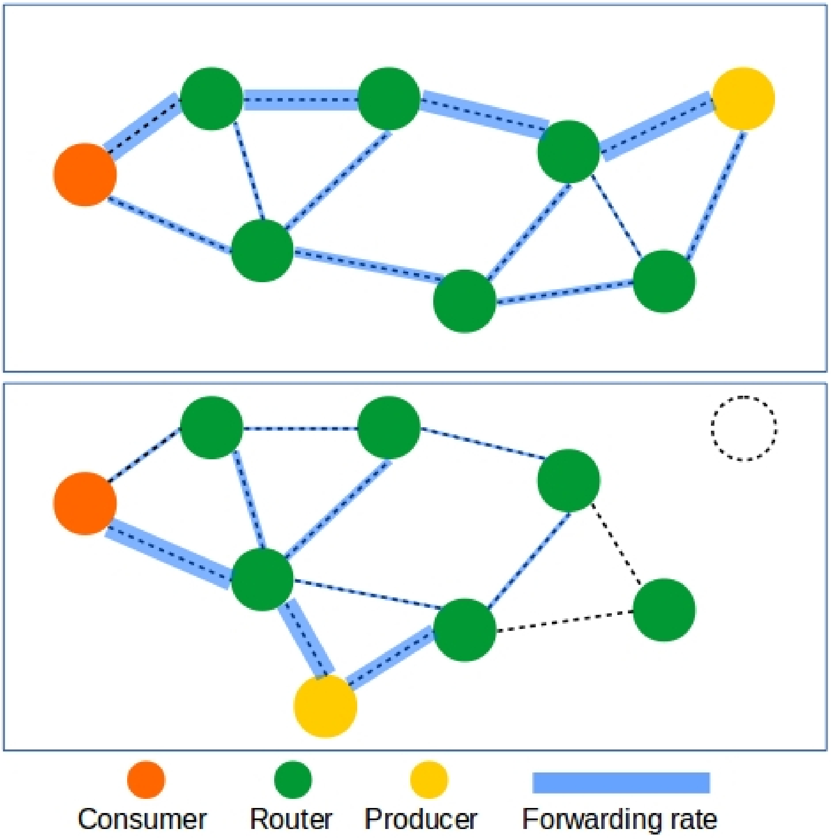


Figure 5.4: Forwarding rate illustration in NAIF

data retrieval rate for the requested content and their distance to the data consumer. That is, if a node has a low data retrieval rate or it is too far away from the data consumer, the node drops the Interest. At the second phase, the eligible forwarder probabilistically drops the Interest based on the updated forwarding rate (F) for the requested content.

Figure 5.4 gives an illustrative example of the forwarding rate adjustment used in NAIF. When the data producer moves, the each node on the new path detects that the number of unsatisfied Interests increases and adjust its own forwarding rate accordingly.

NAIF has been evaluated in a network environment similar to the one used with LFBL. Comparative evaluation results show that NAIF achieve a 30-60% higher delivery ratio in multi-consumer data retrieval scenarios than LFBL. Moreover, NAIF significantly reduces bandwidth usage by half while maintaining round-trip time comparable to BF. Furthermore, although LFBL performs well in single-consumer data retrieval scenarios, BF and NAIF are more robust in multi-consumer scenarios.

Overall, good performances are obtained with NAIF in terms of delivery ratio, but the round-trip time is quite high. Since NAIF is based on a purely statistical method, nodes need to first get enough information and reactively adapt to changes in the situation. This makes the approach slow to converge under dynamic configurations in which small and independent content items are requested. Thus, it is more suitable for downloading large content (e.g. files) but is not so suitable for chunks of small content items of IoT applications. According to the reported results, NAIF achieves almost the same performance as the BF strategy under high mobility scenarios. More importantly, the NAIF algorithm greatly relies on statistics that require nodes to listen to every communication in their neighborhood. This approach is not reasonable in constrained networks with reduced bandwidth and limited energy.

5.2.6 Q-routing: A Search-and-Learn Approach

Routing mechanisms in wireless mesh/ad-hoc networks can broadly adopt a structure-based or a structure-less approach. Basically, structure-based protocols operate on logical routing topologies (e.g., RPL) or establish explicit routes between endpoints (e.g., AODV). To achieve that, an initialization phase is frequently used and a maintenance of routes or topologies is required to keep routing information updated. This becomes even more important in the case of dynamic networks.

Instead of maintaining routing structures, structure-less protocols use numerical values, such as a *cost-to-go*, to derive routing decisions. Cost-to-go values can be carried in packets as thus be propagated and updated implicitly throughout the hosts. Typically, hosts are mapped to cost-to-go values that express their eligibility to reach an objective. Cost-to-go can express any value that has to be minimized; for example, if the shortest path is the routing objective, cost-to-go is the minimum number of hops from a node to the sink node.

Search-and-learn protocols such as Q-routing [Boyan and Littman 1993] presented here, and flooding protocols such as Constrained Flooding [Zhang and Fromherz 2006] described further are structure-less protocols.

In Machine Learning, reinforcement learning basically makes an agent able to learn by interacting with its environment, without training on a dataset or help from an expert. Q-learning [Watkins and Dayan 1992] is an efficient model-free reinforcement learning framework in which an agent learns, by trial-and-error, the best action to perform according to its current state. During the exploration phase, the agent tries an action a and gets a reinforcement value. The agent uses the reinforcement to estimate which action is the best in each state. The reinforcement is formed by the observed reward and the estimation of the best future state-value. The incremental algorithm that updates the value of a state s with an action a is expressed by the following equation:

$$Q(s_i, a_i) = (1-\alpha)Q(s_i, a_i) + \alpha \left(r_i + \gamma \max_a Q(s_{i+1}, a)\right) \tag{5.5}$$

Where r is the reward for the current state. α is the learning rate that controls how much the new state-value overrides the old value, γ is the discount factor that determines the importance of the optimal future reward (i.e. $Q(s_{i+1}, a)$). Each evaluated state-action combination is stored in a two-dimension Q-values table. The exploitation phase uses the learned values to construct a policy that chooses the best actions to reach the goal.

The Q-learning framework was adapted for routing operations long ago (i.e., 1994). The idea is to use Q-learning to learn a representation of the state of the network through Q-values, and then these values are used to make forwarding decisions.

Each node x in the network represents its own view of the network state through its Q-table Q_x.

Given this representation, the action a at node x is to choose the neighbor y such that it takes minimum time for a packet to reach node d.

That is, the action space corresponds to the possible next-hop nodes (i.e., neighbors), and the states correspond to the destination nodes.

In Q-Routing, each node x maintains a table of Q-values $Q_x(y, d)$, where $d \in V$, the set of all nodes in the network, and $y \in N(x)$, the set of all neighbors of node x. To learn a path about a destination d, node x tries a next hop y among its available neighbors. Then,

x immediately gets the shortest remaining trip time as estimated by y, and evaluates the reward as the measured trip time from x to y. The future state value is returned by y and corresponds to its minimal learned trip time to the destination d, obtained through its own trial-and-error process. The Q-values table plays then the role of a routing table and is checked to take forwarding decisions.

In the steady state, when the Q-values in all the nodes represent the true state of network, the Q-values of neighboring nodes x and y should satisfy the following relationships:

$$Q_x(y,d) \leq \delta_y + Q_y(z,d) \forall y \in N(x) and \forall z \in N(y) \tag{5.6}$$

where δ_y is the round-trip delay (including time in queue) measured between x and y.

After sending a packet through node y, node x gets the minimum estimation from y estimates the new learned value as follows:

$$Q_x(y,d) = (1-\alpha)Q_x(y,d) + \alpha(Q_y(\hat{z},d) + \delta_y) \tag{5.7}$$

where $Q_y(\hat{z},d)$ is the lowest round-trip delay estimated by node y (i.e., equivalent to $\min Q_y(z,d)$).

The update rule given by Equation 5.7 guarantees that if the old learned value satisfies the inequality (Equation 5.6), then its updated value also satisfies it. This gives the following property:

$$Q_x(y,d) \leq \delta_y + Q_y(z,d) \Rightarrow Q'_x(y,d) \leq \delta_y + Q_y(z,d) \tag{5.8}$$

where $Q'_x(y,d)$ is the updated value of $Q_x(y,d)$.

This property proven in [Shailesh Kumar 1998] using the Q-learning update properties and initial Q-routing conditions.

5.2.7 Constrained Flooding: A Paradigm-agnostic Approach

Constrained Flooding [Zhang and Fromherz 2006] is very similar to Q-routing. However, we consider it as paradigm-agnostic since it does not use any host identification or

Table 5.1: Summary of some NDN wireless forwarding approaches

Name	Approach principle	Pros	Cons	Evaluation
CF	Interest flooding Random-delayed transmissions	Satisfaction rate Mobility support	High overhead Packet redundancy	Simulation (IEEE 802.11)
LFBL	Proportional-delay transmission Content source identification	Low round-trip time Low overhead	Distance table Node identification Mobility	Simulation (IEEE 802.11)
NAIF	Adaptive Forwarding rate	Mobility support Satisfaction rate	High round-trip time High overhead	Simulation/Emulation (IEEE 802.11)
RONR	Interest flooding Prefix to MAC address mapping	Low round-trip time Low overhead	Mobility NDN potential limited by unicast	Real-world (IEEE 802.15.4)

point-to-point communication. Thus, it can be used for IP as well as NDN networks. It has been proposed as a forwarding solutions to route packets to a single node (e.g., sink) in Wireless Sensor Networks. In Constrained Flooding, each node takes advantage of broadcast to overhear packets and to learn by reinforcement its ability to forward packets. Each packet carries the cost value learned by its sender node. Upon receiving a packet, the potential relay uses its own learned cost to decide whether to forward the packet or not; if so, the delay time to wait before transmitting is computed based on the difference between the relay's cost and the sender's cost (contained in the packet).

However, this approach was designed to handle simple sensors communication toward one destination, which makes it unsuitable for complex communication modes introduced by IoT deployments. Moreover, forwarding with NDN is not based on host addresses, may involve multiple data flows and caching. That is, improvements are required, but feasible, to build a constrained flooding technique for NDN.

5.2.8 Summary

Table 5.1 summarizes the main forwarding approaches presented in this section with their advantages and disadvantages.

Given that most of solutions for wireless NDN forwarding use broadcast, we need to study the relevance of the broadcast pattern in NDN constrained networks. This study is reported in the next section.

5.3 Broadcast in Constrained Networks

To investigate the advantages and disadvantages of the broadcast in constrained networks, we separate our study into two aspects. First, a simple scenario in which we study the CF strategy in a tree-topology network, using the analytical model presented in the previous chapter. The purpose is to find if there are reasonable conditions under which a simple broadcast forwarding strategy can be used. Furthermore, we aim to establish which aspects have to be improved in order to design an efficient forwarding scheme in a simple network topology over IEEE 802.15.4. Second, a more complex scenario in which we study different NDN-to-MAC mappings and their impact on communication performance in a grid-topology network with mobile nodes. The objective here is to understand the mechanisms needed to build an efficient forwarding strategy based on broadcast. For that, we use the NDN-OMNeT simulation framework. Finally, a summary of what we learn through this study is presented, followed by some guidelines for the propositions formulated later.

5.3.1 Simple Networks: Tree Topology

In this section, we assess the accuracy of the model formulated in the previous chapter, and we study the impact of broadcast communications in a simple NDN network. To do so, we simulate the CF strategy with the NDN framework for OMNeT++ [Abane et al. 2018].

For the reader's convenience, we recall in Table 5.2 the relevant model parameters introduced in the previous chapter.

We consider a binary-tree topology of depth $N = 4$. The gateway requests content from a total of $M = 3000$ items, distributed in $K = 50$ classes of decreasing popularity, each one with $m = 60$ items. Different popularity distributions have been simulated with $\alpha \in \{1.5, 2, 2.5\}$. The request rate at the gateway is $\lambda = 1$ request/s. Each simulation result corresponds to a run of 10 hours.

Each Interest packet has a size of 30 bytes and each Data packet 90 bytes. We set up a cache of size $x = 300$ packets at each level of the tree. That is, the nodes at each level

Table 5.2: Evaluation parameters

Parameter	Meaning
N	Tree depth.
K	Number of popularity classes.
M	Number of total content items ($m = M/K$ in each class k).
x	Cache size in number of chunks.
λ	Content request rate at the gateway.
α	Zipf distribution parameter for content popularity ($\alpha > 1$).
dw	Defer window used in the CF strategy.
τ	Defer slot-time.
r_I	Time needed to send an Interest over one hop (excluding waiting delays).
r_D	Time needed to send a Data over one hop (excluding waiting delays).
p_f	Probability that the link-layer avoids a collision given that two nodes transmitted an Interest.

have to totalize approximately 26 kb of RAM for caching. According to the measurements reported in the previous chapter, this is feasible considering Arduino DUE at lower levels (e.g., level 1 and 2), and Arduino UNO at higher levels (i.e., close to leaves) as the number of devices is higher so every device stores only a little part of the 300 packets.

Preliminary simulations have been used to set optimal values at $dw = 127$ and $\tau = 0.032\mu s$. We also measured $r_I = 1.36ms$, $r_D = 3r_I$ and $p_f = 0.8$ with preliminary simulations.

The Interest satisfaction rate is reported in Table 5.3. We observe that in our configuration, $dw = 127$ always achieves better Interest satisfaction rate than 255. The reason is that 127 is low enough to make relay nodes transmit more packets and explore the network without being too low to create a lot of collisions. However, the remaining results show that this Interest satisfaction rate is achieved at the cost of much more transmissions than $dw = 255$.

Figures 5.5, 5.6 and 5.7 show the CPR according to content popularity for $\alpha = 1.5$, 2 and 2.5 respectively. According to the results, the value of α has in impact on the efficiency of NDN. In fact, small values of α reduce popularity difference between classes, which introduces more diversity in the requests and thus increases the cache miss rate and CPR. Inversely, when α is high (e.g., 2) applications frequently request the most popular content classes, which takes advantage of caching and reduces the CPR.

The model is also affected by α but can accurately predict the CPR according to popularity classes. The highest discrepancies between the model and simulations are observed for higher values of α. The reason is that a cache miss is more likely to occur when α is higher, which leads to more transmissions. Since $dw = 127$ is not high enough to avoid all redundancies, the behavior of the nodes becomes more dependent on the link-layer, which is not included in the model.

As a reference, we represent the CPR for a perfect-unicast scenario, which refers to the CPR expected in the tree if a host-based routing protocol is used instead of NDN. We note that NDN with CF outperforms perfect-unicast concerning the most popular content. This shows that transmission overhead induced by broadcast can be attenuated by small caches in the presence of popular content. Moreover, this attenuation can outperform the unicast communication pattern, which is here theoretical as it does not include route discovery/maintenance cost.

Figures 5.8, 5.9 and 5.10 report on the mean RPR for $\alpha = 1.5$, 2 and 2.5 respectively. The same type of observations can be made as for CPR, but a greater dissimilarity is observed between the model and simulations. The reasons are the same as for CPR, with an additional fact related to medium access time. As mentioned above, dw is not high enough to avoid redundant packet transmissions. Hence, the link-layer has to resolve more medium access contentions, leading to less accuracy in our model. This can be confirmed by observing raw simulation results (i.e., blue dots) which present higher scatter as α gets higher.

Figures 5.11 and 5.12 show the CPR and RPR respectively for $\alpha = 2$ with $dw = 255$. We observe that CPR becomes better when a higher value of dw is used. Compared to $dw = 127$, up to two transmissions per request are saved for the least popular content when using $dw = 255$. This makes NDN even more efficient than the host-based (unicast) approach. However, this comes at the cost of a higher RPR since waiting delays also increase when dw is higher. With $dw = 255$, an increase of *15ms* of round-trip delay per request is observed for the least popular content then whith $dw = 127$.

Overall, we find that a trade-off between cost and round-trip delays is difficult to achieve with the CF mechanism. On the one hand, trying to reduce waiting delays by reducing dw increases the number of transmissions and collisions as nodes do not have enough time to listen to each other. We should note that this situation becomes even

Table 5.3: Interest satisfaction rate

	$dw = 127$	$dw = 255$
$\alpha = 1.5$	87.8%	84.1%
$\alpha = 2.0$	95.1%	93.2%
$\alpha = 2.5$	98.2%	97.4%

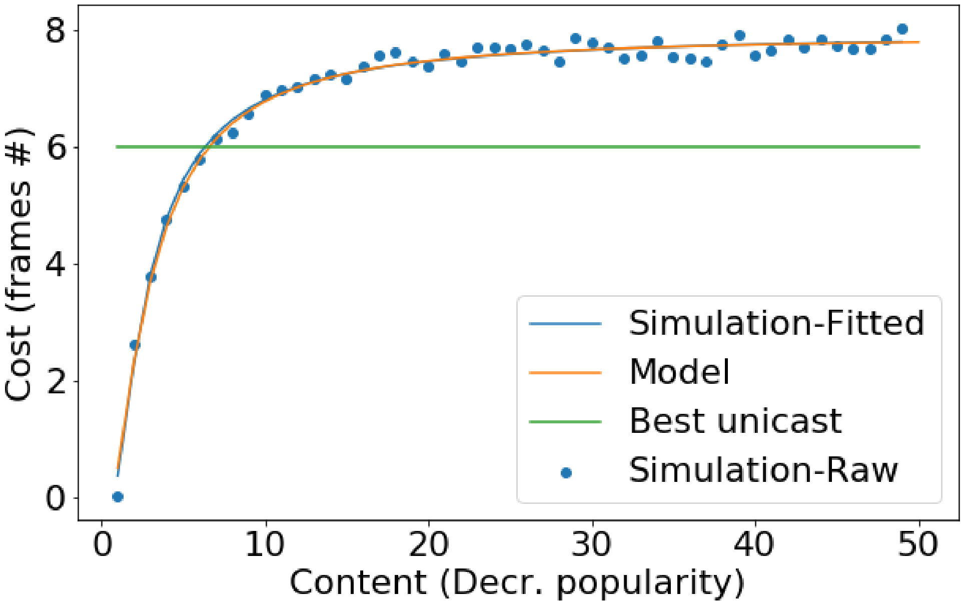


Figure 5.5: CPR: $dw = 127$, $\alpha = 1.5$

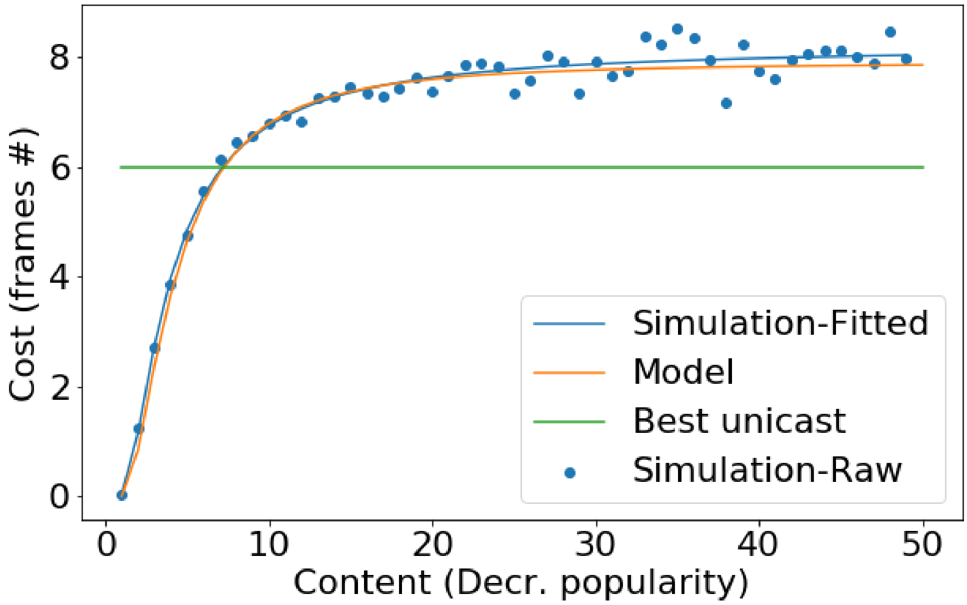


Figure 5.6: CPR: $dw = 127$, $\alpha = 2$

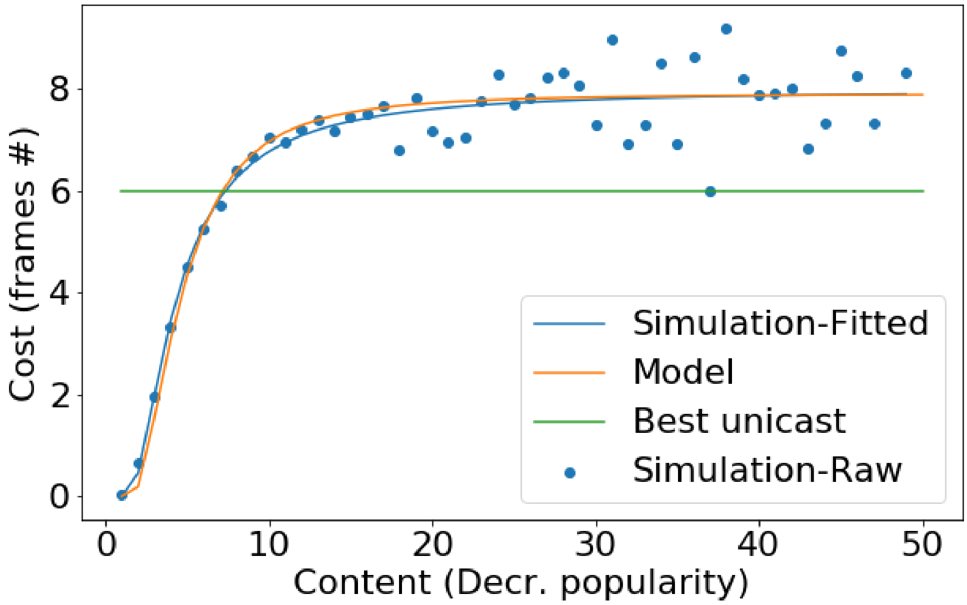


Figure 5.7: CPR: $dw = 127$, $\alpha = 2.5$

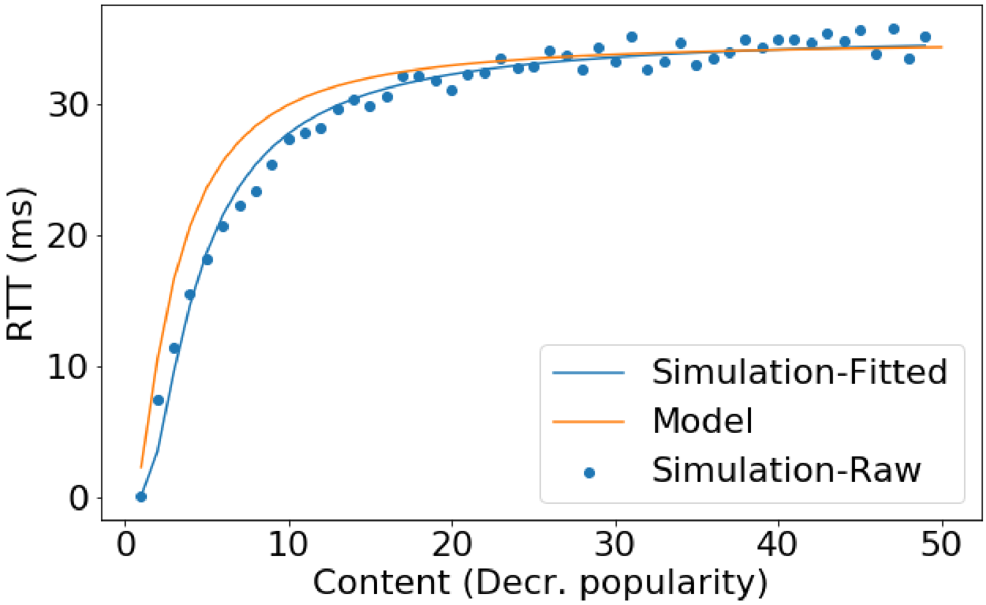


Figure 5.8: RPR: $dw = 127$, $\alpha = 1.5$

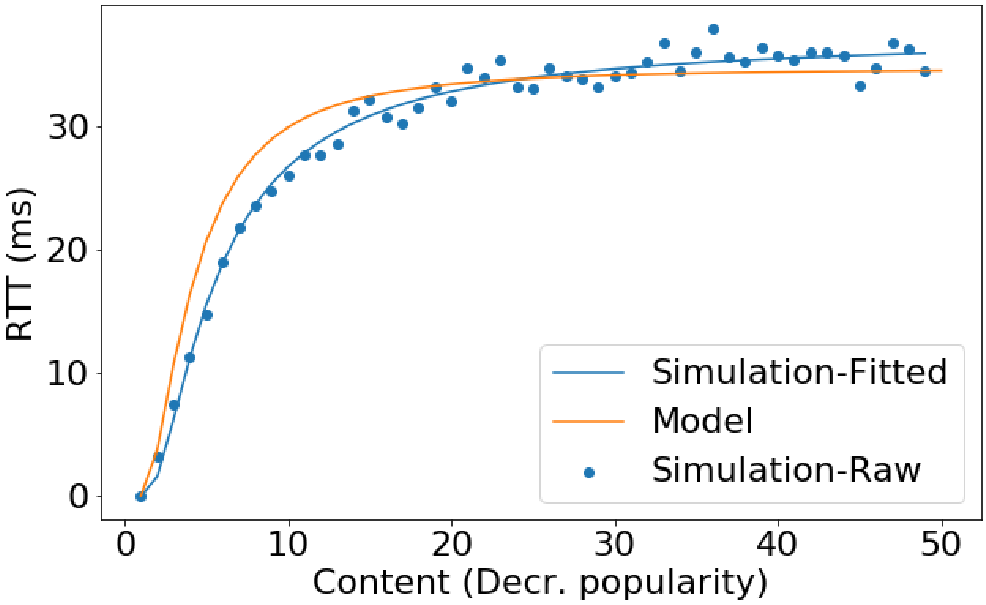


Figure 5.9: RPR: $dw = 127$, $\alpha = 2$

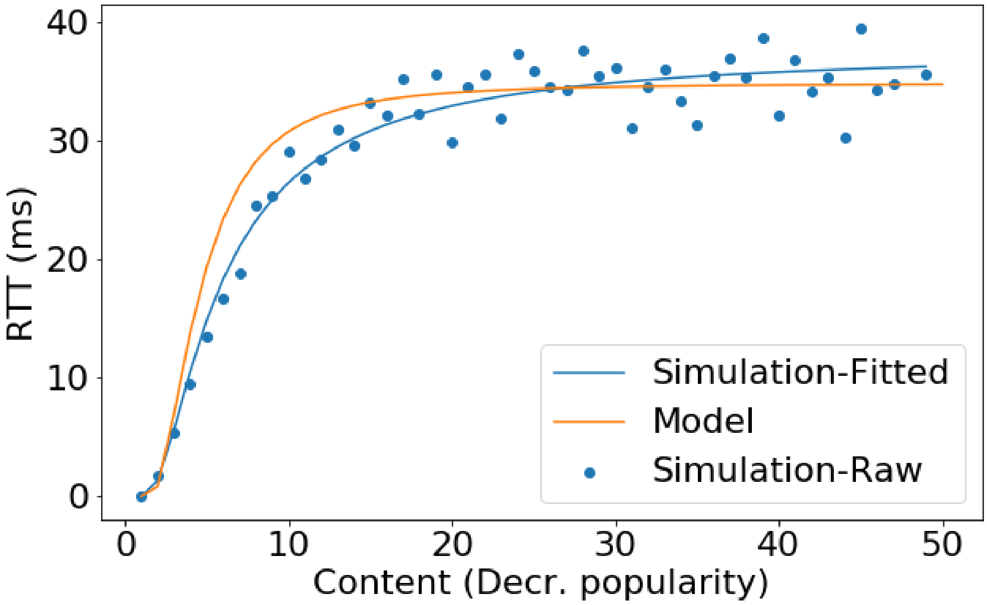


Figure 5.10: RPR: $dw = 127$, $\alpha = 2.5$

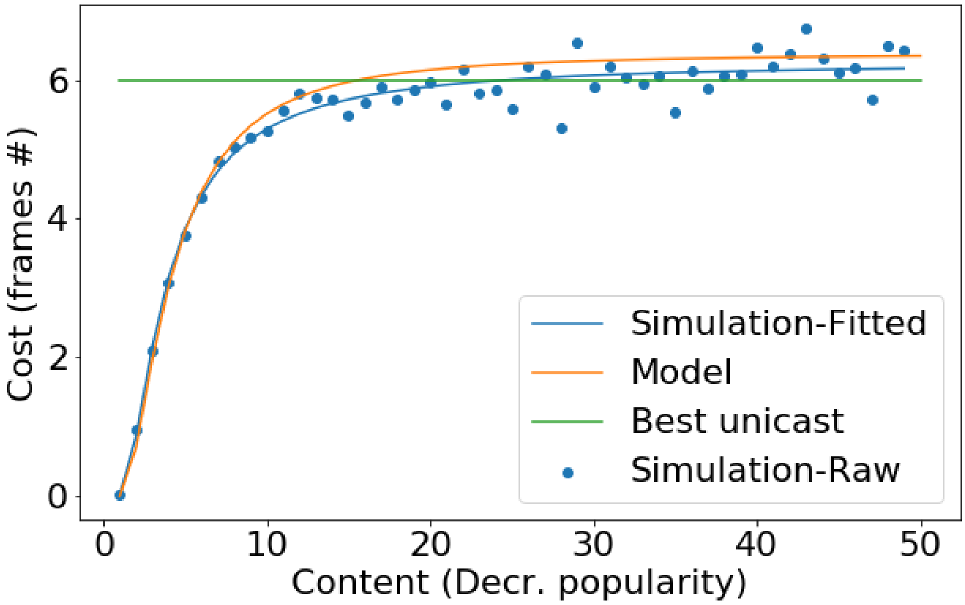


Figure 5.11: CPR: $dw = 255$, $\alpha = 2$

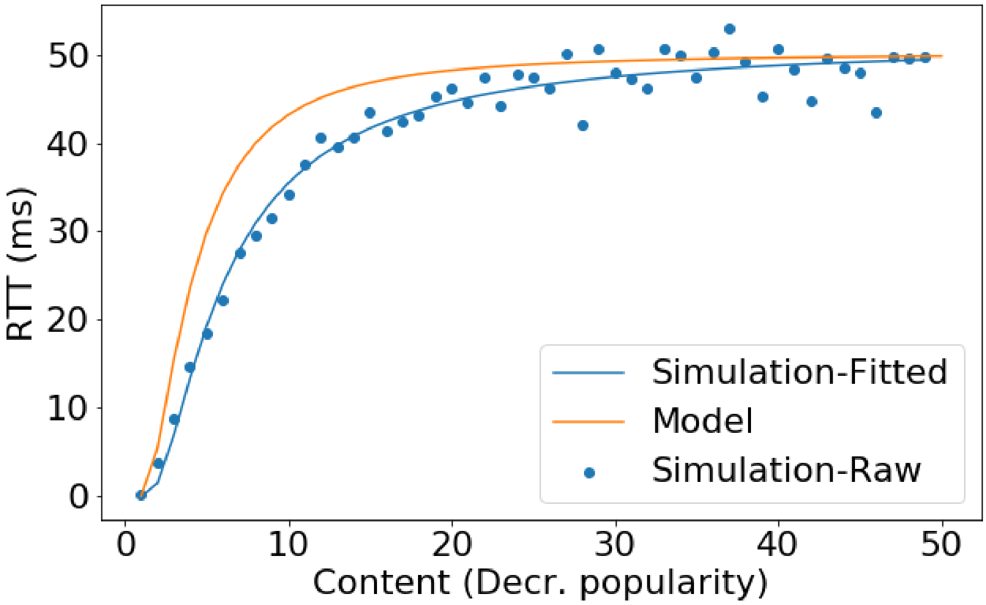


Figure 5.12: RPR: $dw = 255$, $\alpha = 2$

worse in a complex topology when more than two forwarders are available. On the other hand, reducing cost with higher values of dw will induce higher waiting delays. Moreover, nodes are still listening to transmissions when waiting, which is not helpful for energy consumption.

5.3.2 Complex Networks: Grid Topology

To reduce link-layer broadcast in constrained IoT networks, the authors in [Kietzmann et al. 2017] investigated the possible mappings between content names and MAC host addresses. The study includes the following mapping schemes:

1. Interest Broadcast, Data Broadcast (IBDB). All nodes forward Interests by broadcast when a matching prefix is found in the FIB. When a node receives a Data with a corresponding PIT entry, the Data is also forwarded by broadcast.

2. Interest Broadcast, Data Unicast (IBDU). Similarly to IBDB, all nodes forward Interests by broadcast when a matching prefix is found in the FIB. However, when

a node receives the corresponding Data packet, it forwards the Data by unicast to the sender of the Interest alone, according to the PIT entry.

3. Interest Unicast, Data Broadcast (IUDB). Every node keeps in the FIB a binding between discovered prefix-names and unicast addresses (i.e., next hop). When a matching prefix is found in the FIB, the Interest is sent only to that unicast address. However, Data packets are always forwarded by broadcast, regardless of the address to which the Interest is associated in the PIT.

4. Interest Unicast, Data Unicast (IUDU). Similarly to IUDB but Data packets are forwarded to the unicast address to which the Interest is associated in the PIT.

Real-world experiments have been conducted with a network of 30 nodes, with 1 consumer, 20 producers and 10 content items per producer. The following two setups have been considered: (i) a direct assignment of the next-hop MAC address to the corresponding face on the path to the producer, and (ii) a common prefix route where the corresponding face is mapped to a broadcast MAC address.

Overall, results show that forwarding packets with unicast significantly reduces energy consumption and processing time in comparison to broadcast forwarding. However, unicast forwarding suffers from packet loss whereas broadcast forwarding is much more efficient in retrieving content. The efficiency of a unicast mapping requires a route maintenance mechanism and thus additional overhead and processing.

These observations lead us to study the NDN-to-MAC mapping in a more complex networking scenario in terms of network overhead, communication efficiency.

For that purpose, we simulate the RONR strategy in a constrained IoT network using the four name-to-address mappings listed above. We consider a complex scenario that reflects a small-scale IoT monitoring application. The network topology is inspired by mobile WSNs. It consists of 16 fixed routers organized in a grid of 180 x 180 meters, in which 4 mobile consumers request content from 1 mobile producer that is moving following the Random Waypoint Mobility model. The MAC layer configuration reflects the IEEE 802.15.4 properties.

For all the simulations, the reported results correspond to the average values obtained with 10 random executions. The following metrics are measured:

- The total number of transmitted packets, which reflects the accuracy of each forwarding decision and its overhead.
- The mean round-trip time (RTT) for an Interest-Data exchange.
- The Interest satisfaction rate, which measures the efficiency of each approach to retrieve content
- The mean hop count of received Data, which reflects the ability of each mapping to find the nearest source (producer or cache) for the requested content.

Note that the evaluated mappings are not necessary all viable mechanisms; we test all the possible mappings to better understand the impact of broadcast and unicast for Interest and Data transmissions. Furthermore, the results found in this short study help us to choose a communication pattern and draw design guidelines for the R-LF strategy.

According to the results reported in Figure 5.13, mappings with Interest broadcast (i.e. IBDB and IBDU) always transmit the highest number of frames, as expected. Indeed, the broadcast generates more Interests in the network, which leads to more retransmissions and more collisions. For the same reason, both IUDU and IUDB generate the lowest number of frames.

IBDB and IUDB achieve, respectively, the first and the second best Interest satisfaction rate. This shows an advantage of Data broadcast, and NDN in general, to satisfy multiple pending Interests with one Data packet transmission. Moreover in IBDB, the exploratory nature of Interest broadcast takes advantage of caching in relay nodes, which gives the best satisfaction rate. This is confirmed by the mean hop count of received Data, which is the smallest in IBDB and the highest in IUDU.

However, a significant difference is observed in the Interest-Data round-trip time. Data broadcast in IUDB as well as in IBDB causes a high medium-access competition, which leads to higher round-trip time than with Data unicast (i.e. IUDU and IBDU). This raises an important concern about the necessity of using accurate timers to defer packet transmissions. That being said, since mappings with Data broadcast provide the highest satisfaction rate, we can not design a forwarding strategy for NDN without considering the Data broadcast.

To sum up, the results suggest that Interest broadcast and Data broadcast (all-broadcast) mapping should be considered, as it increases content dissemination efficiency and provides a good Interest satisfaction rate. However, the large overhead caused by the Interest broadcast requires a careful design to reduce unnecessary Interest transmissions. Moreover, the vision of NDN that consists in retrieving content without using any host address is nicely satisfied with a broadcast strategy.

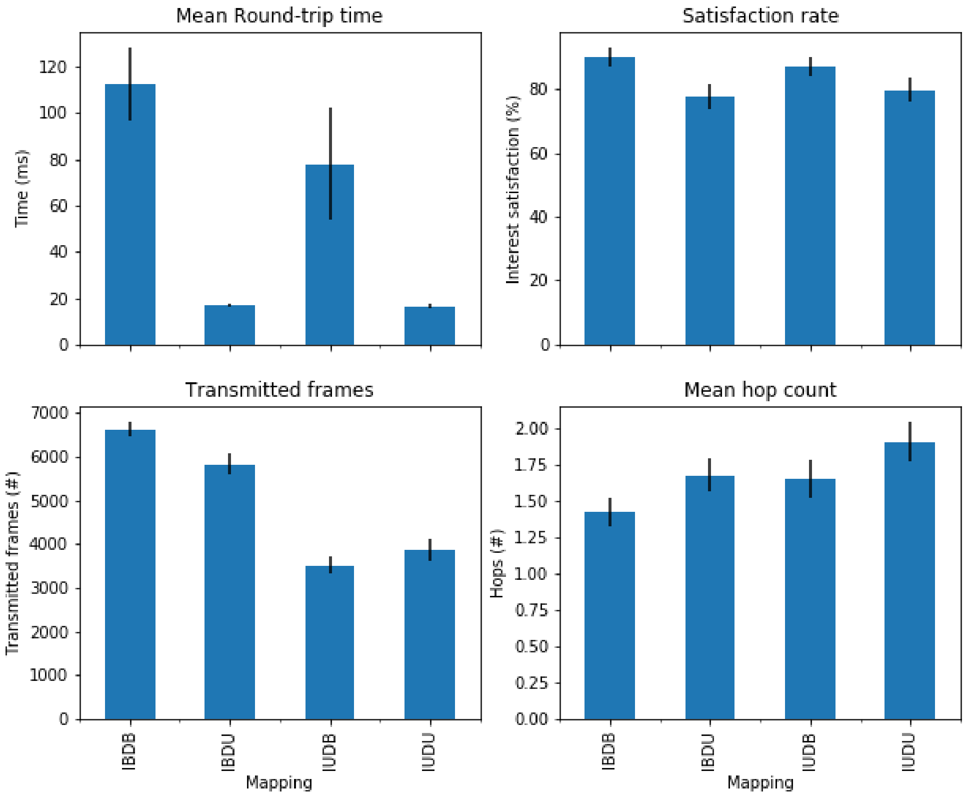


Figure 5.13: NDN-MAC mapping simulation results

5.3.3 Lightweight Wireless Forwarding: Guidelines

5.3.3.1 Summary and Understandings

Through the two previous sections, we studied the impact of broadcast on NDN wireless forwarding. The advantages of broadcast over unicast in terms of content dissemination and simplicity are clear. Particularly, the all-broadcast approach that consists in sending both Interest and Data via broadcast presents considerable features. First, broadcast is the natural communication pattern on wireless links, and point-to-point communication is artificially supported using frames filtering at the equipment level based on destination addresses. Second, when all transmissions are broadcast, nodes can overhear neighborhood transmissions, which allows them to get knowledge about the network without any explicit or additional transmissions such as control packets. Third, broadcast has a very robust support for network dynamics and topology changes due to node mobility. Moreover, it provides path redundancy and data duplication over the network, which can be exploited to enhance reliability. Fourth, our model shows that, in a simple network topology, broadcast combined with caching can outperform unicast in terms of cost in the presence of popular content. Fifth, NDN tables store less information when using broadcast as they do not have to include MAC addresses. In particular, compression techniques can be used to create a constant-sized PIT if Interest source addresses are not required.

However, using broadcast has its shortcomings and raises some challenges. First, unlike unicast, broadcast at the link-layer does not provide any acknowledgement mechanism to handle frame retransmissions, for example. Second, without a control mechanism, broadcast generates an extremely large network overhead and unnecessary packet duplication. Moreover, duplicated packets, particularly Interests, lead to important link-layer access contention and increase the risk of collisions. Third, as no packet filtering for broadcast is supported by current wireless equipment, all received packets have to be processed by the CPU, which is not reasonable with resource-constrained devices. Fourth, the overhearing possibilities offered by broadcast come at a cost of active listening to radio transmissions, which consumes a significant amount of energy.

5.3.3.2 Design Guidelines

The results obtained on various evaluation scenarios, and observations on the approaches described before can be combined to draw up the following guidelines and requirements for a lightweight forwarding strategy:

1. Rely on a minimal state to process packets without maintaining explicit routes;
2. Avoid reverting to a random Interest flooding phase in order to provide accurate forwarding decisions while reducing collision risks, network congestion and overhead;
3. Avoid node identification or addressing to correctly match the NDN vision;
4. Avoid additional data structures to preserve the lightweight aspect of the NDN stack and allow more space for caching;
5. Distribute decisions and computation tasks over the network, and minimize complexity to meet IoT device capabilities.

Taking the choice of using broadcast, and based on the guidelines designed to limit its disadvantages, we designed two approaches; the first one operates at the network layer and the second one at the link layer, as detailed in the two next sections.

5.4 L3 Solution: R-LF

5.4.1 Approach and Assumptions

Routing and forwarding operations are significantly different in NDN and IP. In IP, only the routing operation is smart in the sense that different routing protocols can be envisioned. The forwarding operation always consists in finding the longest match available in the routing table and sending the packet to the corresponding next hop. In NDN however, in addition to the routing operation that may be smart as in IP, multiple approaches are possible to handle packet forwarding with more or less additional information and with or without caching.

Our proposed strategy does not use an explicit routing phase to gather or update forwarding information. The routing phase is implicit as in the mechanisms presented in the related work. R-LF operates according to the following steps: (i) the nodes overhear Data packets and learn a cost value by reinforcement, (ii) the nodes decide to forward an Interest with a delay according to their cost-based eligibility, (iii) the nodes update their cost from the result, which can be an Interest timeout or a received Data packet.

We should note that Reinforcement Learning has been adapted before to NDN in [Chiocchetti et al. 2013], [Fu et al. 2017], [Akinwande 2018] and [Zhang et al. 2018]. However, all these solutions use the Reinforcement Learning for NDN in wired networks scenarios, without dealing with wireless broadcast and overhearing mechanisms. To the best of our knowledge, there is no NDN forwarding strategy based on Reinforcement Leaning for wireless networks, particularly constrained environments such as IEEE 802.15.4.

The following describes the general R-LF approach and the mathematical formalization.

5.4.2 General description

To forward packets, a node traditionally decides in terms of what the next hop is, which can be considered as a spatial forwarding decision. However, as NDN forwarding is based on content names and R-LF uses only broadcast directly on top on the MAC layer, a node decides in terms of when it should forward the Interest; i.e., how long it should wait before forwarding. Such a process can be seen as a temporal forwarding decision. As reported above, this vision has already been proposed, and its basic principle consists in ensuring that the most eligible node will forward the Interest first.

To describe the forwarding approach, the following assumptions are made:

- Interest and Data packets carry the cost value of the sending node, denoted as *C-field*.
- An Interest flooding with random delays is used to find the producer when the first Interest is issued for an unknown content item.
- Nodes are able to overhear Interest and Data packets related to other communications.

R-LF operates according to two phases: (i) a reinforcement learning that consists in maintaining a cost value for each available content prefix, (ii) an adjustment of the waiting delay based on the neighborhood activity.

The first learning phase starts after a source node receives a randomly flooded Interest, and acts as follows (see Figure 5.14 steps 1 and 2): (i) the source node responds with a Data packet carrying the initial cost, (ii) the first forwarder on the source-consumer path computes its cost with a reinforcement technique, replaces the C-field with the value obtained, and forwards back the Data packet, (iii) each node on the path follows the same procedure until the Data packet reaches the consumer, (iv) in the vicinity of the path, the nodes that overhear the Data packet can perform a passive cost update to learn their eligibility relative to the data source.

The random flooding phase is then over, and the first learning phase is set up. Each node updates the cost related to the corresponding content prefix after retrieving (or overhearing) a Data packet with a smaller cost. Let us refer to this phase as the *reinforcement* phase.

To describe the forwarding process, we define the delay to wait before forwarding an Interest as $\Phi(a)$. The formal definition of $\Phi(a)$ will be detailed later. The forwarding decision in a relay node consists in finding the appropriate value of a that gives a correct delay to wait. Since a is calculated in two steps, let $a = \Delta + \theta$, with Δ and θ as explained in the following.

Let $C_x(p)$ and $C_y(p)$ be the current cost for prefix p at nodes x and y respectively. Whenever node x receives an Interest issued (or forwarded) by node y, it computes the value $\Delta = C_y(p) - C_x(p)$. Here, Δ quantifies the global eligibility of node x to forward the Interest. If $\Delta \geq 0$ then node x can potentially forward the Interest.

The value of θ is locally computed by the forwarder based on its neighborhood activity to refine Δ before calculating the delay time. Let us refer to this phase as the *Delta adjustment* phase.

After computing a, the Interest forwarding is delayed for $\Phi(a)$ units of time. During the delay-listening time, if node x detects that a forwarder z is transmitting a packet with the same name, it deduces that z is more eligible to handle the Interest and cancels its

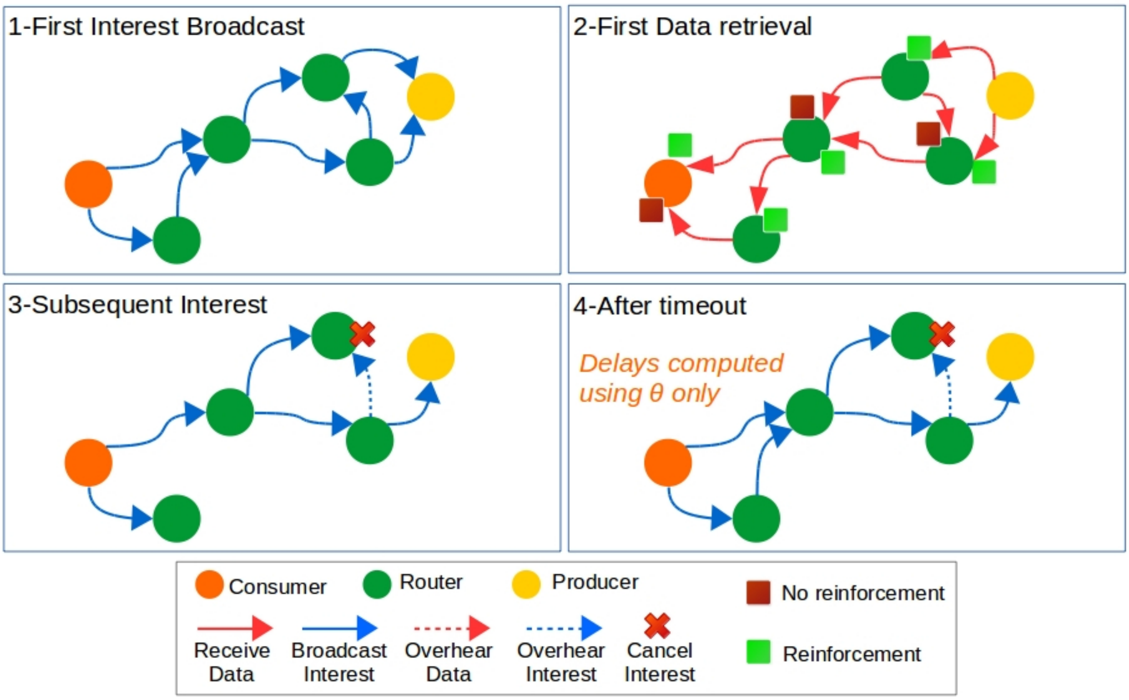


Figure 5.14: Common forwarding situations with R-LF

pending transmission (see Figure 5.14 step 3).

With the delta adjustment, the random-delayed forwarding is used only when the prefix is unknown by the forwarder (x) and is reset by the sender (y). Thus, even after an Interest timeout in x and y, the value of θ can be used in most cases to distinguish nodes eligibility (see Figure 5.14 step 4).

The next subsection provides the mathematical details.

5.4.3 Details and mathematical formalism

Reinforcement phase: The cost value at node x is updated according to the following reinforcement equation 5.9:

$$C_x(p) = (1-\alpha)C_x(p) + \alpha\,(r + \min C_y(p)) \tag{5.9}$$

In this equation used in Q-routing, α is the learning rate, r is the reward, $C_x(p)$ is the cost at node x for the prefix p, and $\min C_y(p)$ is the smallest cost heard by node x from node y.

Assuming the hop-count as a metric, the reward is always equal to *1*, and the cost of each node increases as the distance to the content source increases. The cost at the content source is *0*.

The cost values reflect the distance to the content source, and are used to decide on a forwarder's eligibility. Moreover, the approximate nature of the update formula produces a large number of possible cost values over the network, which helps to avoid obtaining the same waiting delay for different nodes.

Since the nodes remember only the smallest heard cost, a node may have an obsolete estimation of its cost value. To avoid that, after an Interest timeout, a node resets its cost value (i.e. $C_x(p)$) to *0* and the smallest heard cost (i.e. $\min C_y(p)$) to the maximum cost $\hat{\Delta}$, in order to accept cost updates. Note that in Interest packets, a cost value of *0* indicates that the sender has reset its cost or it has no information about the content prefix. Thus, it does not interfere with the *0* cost value of a Data packet which actually means that the packet has been sent by the producer. The estimation of $\hat{\Delta}$ is presented below.

When caching is enabled, the cost carried in a cached Data packet may introduce uncertainty in the reinforcement calculus, especially when a relay node has cached only a few chunks of the requested content. To overcome this, when a cached Data is returned by a relay node, the cost carried in that Data is the highest expected value (i.e., $\hat{\Delta}$). In this way, cached Data packets do not lead to a reinforcement update at other nodes, since cached Data packets may not carry accurate cost information.

Delta adjustment: The adjustment serves two purposes: it refines Δ to deal with local uncertainty in real time, and allows each node to handle multiple content prefixes simultaneously. In fact, using only Δ to compute delays, even if it is accurate, does not allow different content names cohabitation to be supported.

To compute θ, let N_a be the neighborhood activity rate for all data names. From the perspective of a node, N_a can be computed by $N_a = D_u/I_d$, where D_u is the number of unsolicited received Data and I_d is the number of non-forwarded Interests (dropped

Interests).

Then, θ may be simply defined as

$$\theta = Th - N_a \tag{5.10}$$

where $Th \leq 1$ is the activity threshold above which the waiting time should be increased. For simplicity, but without losing accuracy, N_a is kept between *0* and *1*. Thus, if Th is lower than *1* (e.g. *75%*), θ can be negative. In this case, the value of Δ is reduced, which will increase the waiting time. When no statistic is available, $N_a = Th$.

Delay function: After defining the appropriate value of a, the delay time is computed with a function that is inversely proportional to the value of a. Such a function can be intuitively defined by:

$$\Phi(a) = \frac{M}{e^{a/2}} + m \tag{5.11}$$

This function ensures that when two nodes can both forward an Interest, the node with the highest value of a will delay its transmission for a shorter time than the node with the lowest value, as depicted in Figure 5.15. In addition, m forces the forwarder to wait for a minimum time to allow the transmission of the corresponding Data packet if any, while M controls the upper-bound of the calculated delays.

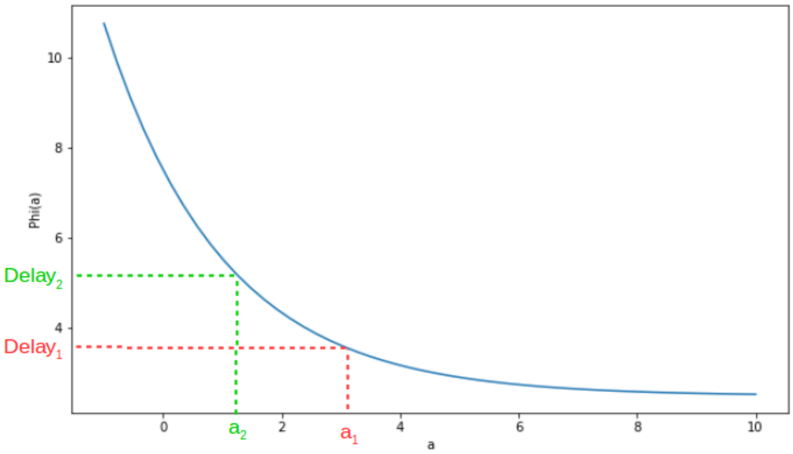


Figure 5.15: Delay function example

The importance of Φ is capital and a parameter calibration is needed to have an efficient distribution of waiting times.

We can observe that $\lim_{a \to \infty} \Phi(a) = m$. Therefore, we need to ensure that $\Phi(\hat{a}) > m$, where $\hat{a}$ is an estimation of the highest value of a.

According to Equation 5.10, θ is $\leq Th$. Given that the lowest cost value that can be computed by a node is close to *0*, we can deduce the highest gap between two cost values as being the highest cost value in the network. For that, we use the following property of the Q-learning formula presented in Equation 5.6.

Given that NDN packets do not loop by design and considering a grid topology of n nodes, we assume that the average distance between two nodes should not exceed $\sqrt{n}$ hops. Then, we use Equation 5.8 to recursively estimate the maximum expected value of Δ as $\hat{\Delta} = (\sqrt{n} + 1)$. After that, we deduce an estimation of $\hat{a}$ to set the parameters of $\Phi(a)$.

The forwarding decision process for a node x with a cost of $C_x(p)$, receiving an Interest from node y with a cost of $C_y(p)$ for a prefix p is summarized in Algorithm 1.

5.4.4 Evaluation

For evaluation purposes, we implemented the NDN module in the OMNeT++ simulator including R-LF, CF and RONR forwarding strategies. We first study the impact of the learning rate (α) on R-LF performances, and we fix the values for parameters *M*, m and *Th*. Then, we evaluate R-LF in comparison with RONR, a completely unicast approach, and CF which is a broadcast-only mechanism that uses delayed transmissions. The evaluation studies three configurations: (i) a variable number of simultaneous data flows, (ii) one data flow with multiple consumers and cache enabled in relay-nodes, (iii) multiple data flows with a variable producer mobility speed.

After that, we compare R-LF to AODV in order to study the advantages of NDN, with an all-broadcast forwarding, over IP networks, with a host-based routing protocol. The evaluation configuration has been adapted to accommodate both NDN and IP approaches, and are detailed in Section 5.4.4.6.

```
Function ProcessInterest
Data: Interest packet
begin
    if $p$ is unknown then
        if $C_y(p) == 0$ then
            Broadcast with random delay
        else
            Drop Interest (node not eligible)
        end
    else
        if $C_y(p) == 0$ then
            $\Delta = \hat{\Delta} - C_x$
        else
            $\Delta = C_y - C_x$
        end
        if $\Delta \geq 0$ then
            $\theta = Th - N_a$
            $a = \Delta + \theta$
            Broadcast with $\Phi(a)$ delay
        else
            Drop Interest (node not eligible)
        end
    end
end
```

Algorithm 1: Interest forwarding process

Finally, some experimental measures are reported on the R-LF implementation on the Arduino platform over IEEE 802.15.4.

5.4.4.1 Simulation design

We consider a scenario that reflects a small-scale IoT monitoring application. The topology consists of *16* fixed relay-nodes organized in a grid of *180m* x *180m*, through which producer and consumer nodes move following the Random Way Point mobility model.

The purpose of this simulation is to verify if the impact of the broadcast is attenuated by the forwarding strategy relatively to the network size. Moreover, the forwarding strategy is intended to be used in local networks of IoT devices. Given that R-LF strategy does not store additional topology (or node) information, the efficiency of NDN in terms of scalability achieved through caching and Interest filtering is fully preserved. Thus, a network of *16* nodes has been chosen as a typical local network size to evaluate the communication overhead of R-LF.

In practice, wireless NDN should operate without considering a logical topology based host identifications (i.e., addresses) and point-to-point links. That is, the evaluation topology does not provide any particular information to the R-LF strategy; the design of R-LF itself is clearly topology-independent. Instead, the grid topology ensures that each host is connected to *2*, *3* or *4* neighbors. Permanent host connectivity is required in R-LF to overhear packets and learn content names. Moreover, the grid topology is commonly used to evaluate (mobile) WSNs routing protocols considering mobile sensors/sink for example.

The MAC layer configuration reflects the IEEE 802.15.4 properties. Figure 5.16 gives an example of the simulated topology, and Table 5.4 reports the relevant simulation parameters.

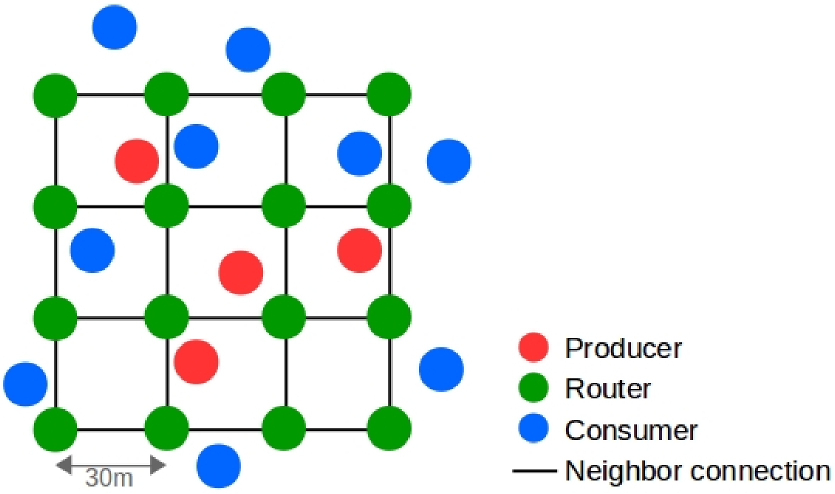


Figure 5.16: Simulated topology example

In all result figures, BF represents the broadcast-based Controlled Flooding (CF) strategy, RONR represents the unicast-based Reactive Optimistic Name-based Routing (RONR), and R-LF stands for our reinforcement-based forwarding strategy.

In all the simulations, the reported results correspond to the average values obtained with 10 random executions. The following metrics are measured:

- The total number of successfully transmitted packets, to study the accuracy of the

Table 5.4: Simulation parameters

Parameter	**Value**
Data packet size	*100 B*
Interest packet size	*50 B*
Interest send interval	*1 s*
Interest lifetime	*2 s*
Max Interest re-transmissions (at consumer node)	*1*
Cache size	*20* packets
Cache replacement	FIFO
Data freshness	*10 s*
Wireless bit-rate	*250 Kbps*
Wireless MAC protocol	*IEEE 802.15.4 CSMA*
Communication range	*35 m*

forwarding decisions and the generated overhead.

- The mean round-trip time (RTT) for an Interest-Data exchange.
- The Interest satisfaction rate to, measure the global efficiency of the approach.
- The mean hop count of received Data, to study the ability of the strategy to find the nearest source (producer or cache) for the requested content.

These metrics have been chosen for three reasons. First, they are related to the objective of our forwarding strategy, which is to take advantage of broadcast without its drawbacks. That is, the number of transmitted frames over the network indicates if the broadcast effect is attenuated, the hop-count shows the efficiency of each strategy in choosing the right forwarder, the RTT is used to check that waiting delays (when present) are not too high, and the Interest satisfaction rate measures the data delivery efficiency of each approach. Second, the four metrics mutually impact one another, and it is difficult to optimize all of them at same time. Third, evaluation of forwarding strategies in related work (as reported above) usually measures these metrics or equivalent ones.

5.4.4.2 Impact of the learning rate

R-LF involves a set of parameters that need to be calibrated to provide the best performance. Empirical simulation observations give the parameter values as follows:

M=5, *m=2.5* and *Th=0.75*.

Fixing the learning rate is required. Considering the four metrics at the same time and their respective stability over multiple simulations (i.e., confidence interval), an optimal value should be found and used further in the comparative simulations.

To study the impact of α on R-LF performance, we simulate a scenario in which two consumers request the same content served by one producer, with *20* packets caching enabled in relays nodes. The results are presented in Figure 5.17.

Small values of α (i.e. ≤ 0.6) globally give poor performances. This can be explained by the fact that small values of α prevent nodes from learning fast by slowing down the update process. Likewise, when $0.9 < \alpha \leq 1.0$, the nodes overwrite previously learned values, which leads to more similar values and thus to more frequent collisions.

We find that $0.8 \leq \alpha \leq 0.9$ generally gives the best performance results. Moreover, these values provide stable results through the different executions. Even if $\alpha = 0.9$ and $\alpha = 1.0$ achieve good performances, $\alpha = 0.8$ is more likely to keep the same performance through different executions. We conclude that *0.85* is a good value to obtain a satisfactory performance for the metrics studied. The value of α is fixed to *0.85* for the rest of the simulations.

Figure 5.18 shows the distribution of the values of *a* over time for the *16* relay nodes. Globally, only a few similarities are observed in the values and the range of the possible values is completely exploited by the nodes. This is due to the reinforcement formula and the delta adjustment that helps to reduce similarities in the computed values. More similarities in *a*-values are observed between *-0.5* and *1.5*. However, according to the Φ function (see Figure 5.15), the delays corresponding to this range are high enough to let a node with a higher value of *a* re-transmit first.

5.4.4.3 Multiple data-flows

The forwarding strategy has to support multiple namespace co-habitations. To create multiple data flows, each pair of consumer-producer exchanges packets under one name prefix. For example, consumer *C1* requests content created by producer *P1* under the

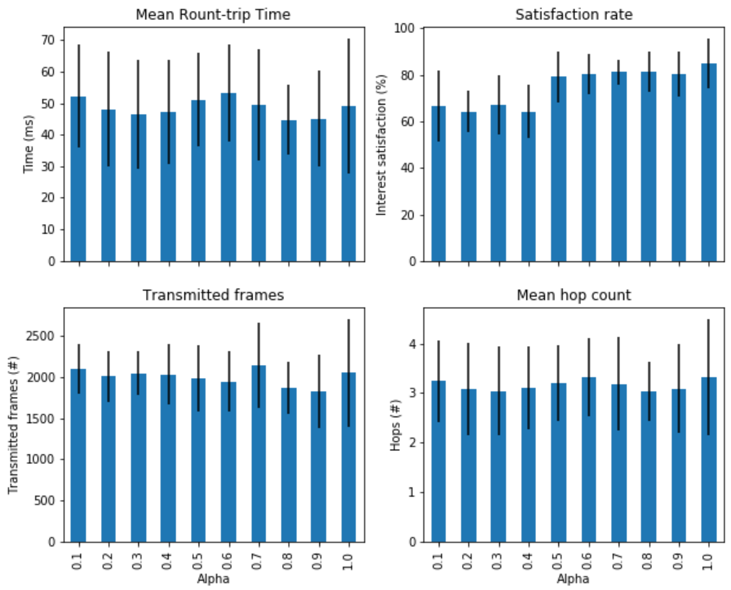


Figure 5.17: Impact of the learning rate

prefix */ndn/home/humidity/**, and consumer *C2* requests content created by producer *P2* under the prefix */ndn/home/temperature/**. In practice, this case may occur when data is collected according to its type, or when multiple applications are running in the same wireless network. Since only one consumer is requesting contents from one producer, caching is disabled in all nodes for this simulation. Figure 5.19 shows the metric measured according to 2, 4 and 8 simultaneous data-flows, which we believe is reasonable in a local monitoring application.

The total number of frames transmitted by R-LF is very low, given that it is a broadcast-only technique. We observe that the number of frames transmitted with R-LF is much closer to RONR (unicast) than to CF (broadcast). The low number of frames is the result of the Interest forwarding controlled by the reinforcement learning, followed by the delta adjustment step which can increase the waiting delay if the neighborhood activity is high. Moreover, the same difference between the three approaches is observed for 2, 4 and 8 data flows.

The satisfaction rate is roughly the same for the three approaches. However, the broad-

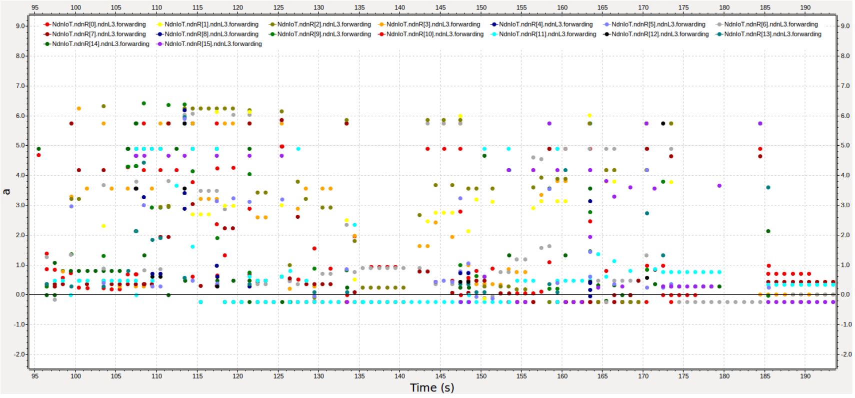


Figure 5.18: *a*-values example

cast strategies (i.e. R-LF and CF) seem to be more suitable to support more simultaneous data-flows.

On average, R-LF and CF retrieve content with almost one hop fewer than RONR. Indeed, the unicast forwarding approach needs the expiration of the FIB entry to update the next hop address, while the broadcast forwarding can retrieve a content item from any available source in the vicinity of the forwarder. However, we can observe that this performance is achieved by CF at the cost of a large overhead, while R-LF gives the same efficiency with an overhead close to RONR.

R-LF provides an RTT lower than CF as it uses proportional timers instead of random ones. Moreover, R-LF scales better with more nodes/data-flows and becomes close to RONR in terms of RTT. Obviously, RONR achieves the lowest RTT as forwarders do not use any delays before forwarding packets. Having a reduced RTT is the main advantage of using a unicast strategy in this configuration.

5.4.4.4 Multiple consumers

In this case, only one content prefix (e.g. */ndn/home/**) is involved in the network. Multiple consumers simultaneously request content provided by one producer, and/or by intermediate caches. This case is much closer to a many-to-one communication, such as a gateway-to-devices communication. However, it can also be considered as many-to-many due to caching that allows any relay node to be a partial source of a content.

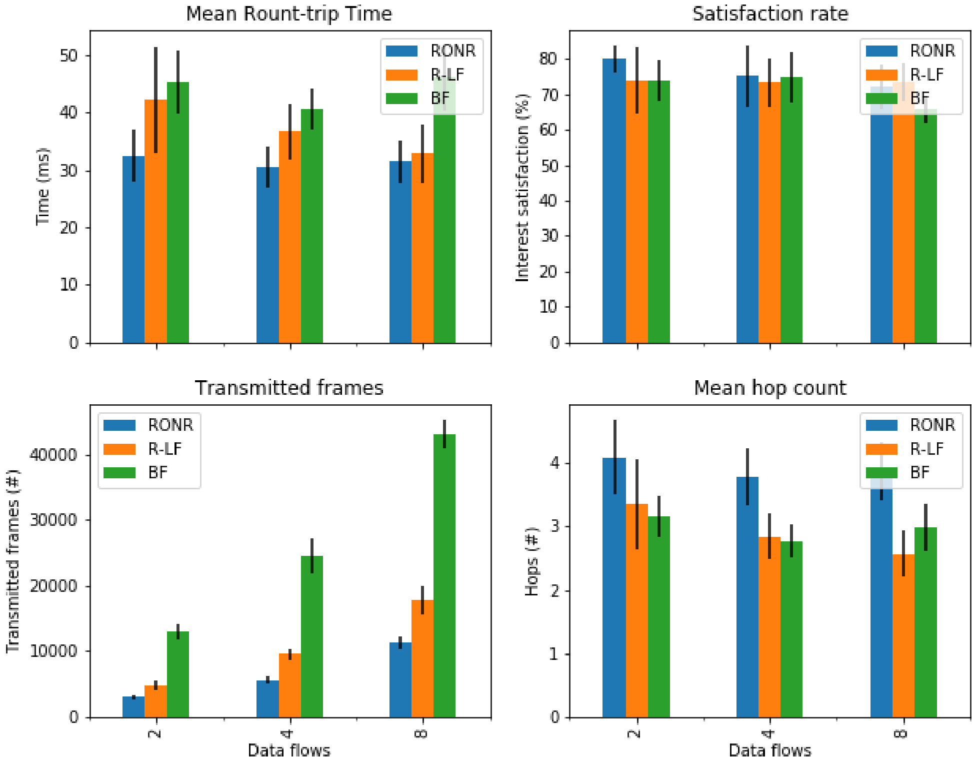


Figure 5.19: Multiple data-flows scenario results

Figure 5.20 shows the results obtained for 2, 4 and 8 consumers.

Broadcast approaches (i.e. CF and R-LF) make better use of caching and achieve the highest satisfaction rate. The difference in the hop-count for received Data packets is still to the advantage of R-LF and CF. However, R-LF keeps its reduced overhead in comparison to CF.

When enabling caches, CF and R-LF are clearly penalized by the RTT in comparison to RONR. This is certainly due to the high competition access to the medium as the number of content sources increases with caching. Here, it seems that accurate Interest broadcast and delayed forwarding of R-LF are not sufficient to reduce medium access contention and provide a low RTT.

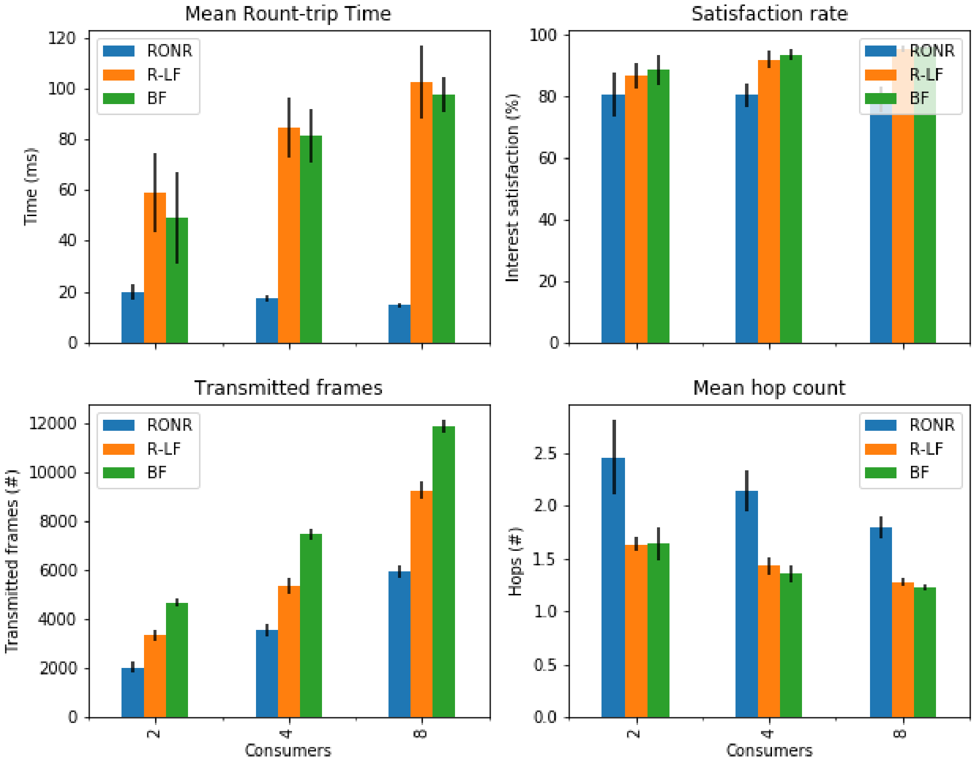


Figure 5.20: Multiple consumers and caching scenario results

5.4.4.5 Producer speed

This case is the same as the first simulation scenario, with *4* dataflows. To study the forwarding strategies support for higher mobility, we increase the producer speed from *1 mps* to *7 mps*.

Figure 5.21 shows the results obtained. As for the first scenario, the R-LF performance is close to the best among RONR and CF for each metric. We observe that when producers move faster, the RTT achieved by R-LF becomes close to RONR, whereas it increases in the CF strategy. However, the satisfaction rate with RONR decreases significantly when the speed increases, whereas it decreases reasonably with R-LF.

Even under producer mobility, CF achieves a good satisfaction rate as broadcast is not much affected by mobility. However, the high satisfaction rate achieved by CF comes with a very large overhead, as shown by the total number of transmitted frames. Moreover, we observe that the number of frames transmitted in R-LF is slightly higher than RONR, but very low in comparison to CF. Finally, the shortest path is still ensured by R-LF, even with a higher speed. The unicast mapping shows its limitations as regards keeping a good performance in the presence of mobile nodes, while R-LF is able to provide a satisfactory performance.

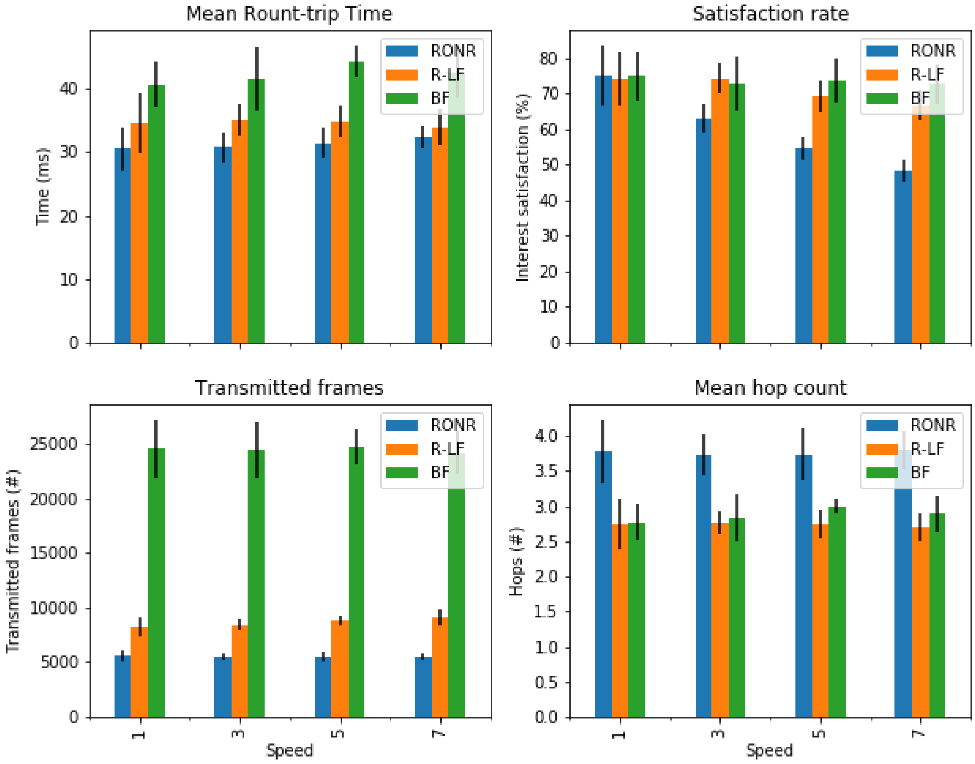


Figure 5.21: Producer speed scenario results

5.4.4.6 Comparison to host-based networking

We compare R-LF to AODV according to two scenarios. In the first scenario, multi-consumer, we consider one content producer and a set of consumers; thus, one data-flow is present in the network. This means that one destination is requested in AODV and one prefix-name is requested in R-LF. In the second scenario, multiple data-flows, different numbers of consumer-producer pairs are tested. For that, each consumer-producer pair uses a separate prefix-name in R-LF, and each consumer requests content from one host (i.e., producer) in AODV.

In both scenarios all nodes are static to accommodate AODV, which does not perform well under high network dynamics. In order to get a fair comparison, link-layer frame acknowledgement is disabled for AODV (i.e., unicast communications). Moreover, no packet/frame retransmission is enabled; the content request simply operates as the ping process for AODV. The measured metrics are the following: (i) the number of successfully transmitted frames, (ii) the mean RTT, which corresponds to the delay for an Interest-Data exchange for R-LF and a request-response in AODV, (iii) the Interest and request satisfaction rate for R-LF and AODV respectively, (iv) the total back-off time spent by a node on average to access the wireless medium.

Figure 5.23 and 5.22 show the results obtained in the multi-consumer and the multiple data-flows scenarios respectively.

AODV and R-LF are based on different paradigms and use completely different routing approaches. The objective of this comparison is to show how a full-broadcast forwarding technique can approach, and sometimes outperform, a host-based routing protocol that relies on explicit routes and host addresses. The results obtained show that this is completely feasible, although a broadcast-based forwarding without explicit routes can not outperform the unicast approach in all aspects.

With multiple consumers and one content source, request satisfaction is kept stable with AODV while it slightly decreases with R-LF. Under multiple consumers and producers (i.e., multiple data-flows), R-LF can approach and outperform AODV in terms request/Interest satisfaction. This is consistent with the multicast-friendly feature of ICN/NDN.

In terms of round-trip delay, R-LF approaches AODV when multiple consumers are involved. The same behaviour can be observed in the multiple data-flows scenario when the number of consumers increases. This can be considered as a significant achievement since R-LF uses delayed transmissions, which proves that delays are relatively accurate and not superfluous.

In both scenarios, when the number of consumers is low, R-LF is almost as accurate as AODV as shown through the number of transmitted frames and mean backoff-time. Obviously, the number of frame transmissions in R-LF becomes much higher than AODV when more nodes are communicating (i.e., more consumers). This is expected as R-LF is based on broadcast only while AODV mostly uses unicast, especially here when only one destination is requested (e.g., multi-consumers). For the same reasons, the mean backoff-time is also higher for R-LF due to the high medium access contention. However, the difference is not too great if we consider the obvious simplicity of R-LF compared to AODV.

Overall, AODV performs better than R-LF when the number of consumers increases under the same data-flow, while R-LF achieves a better satisfaction rate when multiple data-flows are present.

In similar scenarios with mobile nodes, the performances achieved by AODV were very poor in comparison to R-LF. For this reason, we chose not to report them here.

5.4.4.7 Implementation on IoT devices

The R-LF strategy has been implemented and tested in our NDN-802.15.4 testbed described in the last chapter. As equipment and libraries have not changed, we evaluate the impact of R-LF by measuring the time required to forward Interest and Data for both Arduino UNO (*16 Mhz*) and DUE (*84 Mhz*) with R-LF.

The results are reported in Table 5.5. The R-LF forwarding process delay for the first Interest (i.e., unknown prefix name) is approximately 145μs on Arduino UNO and 55μs on Arduino DUE. Subsequent Interests forwarding (i.e., known prefix name) takes about 50μs on Arduino UNO with *5* entries in the FIB, and 10μs on Arduino DUE considering *10* FIB entries. This delay difference between first and subsequent forwarding is mainly due to the random number generation which is used in R-LF for only the first Interest forwarding. With R-LF, a FIB entry update after receiving a Data packet consists on a reinforcement learning computation and takes 70μs on Arduino UNO and 18μs on Arduino DUE. The

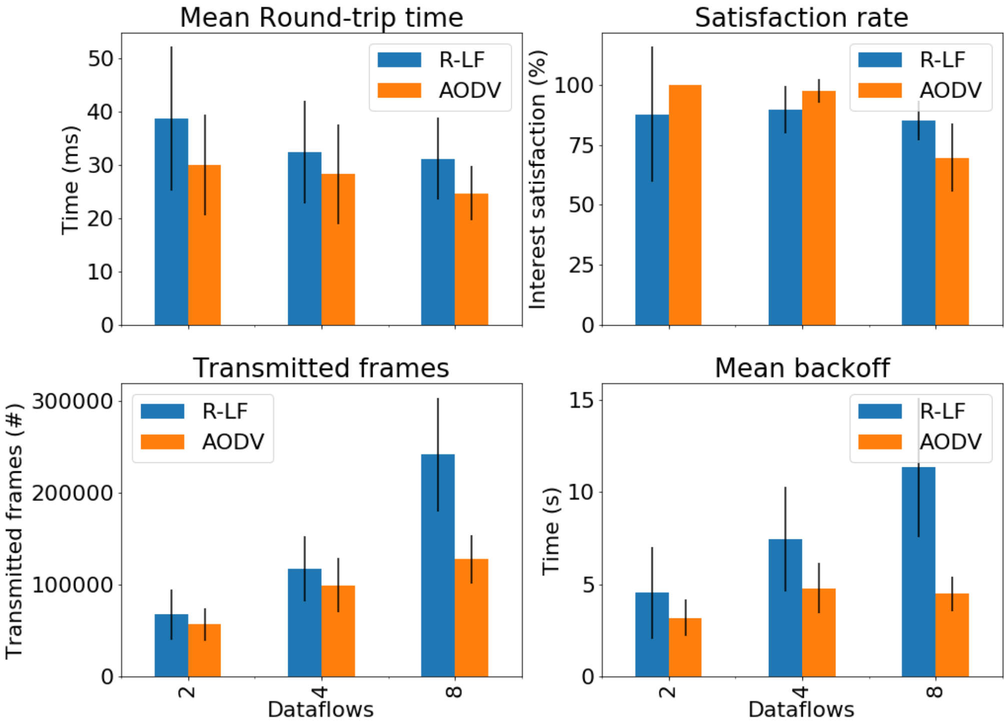


Figure 5.22: R-LF and AODV comparison with multiple data-flows

measured values show the simplicity of forwarding decisions in the R-LF strategy.

As concerns the memory space required by the implementation, the measurements reported in Chapter 4 already include the R-LF implementation, as mentioned.

Table 5.5: R-LF measures on Arduino

Operation	**Time (UNO)**	**Time (DUE)**
First Interest forwarding	$145\mu s$	$55\mu s$
Subsequent Interest forwarding	$50\mu s$	$10\mu s$
Data forwarding	$50\mu s$	$10\mu s$
Reinforcement	$70\mu s$	$18\mu s$

5.4.5 Discussion

In the three simulated scenarios, the R-LF performance is always close to the best one among RONR and CF for all the metrics. This shows the adaptability of R-LF to handle

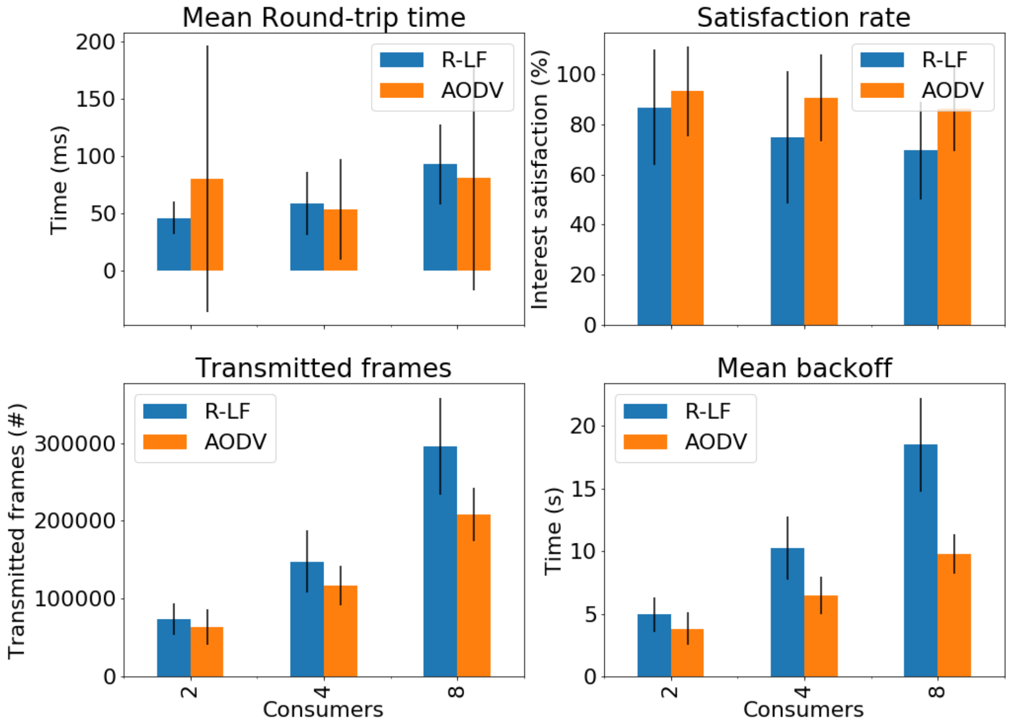


Figure 5.23: R-LF and AODV comparison with multiple consumers

different communication scenarios. The only exception to this observation is the RTT in the multi-consumer scenario, which certainly requires more sophisticated techniques for forwarding cached Data packets.

In our strategy, random Interest flooding rarely occurs, as the delta adjustment step can be used to compute a waiting delay even if the cost value is reset. This contributes to significantly reduce the network overhead and represents an important improvement regarding the basic broadcast-and-learn schema.

The reinforcement learning provides a satisfactory performance without requiring an explicit exploration phase. Hence, to forward an Interest, a node only needs a simple FIB look-up as expected in NDN.

In wireless forwarding strategies so far, the efficiency of broadcast comes at the cost of high overhead while the performance of unicast holds only in simple communication scenarios, such as with static nodes. Indeed, more complex scenarios show that RONR is beaten by R-LF in terms of Interest satisfaction-rate, hop count, and their respective latency become the same. For example, when producer speed increases, the RONR satisfaction rate drastically drops whereas R-LF is still stable. Hence, considering different communication scenarios, the results obtained show that R-LF is a satisfactory trade-off solution in many configurations.

5.5 L2 solution: ND-CSMA

5.5.1 Approach

Considering the CF strategy studied in Section 5.3.1, an ideal improvement one may look for is to reduce the round-trip time and energy consumption by eliminating waiting delays while keeping the lowest number of frame transmissions (i.e., CPR). Since eliminating waiting delays will significantly impact the CPR, we have to question whether a trade-off theoretically exists that may achieve reasonable CPR, low RTT and a reasonable Interest satisfaction rate.

According to the CF strategy model, the tree is explored depending on whether both sibling-nodes forward the Interest or only one of them does. In the model evaluation, we find that the best performance for CF is achieved with $dw = 127$. Let the corresponding Interest transmission probability for each sibling-node pair be p_t^*. Then, we can easily observe that there is no value of p_t lower than p_t^* that can achieve the same or better satisfaction rate. That is, a compromise at L3 level that achieves optimal performance is not possible in our configuration.

We believe that this is due to the fact that in the forwarding decision of a node, only the sibling-node is involved. Therefore, by shifting the transmission decision to the L2 level (with some modifications) instead of using deferred transmissions, one may expect better performances since the CSMA algorithm natively considers multiple-access contention.

As mentioned before, eliminating waiting delays will inevitably increase the number of unnecessary packet transmissions and channel access contention. To handle that, we consider $p_t = 1$, which corresponds to $dw = 0$, and we modify the CSMA algorithm of the IEEE 802.15.4 in such a way that the number of attempts to access the channel is lower when transmitting an Interest than when transmitting a Data. In practical terms, we replace the waiting delays by a priority-based CSMA scheme designed for NDN. In this section, we describe the legacy CSMA algorithm followed by the design and evaluation of the ND-CSMA scheme.

5.5.2 Legacy CSMA

IEEE 802.15.4 [IEEE 2011] defines a standard for the physical and MAC layers of low-rate wireless networks. The standard uses slotted or unslotted CSMA as a medium access mechanism. In this section, we consider the unslotted version of CSMA.

The CSMA algorithm works with a set of default parameters and each node maintains two values when running the algorithm: Number of Back-offs (NB) and Back-off Exponent (BE). NB is always initialized to 0 for a new packet transmission, and it denotes the number of access attempts for the current packet transmission. BE is used to compute the random back-off period that a device should wait before attempting to assess the channel. Default parameter values are shown in Table 5.6.

The CSMA algorithm operate as follows:

1. **Step 1.** The values of NB and BE are initialized according to the IEEE 802.15.4 standard.

2. **Step 2.** The delay of random back-off period is selected in the range from 0 to 2^{BE-1}.

3. **Step 3.** After the waiting time, the node performs a Clear Channel Assessment (CCA). If the channel is idle, the node starts transmission. If the channel is assessed to be busy, the algorithm increments NB by 1 and updates BE as follows: $BE = min(BE + 1, aMaxBE)$. Then, if NB is lower than the maximum number of back-offs (i.e., *macMaxCSMABackoffs*) the algorithm goes to Step 2; if not, the transmission is canceled and considered to have failed.

Table 5.6: Default values for IEEE 802.15.4 CSMA

Parameters	**Value**
AMaxBE	5
MacMaxCSMABackoffs	4
MacMinBE	3

5.5.3 The Named-Data CSMA Scheme

In legacy CSMA, all nodes access the shared channel with a fair chance. However, priority-based CSMA [ZHAO et al. 2013] uses the difference in traffic type to introduce differentiated channel access for nodes. Therefore, the priority-based CSMA mechanism is designed to make nodes with high priority traffic have a greater chance of accessing the channel. The ND-CSMA algorithm we propose is inspired by the priority-based CSMA approach.

The frames are classified into two priority classes according to the packet type they transmit: (i) frames that contain a Data packet at any node, and frames that contain a locally issued Interest (i.e., consumer node) are assigned a priority *0*. (ii) frames that contain an Interest packet to forward (i.e., at relay nodes) are assigned a priority *1*. The other CSMA parameters and values are kept the same in ND-CSMA.

By distinguishing between Interest and Data frames, the ND-CSMA algorithm operates in the same way as legacy CSMA described above, with one difference. When the channel is assessed to be busy, retrying another back-off depends on the priority class of the frame to transmit. If the frame has a priority *1*, the number of back-off attempts is limited by a threshold value th. Then, the transmission is canceled if the number of attempts reaches th. The algorithm operates as usual for the frames of priority *0*. According to the number of back-offs allowed by the CSMA parameters, the values of th should be between 1 and 4, while $th = 5$ makes ND-CSMA equivalent to the legacy CSMA.

ND-CSMA scheme is summarized in Figure 5.24.

5.5.4 Evaluation

To evaluate the ND-CSMA scheme, we simulate three scenarios.

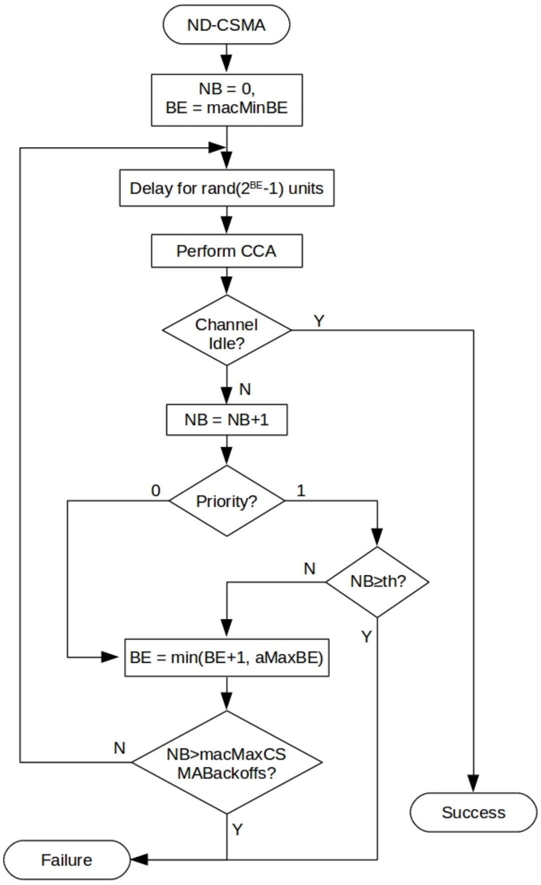


Figure 5.24: ND-CSMA algorithm

- **CF.** The CF strategy as described in Section 4.4.1 with the legacy CSMA algorithm.
- **BF.** The BF strategy presented in Section 5.2.2, also using the legacy CSMA. Notice that this scenario is equivalent to the CF strategy without waiting delays.
- **ND-CSMA-x.** The BF strategy using the ND-CSMA mechanism with $th = x$.

In the BF strategy, upon receiving an Interest or Data, a node immediately tries to forward the packet and lets the link-layer medium access algorithm (i.e., legacy CSMA or ND-CSMA) resolve the contention.

All the scenarios are simulated under the same conditions and parameters as those used in Section 5.3.1: a tree topology with depth $N = 4$, $M = 3000$ content items distributed in $K = 50$ classes, each one with $m = 60$ items, and a cache of size $x = 300$ packets at each level. We set $\alpha = 2.0$. For the CF strategy, we use $dw = 127$ and $\tau = 0.032\mu s$. We measure the following metrics:

- **RPR.** The mean time needed for the gateway to retrieve a content item from a device.
- **Transmitted frames**. The total number of frames transmitted during the simulation. We also refer to this metric as the cost.
- **Interest satisfaction rate.** This corresponds to the number of Data received by the gateway over the number of Interests it sent.
- **Mean back-off time.** The average time the nodes spent in back-off to access the wireless medium. This is also used as an indicator for energy consumption.

Figure 5.25 shows the results obtained. The objective of the ND-CSMA mechanism is to provide a trade-off between cost and round-trip time while keeping an acceptable Interest satisfaction rate.

The results show that ND-CSMA with $th = 1$ achieves the lowest RPR compared to CF and even BF with legacy CSMA. The mean back-off time with ND-CSMA-1 is the smallest among the evaluated scenarios, while BF achieves the highest back-off time due to the large number of forwarding decisions generated after eliminating waiting delays. That means, legacy CSMA with BF has to resolve medium access contention with more back-off periods whereas ND-CSMA has the possibility to cancel some Interest transmissions when the channel is busy rather than waiting for other back-off periods. For this reason, the RPR achieved by BF with legacy CSMA is slightly higher than ND-CSMA. CF also is capable of canceling scheduled transmissions using deferred transmissions. However, it achieves that with a wait-and-listen mechanism which induces higher round-trip delays, and a relatively high back-off time is required when the chosen random delays are not different enough.

Moreover, the results show that forwarding with ND-CSMA-1 can ensure necessary packet transmissions while keeping the total cost at a minimum compared to the two other schemes. The Interest satisfaction rate is quite similar for all the approaches, which indicates that ND-CSMA, even with low th does not reduce the efficiency of Interest flooding/broadcast. Furthermore, simulations with small caches (e.g., tens of packets) achieved approximately the same results, but with a slightly lower satisfaction rate for ND-CSMA.

Overall, ND-CSMA-1 seems to be the best compromise for the measured metrics. To return to our theoretical expectations, the results confirm that a link-layer adaptation is able to keep the benefits of a broadcast-based forwarding strategy in terms of satisfaction rate, while reducing medium access contention and the number of transmissions; it then achieves the trade-off we were looking for.

5.5.5 Discussion

To return to our theoretical expectations, the results confirm that an L2 mechanism is able to keep the benefits of a forwarding strategy with $p_t = 1$ in terms of satisfaction rate while reducing medium access contention and the number of transmissions; it then achieves the trade-off we were looking for.

Based on a simple modification of the IEEE 802.15.4, preliminary results shed light on the necessity of rethinking typical link-layer schemes for ICN/NDN such as the CSMA algorithm. Moreover, the results show that forwarding with ND-CSMA can ensure neces-

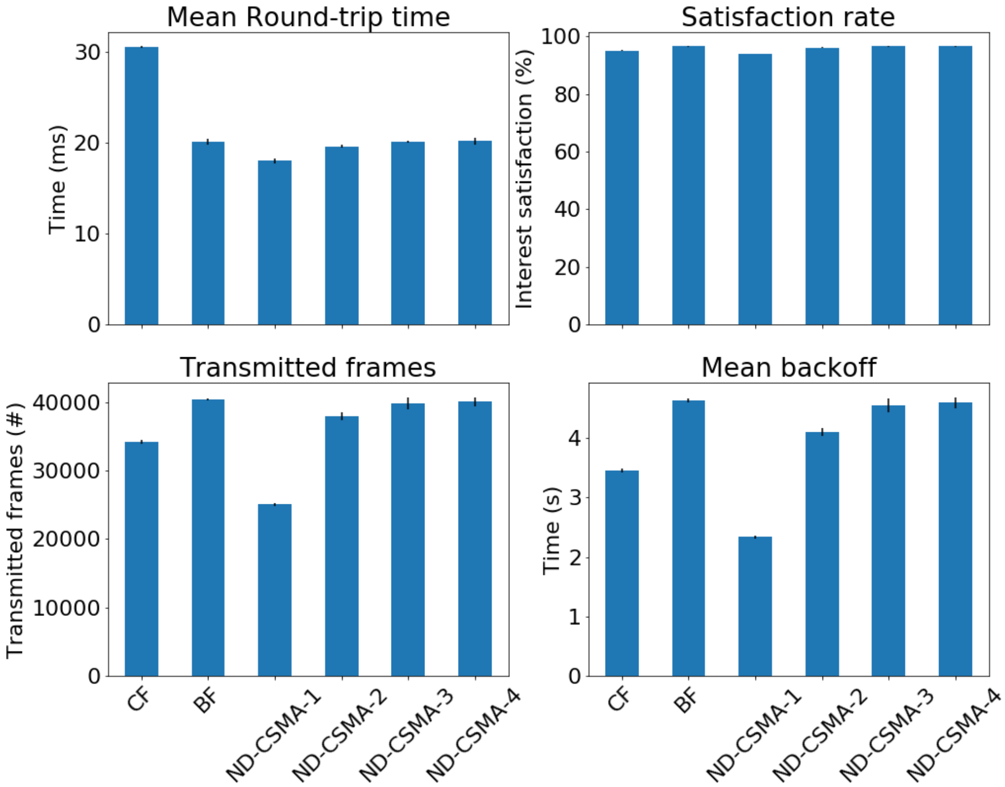


Figure 5.25: ND-CSMA evaluation

sary packet forwarding while keeping the total cost at a minimum compared to the two other broadcast-based schemes.

However, we can easily foresee that the ND-CSMA adaptation is specific to the topology and scenario under consideration, and thus can not be applied as a general solution to all network configurations.

5.6 Summary and Discussion

To support the NDN wireless forwarding over IEEE 802.15.4, we proposed two broadcast-based solutions operating at two different levels.

The first solution, R-LF, operates at network level and exploits broadcast communications with reinforcement learning to take accurate forwarding decisions. According to the results, the R-LF performance is always close to the best among RONR and CF for all the metrics. This shows the adaptability of R-LF to different communication scenarios including complex ones. The results obtained through different scenarios show that the accuracy of the unicast mapping can be approached by a broadcast-based strategy. Moreover, they show the adaptability of the R-LF approach to support multiple data-flows, multiple consumers and node mobility.

In the second solution, we explored the link-layer level to design an adaptation of the IEEE 802.15.4 CSMA algorithm. The purpose of this adaptation is to reduce the overhead effect of broadcast while keeping its advantages. To the best of our knowledge, no previous proposals have been made to adapt the CSMA scheme of the IEEE 802.15.4 link-layer to improve NDN wireless forwarding.

5.7 Conclusion

The NDN forwarding in wireless mesh networks brings a different vision of communication in wireless environments. To be in keeping with the NDN vision, wireless forwarding must focus on content names without any host identification such as MAC addresses. Moreover, the communication pattern in the IoT suggests that communications can involve more than two identified hosts as the same content can be shared between multiple nodes. Therefore, disseminating content in IoT networks may be achieved more efficiently with broadcast communications, which are more natural and more compliant with wireless diffusion technologies. For example, issuing a command towards all the cow sensors in our cow health monitoring system may be achieved more naturally using reliable broadcast communications. However, most existing (host-based) solutions would do this using multicast with the point-to-point communication pattern. This is due to the fact that IP is not compliant with the broadcast pattern, whereas the NDN communication scheme is completely broadcast-friendly.

In this context, our first objective through the two approaches proposed in this chapter is to show that broadcast can be used successfully in constrained networks, while ensuring reduced overhead and accurate forwarding decisions. For that, we designed a forwarding strategy based only on content names, and broadcast without any host identification. Moreover, to further reduce overhead, we designed a reactive strategy that does not require additional communication to maintain forwarding information. Results obtained by our forwarding approach show that a compromise between communication efficiency and reduced overhead can be achieved using broadcast.

The second objective is to explore how current communication technologies can be affected by the NDN paradigm. For that, we designed an adaptation of the CSMA algorithm used in IEEE 802.15.4 to support Interest and Data distinction to handle medium access contentions. Although this adaptation is simple, the results obtained attest that link-layer adaptations of current technologies can improve forwarding efficiency.

General Conclusion and Perspectives

At the end of this manuscript, we shall summarize the work carried out, followed by a discussion on its main limitations, and a typical scenario that may fully exploit the achievements presented in this thesis. Finally, we draw up some perspectives and ongoing work for new lines of research.

Summary

The work done in this thesis started with one main purpose which was enabling NDN in low-end IoT environments. To achieve that, we needed a combination of complementary tools that allowed us to achieve partial objectives, and which have now become a part of the global work. Broadly speaking, we investigated how to take advantage of NDN for the IoT as soon as possible. To that purpose, a realistic NDN-802.15.4 architecture has been designed and built considering the IEEE 802.15.4 wireless technology. After identifying the integration of NDN in the low-end IoT as the most realistic approach, the main integration issues have been discussed, and some propositions have been made by giving another look at some IP-based solutions for the IoT, such as 6LoWPAN. The proposed mechanisms show the flexibility of NDN to support low-rate wireless technologies such as the IEEE 802.15.4. The NDN-802.15.4 architecture obtained aims to shape a novel and strong NDN-IoT duo. More importantly, lightweight NDN forwarding in wireless networks with broadcast has been investigated. The results obtained show that we can use broadcast communications to achieve relatively accurate forwarding decisions with reduced overhead and satisfactory performance. In short, we were able to preserve the advantages of broadcast while reducing its drawbacks. While investigating wireless forwarding with NDN, an environment has been developed including different but complementary evaluation tools. These tools can provide real-world measurements, simulation results and mathematical analysis of NDN forwarding in wireless networks. Furthermore, this evaluation environment can be exploited again for other related purposes.

Overall, the main limitation we may identify in this work is the lack of direct performance comparisons between NDN and IP. Although it could be useful, this can be explained by several reasons. First, we greatly rely on the discussions about IP limitations and NDN native features to show the superiority of NDN, which is indisputable in many aspects such as security, native caching and simplicity. Second, direct comparisons between NDN and IP are inconclusive due to the paradigm difference as discussed in Chapter 2. Third, NDN implementations, including the one presented in this document, are still in an experimental stage while IP-based solutions are mature and commonly industry-oriented.

Finally, the last thought to conclude about the contributions presented in this manuscript is the following. They do not represent and end in themselves or a finalization of a work, but rather, a starting point that paves the way for some really exciting steps. One example is the design of an IoT product, or platform, based on NDN, as described in the next section.

Towards an NDN Product for the IoT

In this section, we provide a simple scenario in which the work presented in this manuscript can be used to design a PoC for an IoT product with NDN. As mentioned in the introduction, developing a PoC may be the key to creating and commercializing an IoT product.

First, an original idea on the service or the application to design must be found, and an accurate use case should be defined. This can be, for example, a connected greenhouse. A connected greenhouse is a farming facility that uses sensors to capture data on plant growth, irrigation, pest control usage and lighting, and send it to a local or Cloud-based server. Using collected data, a web application allows farmers to configure the system's settings and take decisions, while a mobile application generates alerts and reports on the greenhouse's performance. Most connected greenhouse solutions can use simple devices which support multiple types of sensors, use low-rate wireless connectivity, consume little power and can be inserted into soil or attached to stems. The communication between IoT devices and the Internet can be achieved through a standard IEEE 802.15.4 mesh network, through which nodes exchange data and forward messages sent by sensors until they reach the gateway. Several gateways can be installed across the greenhouse, enabling sensors to connect to the Internet and push data to the server. The cost of a custom IoT greenhouse solution is estimated at $100-150 thousand [Andrei Klubnikin b]. The sum covers the development of embedded systems, a web-based application and a mobile client for alert notifications, as well as consulting services regarding the choice of hardware components.

The second step is to create a PoC. According to the above project description, our NDN-802.15.4 architecture includes all the features needed to create a working prototype, including security. With the working prototype, measurements and simulation results, we believe that amazing demonstrations can be made and particular interest can be gained from investors.

Ongoing and Future Work

At the time of writing this manuscript, work is still in progress. First, the proposed ND-CSMA scheme has been designed for a binary-tree topology. Thus, it is not intended to work on complex network such as grid topologies with mobility. Results obtained with the R-LF strategy suggest that a more sophisticated CSMA adaptation should be envisioned by exploiting R-LF information such as the cost-to-go. That is, a better forwarding strategy may be obtained by combining the strengths of R-LF with the efficiency of ND-CSMA to reduce round-trip time. This idea is currently being investigated. The second ongoing work concerns the implementation of the testbed, which we aim to improve with a more efficient NDN stack that includes lightweight PIT and FIB to reduce memory usage, for example based on bloom filters. Finally, the last work realized in this thesis was the analytical model for wireless forwarding with NDN. However, it is currently simple and restricted to the basic NDN forwarding strategy in a binary-tree topology. The next step is to support more complex topologies in the model. We aim to model a network topology similar to the DAG used in RPL, and make further analytical comparisons between RPL and NDN in a wireless mesh network. In the meantime, the NDN-OMNeT simulation framework is getting additional features in order to provide users with more possibilities.

Publications

1. [**Published**] A. Abane, M. Daoui, S. Bouzefrane, and P. Muhlethaler. *"NDN-over-ZigBee: A ZigBee support for Named Data Networking"*. Future Generation Computer Systems, Volume 93, Pages 792-798. March 2017.

2. [**Published**] A. Abane, P. Muhlethaler, M. Daoui and H. Afifi. *"A Down-to-Earth Integration of Named Data Networking in the Real-World IoT"*. 6th International Conference on Future Internet of Things and Cloud Workshops. August 2018, Barcelona (Spain).

3. [**Published**] A. Abane, P. Muhlethaler, S. Bouzefrane, M. Daoui, and A. Battou. *"Towards evaluating named data networking for the IoT: A framework for OMNeT++"*. OMNeT++ Community Meeting. September 2018, Pisa (Italy).

4. [**Published**] A. Abane, M. Daoui, S. Bouzefrane, S. Banerjee, and P. Muhlethaler. *"A Realistic Deployment of Named Data Networking in the Internet of Things"*. Journal of Cyber Security and Mobility. June 2019.

5. [**Published**] A. Abane, P. Muhlethaler, S. Bouzefrane, and A. Battou. *"Broadcast-Based Yet Lightweight Forwarding for Wireless Named Data Networking"*. Named Data Networking Community Meeting 2019. September 2019, Gaithersburg, MD (USA).

6. [**Published**] A. Abane, M. Daoui, S. Bouzefrane, and P. Muhlethaler. *"A lightweight forwarding strategy for Named Data Networking in low-end IoT"*. Journal of Network and Computer Applications. September 2019.

7. [**Published**] A. Abane, P. Muhlethaler, S. Bouzefrane, and A. Battou. *"NDN over*

*IEEE 802.15.4: Modeling and Challenges". The 8th IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN2019).

Bibliography

M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos. Information-centric networking for the internet of things: challenges and opportunities. *IEEE Network*, 30(2):92–100, March 2016. ISSN 0890-8044. doi: 10.1109/MNET.2016.7437030.

Wikipedia. 6LoWPAN, a. URL `https://fr.wikipedia.org/wiki/6LoWPAN`.

RS Components Ltd. What is the Internet of Things? URL `https://uk.rs-online.com/web/generalDisplay.html?id=i/iot-internet-of-things`.

Eclipse IoT White Paper. The three software stacks required for iot architectures, 2017. URL `https://iot.eclipse.org/resources/white-papers/EclipseIoTWhitePaper-TheThreeSoftwareStacks2RequiredforIoTArchitectures.pdf`.

Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in IoT networking via TCP/IP architecture. Technical Report NDN-0038, NDN, February 2016a.

Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009a. ACM. ISBN 978-1-60558-636-6. doi: 10.1145/1658939.1658941. URL `http://doi.acm.org/10.1145/1658939.1658941`.

Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named Data Networking. *ACM SIG-COMM Computer Communication Review*, 44(3):66–77, July 2014.

Marica Amadeo, Claudia Campolo, Antonio Iera, and Antonella Molinaro. "named data networking for iot: an architectural perspective". *Conference on European Networks and Communications (EuCNC)*, pages 1–5, June 2014a.

W. Shang, A. Afanasyev, and L. Zhang. The design and implementation of the ndn protocol stack for riot-os. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, Dec 2016b. doi: 10.1109/GLOCOMW.2016.7849061.

Wikipedia. Ad-hoc On-demand Distance Vector, b. URL `https://fr.wikipedia.org/wiki/Ad-hoc_On-demand_Distance_Vector`.

Meisel Michael, Pappas Vasileios, and Zhang Lixia. Listen first, broadcast later: Topology-agnostic forwarding under high dynamics. In *Annual conference of international technology alliance in network and information science*, 2010.

Andrei Klubnikin. Internet of things: How much does it cost to build iot solution?, a. URL `https://r-stylelab.com/company/blog/iot/internet-of-things-how-much-does-it-cost-to-build-iot-solution`.

IoT online store. IoT by Numbers. URL `http://www.iotonlinestore.com/IoT-by-Numbers/10`.

Hemendra Singh. How much does it cost to develop an iot application? URL `http://customerthink.com/how-much-does-it-cost-to-develop-an-iot-application/`.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. "the internet of things: A survey". *Computer Networks Elsevier*, pages 2787–2805, 2010.

LoRa Alliance. "website LoRa Alliance". URL `https://www.lora-alliance.org/`.

Sigfox. "website Sigfox". URL `http://www.sigfox.com/`.

Andrei Klubnikin. IoT Agriculture: How to Build Smart Greenhouse?, b. URL `https://r-stylelab.com/company/blog/iot/iot-agriculture-how-to-build-smart-greenhouse`.

John Mueller. "Understanding SOAP and REST Basics And Differences", 2013. URL https://smartbear.com/blog/test-and-monitor/understanding-soap-and-rest-basics/.

Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014. URL https://rfc-editor.org/rfc/rfc7252.txt.

Klaus Hartke. Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641, September 2015. URL https://rfc-editor.org/rfc/rfc7641.txt.

OASIS.

IBM. Mqtt for sensor networks (mqtt-sn), 2013. URL http://www.mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf.

ZigBee Alliance. "what is ZigBee?". URL http://www.zigbee.org/what-is-zigbee/.

Carsten Bormann, Mehmet Ersue, and Ari Keränen. Terminology for Constrained-Node Networks. RFC 7228, May 2014. URL https://rfc-editor.org/rfc/rfc7228.txt.

Gabriel Montenegro, Jonathan Hui, David Culler, and Nandakishore Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007. URL https://rfc-editor.org/rfc/rfc4944.txt.

Lars-Erik Jonsson, Kristofer Sandlund, and Ghyslain Pelletier. RObust Header Compression (ROHC): ROHC over Channels That Can Reorder Packets. RFC 4224, January 2006. URL https://rfc-editor.org/rfc/rfc4224.txt.

Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Hui, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012. URL https://rfc-editor.org/rfc/rfc6550.txt.

Hanane Lamaazi, Nabil Benamar, and Antonio J. Jara. Rpl-based networks in static and mobile environment: A performance assessment analysis. *Journal of King Saud University - Computer and Information Sciences*, 30(3):320 – 333, 2018. ISSN 1319-1578.

doi: https://doi.org/10.1016/j.jksuci.2017.04.001. URL http://www.sciencedirect.com/science/article/pii/S1319157816301574.

Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. RFC 6763, February 2013. URL https://rfc-editor.org/rfc/rfc6763.txt.

Kerry Lynn, Peter Van der Stok, Michael Koster, and Christian Amsüss. CoRE Resource Directory: DNS-SD mapping. Internet-Draft draft-ietf-core-rd-dns-sd-04, Internet Engineering Task Force, March 2019. URL https://datatracker.ietf.org/doc/html/draft-ietf-core-rd-dns-sd-04. Work in Progress.

Michael Meisel, Vasileios Pappas, and Lixia Zhang. Ad hoc networking via named data. In *Proceedings of the Fifth ACM International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '10, pages 3–8, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0143-5. doi: 10.1145/1859983.1859986. URL http://doi.acm.org/10.1145/1859983.1859986.

Charles E. Perkins. IP Mobility Support for IPv4. RFC 3344, August 2002. URL https://rfc-editor.org/rfc/rfc3344.txt.

Sanjit Biswas and Robert Morris. Exor: Opportunistic multi-hop routing for wireless networks. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 133–144, New York, NY, USA, 2005. ACM. ISBN 1-59593-009-4. doi: 10.1145/1080091.1080108. URL http://doi.acm.org/10.1145/1080091.1080108.

Sheila Frankel and Suresh Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, February 2011. URL https://rfc-editor.org/rfc/rfc6071.txt.

Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. URL https://rfc-editor.org/rfc/rfc5246.txt.

Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012. URL https://rfc-editor.org/rfc/rfc6347.txt.

Göran Selander, John Mattsson, Francesca Palombini, and Ludwig Seitz. Object Security for Constrained RESTful Environments (OSCORE). Internet-Draft draft-ietf-core-object-security-16, Internet Engineering Task Force, March 2019. URL `https://datatracker.ietf.org/doc/html/draft-ietf-core-object-security-16`. Work in Progress.

Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. "A Survey of Information-Centric Networking". *IEEE Communications Magazine*, 50(7):26–36, July 2012.

Dirk Kutscher, Suyong Eum, Kostas Pentikousis, Ioannis Psaras, Daniel Corujo, Damien Saucez, Thomas C. Schmidt, and Matthias Wählisch. Information-Centric Networking (ICN) Research Challenges. RFC 7927, July 2016. URL `https://rfc-editor.org/rfc/rfc7927.txt`.

B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, July 2012. ISSN 0163-6804. doi: 10.1109/MCOM.2012.6231276.

G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys Tutorials*, 16(2):1024–1049, Second 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.070813.00063.

Brent Baccala. Data-oriented networking. Internet-Draft draft-baccala-data-networking-00, Internet Engineering Task Force, August 2002. URL `https://datatracker.ietf.org/doc/html/draft-baccala-data-networking-00`. Work in Progress.

Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, August 2007. ISSN 0146-4833. doi: 10.1145/1282427.1282402. URL `http://doi.acm.org/10.1145/1282427.1282402`.

Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the*

5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '09, pages 1–12, New York, NY, USA, 2009b. ACM. ISBN 978-1-60558-636-6. doi: 10.1145/1658939.1658941. URL http://doi.acm.org/10.1145/1658939.1658941.

Lixia Zhang, Deborah Estrin, and Jeffrey Burke. "named data networking NDN project". Technical Report NDN-0001, NDN, October 2010. URL http://named-data.net/techreports.html.

NDN Website. NDN Protocol Design Principles. URL https://named-data.net/project/ndn-design-principles/.

NDN Team. "NDN Packet Format Specification". URL http://named-data.net/doc/NDN-packet-spec/current/.

NDN Project Team. "NDN technical memo: Naming conventions". Technical Report NDN-0022, NDN, July 2014. URL https://named-data.net/wp-content/uploads/2014/08/ndn-tr-22-ndn-memo-naming-conventions.pdf.

Alexander Afanasyev, Junxiao Shi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Packet fragmentation in NDN: Why NDN uses hop-by-hop fragmentation. Technical Report 0032, NDN, May 2015. URL https://named-data.net/publications/techreports/ndn-0032-1-ndn-memo-fragmentation/.

Junxiao Shi and Beichuan Zhang. NDNLP: A Link Protocol for NDN. Technical Report NDN-0006, NDN, July 2012.

Marc Mosko and Christian Tschudin. ICN "Begin-End" Hop by Hop Fragmentation. Internet-Draft draft-mosko-icnrg-beginendfragment-02, Internet Engineering Task Force, December 2016. URL https://datatracker.ietf.org/doc/html/draft-mosko-icnrg-beginendfragment-02. Work in Progress.

B. Etefia, M. Gerla, and L. Zhang. Supporting military communications with named data networking: An emulation analysis. In *MILCOM 2012 - 2012 IEEE Military Communications Conference*, pages 1–6, Oct 2012. doi: 10.1109/MILCOM.2012.6415685.

Gareth Tyson, Nishanth Sastry, Ivica Rimac, Ruben Cuevas, and Andreas Mauthe. A survey of mobility in information-centric networks: Challenges and research directions. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 1–6, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1291-2. doi: 10.1145/2248361.2248363. URL `http://doi.acm.org/10.1145/2248361.2248363`.

Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. "schematizing trust in named data networking". In *The ACM Conf. on Information-Centric Networking (ICN'15)*, pages 177–186, 2015.

Z. Zhang, Y. Yu, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, and L. Zhang. An overview of security support in named data networking. *IEEE Communications Magazine*, 56(11):62–68, November 2018. ISSN 0163-6804. doi: 10.1109/MCOM.2018.1701147.

Divya Saxena, Vaskar Raychoudhury, Neeraj Suri, Christian Becker, and Jiannong Cao. Named data networking: A survey. *Computer Science Review*, 19:15 – 55, 2016. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2016.01.001. URL `http://www.sciencedirect.com/science/article/pii/S1574013715300599`.

S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo. Recent advances in information-centric networking based internet of things (icn-iot). *IEEE Internet of Things Journal*, pages 1–1, 2019. ISSN 2327-4662. doi: 10.1109/JIOT.2018.2873343.

Wentao Shang, Adeola Bannisy, Teng Liangz, Zhehao Wangx, Yingdi Yu, Alexander Afanasyev, Jeff Thompsonx, Jeff Burkex, Beichuan Zhangz, and Lixia Zhang. Named Data Networking of Things (Invited paper). In *The 1st IEEE Intl. Conf. on Internet-of-Things Design and Implementation*, Berlin, Germany, April 2016c. URL `https://named-data.net/publications/ndn-iotdi-2016/`.

Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. Information centric networking in the iot: Experiments with ndn in the wild. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-

ICN '14, pages 77–86, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3206-4. doi: 10.1145/2660129.2660144. URL http://doi.acm.org/10.1145/2660129.2660144.

E. Baccelli, C. Gündoğan, O. Hahm, P. Kietzmann, M. S. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wählisch. Riot: An open source operating system for low-end embedded devices in the iot. *IEEE Internet of Things Journal*, 5(6): 4428–4440, Dec 2018. doi: 10.1109/JIOT.2018.2815038.

Cenk Gündogan, Peter Kietzmann, Martine S. Lenders, Hauke Petersen, Thomas C. Schmidt, and Matthias Wählisch. Ndn, coap, and mqtt: A comparative measurement study in the iot. *CoRR*, abs/1806.01444, 2018a.

Antonio Carzaniga, Michele Papalini, and Alexander L. Wolf. Content-based publish/subscribe networking and information-centric networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 56–61, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0801-4. doi: 10.1145/2018584.2018599. URL http://doi.acm.org/10.1145/2018584.2018599.

Marica Amadeo, Claudia Campolo, and Antonella Molinaro. "internet of things via named data networking: The support of push traffic". In *International Conference and Workshop on the Network of the Future (NOF)*, pages 1–5, 2014b.

Wentao Shang, Yingdi Yu, Teng Liangy, Beichuan Zhangy, and Lixia Zhang. "NDN-ACE: Access control for constrained environments over named data networking". Technical Report NDN-0036, NDN, December 2015. URL https://named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf.

G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang. Vanet via named data networking. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 410–415, April 2014. doi: 10.1109/INFCOMW.2014.6849267.

Loïc Dauphin, Emmanuel Baccelli, Cedric Adjih, and Hauke Petersen. NDN-based IoT Robotics. ACM ICN'17 - 4th ACM Conference on Information-Centric Networking, September 2017. URL https://hal.inria.fr/hal-01666539. Poster.

Cenk Gündogan, Peter Kietzmann, Thomas C. Schmidt, Martine Lenders, Hauke Petersen, Matthias Wählisch, Michael Frey, and Felix Shzu-Juraschek. Information-centric networking for the industrial iot. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, pages 214–215, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5122-5. doi: 10.1145/3125719.3132099. URL `http://doi.acm.org/10.1145/3125719.3132099`.

Marica Amadeo, Claudia Campolo, Antonella Molinaro, and Giuseppe Ruggeri. Content-centric wireless networking: A survey. *Computer Networks*, 72:1 – 13, 2014c. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2014.07.003. URL `http://www.sciencedirect.com/science/article/pii/S1389128614002497`.

G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida. Navigo: Interest forwarding by geolocations in vehicular named data networking. In *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–10, June 2015. doi: 10.1109/WoWMoM.2015.7158165.

Z. Ren, M. A. Hail, and H. Hellbrück. CCN-WSN - A lightweight, flexible Content-Centric Networking protocol for wireless sensor networks. In *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 123–128, April 2013. doi: 10.1109/ISSNIP.2013.6529776.

M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton. Named data networking: A natural design for data collection in wireless sensor networks. In *2013 IFIP Wireless Days (WD)*, pages 1–6, Nov 2013. doi: 10.1109/WD.2013.6686486.

M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer. Caching in named data networking for the wireless internet of things. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6, April 2015. doi: 10.1109/RIOT.2015.7104902.

J. Shi, E. Newberry, and B. Zhang. On broadcast-based self-learning in named data networking. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9, June 2017.

Peter Kietzmann, Cenk Gündogan, Thomas C. Schmidt, Oliver Hahm, and Matthias Wählisch. The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain. In *ACM ICN 2017 - 4th ACM Conference on Information-Centric Networking* , Berlin, Germany, September 2017. URL `https://hal.inria.fr/hal-01666601`.

Junxiao Shi, Teng Liang, Hao Wu, Bin Liu, and Beichuan Zhang. Ndn-nic: Name-based filtering on network interface card. In *ICN*, 2016.

Lucas Wang, Alexander Afanasyev, Romain Kuntz, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. Rapid traffic information dissemination using named data. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 7–12, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1291-2. doi: 10.1145/2248361.2248365. URL `http://doi.acm.org/10.1145/2248361.2248365`.

Carofiglio G., Morabito G., Muscariello L., Solis I., and Varvello M. From content delivery today to information centric networking. *Computer Networks*, 57(16):3116 – 3127, 2013. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2013.07.002. URL `http://www.sciencedirect.com/science/article/pii/S1389128613002156`. Information Centric Networking.

WANG Guo-qing, HUANG Tao, LIU Jiang, CHEN Jian-ya, and LIU Yun-jie. Modeling in-network caching and bandwidth sharing performance in information-centric networking. *The Journal of China Universities of Posts and Telecommunications*, 20(2):99 – 105, 2013. ISSN 1005-8885. doi: https://doi.org/10.1016/S1005-8885(13)60035-7. URL `http://www.sciencedirect.com/science/article/pii/S1005888513600357`.

Ren Yongmao, Li Jun, Li Lingling, Shi Shanshan, Zhi Jiang, and Wu Haibo. Modeling content transfer performance in information-centric networking. *Future Generation Computer Systems*, 74:12 – 19, 2017. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2017.04.013. URL `http://www.sciencedirect.com/science/article/pii/S0167739X17305861`.

Udugama A, Palipana S., and Goerg C. Analytical characterisation of multi-path content delivery in content centric networks. In *2013 Conference on Future Internet Communications (CFIC)*, pages 1–7, May 2013. doi: 10.1109/CFIC.2013.6566319.

Carofiglio G., Gallo M., Muscariello L., and Perino D. Modeling data transfer in content-centric networking. In *2011 23rd International Teletraffic Congress (ITC)*, pages 111–118, Sep. 2011.

NDN. NFD - Named Data Networking Forwarding Daemon, a. URL `http://named-data.net/doc/NFD/current/`.

NDN. Named Data Network Internet of Things Toolkit (NDN-IoTT), b. URL `https://github.com/remap/ndn-pi`.

Wentao Shang, Qiuhan Ding, Alessandro Marianantoni, Jeff Burke, and Lixia Zhang. Securing building management systems using named data networking. *IEEE Network Journal*, 28(3):50–56, 2014.

NDN. NDN Common Client Libraries (NDN-CCL) Documentation, c. URL `https://named-data.net/codebase/platform/ndn-ccl/`.

NDN. ndn-tools, d. URL `https://github.com/named-data/ndn-tools`.

NDN. NDN Testbed status page, e. URL `http://ndndemo.arl.wustl.edu/`.

Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnsim: ndn simulator for ns-3. 01 2012.

NDN. What is Mini-NDN?, f. URL `http://minindn.memphis.edu/`.

George Xylomenos, Yannis Thomas, Xenofon Vasilakos, Michael Georgiades, Alexander Phinikarides, Ioannis Doumanis, Stuart Porter, Dirk Trossen, Sebastian Robitzsch, Martin J. Reed, Mays F. Al-Naday, George Petropoulos, Konstantinos V. Katsaros, Maria-Evgenia Xezonaki, and Janne Riihijärvi. IP over ICN goes live. *CoRR*, abs/1804.07511, 2018. URL `http://arxiv.org/abs/1804.07511`.

Luca Muscariello, Giovanna Carofiglio, Jordan Auge, and Michele Papalini. Hybrid Information-Centric Networking. Internet-Draft draft-muscariello-intarea-hicn-01, Internet Engineering Task Force, December 2018. URL `https://datatracker.ietf.org/doc/html/draft-muscariello-intarea-hicn-01`. Work in Progress.

H. Wu, J. Shi, Y. Wang, Y. Wang, G. Zhang, Y. Wang, B. Liu, and B. Zhang. On incremental deployment of named data networking in local area networks. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 82–94, May 2017. doi: 10.1109/ANCS.2017.18.

Teng Liang, Ju Pan, and Beichuan Zhang. Ndnizing existing applications: Research issues and experiences. In *5th ACM Conference onInformation-Centric Networking (ICN '18)*, September 2018. doi: 10.1145/3267955.3267969.

IEEE. Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016. doi: 10.1109/IEEESTD.2016.7786995.

IEEE. Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, Sep. 2011. doi: 10.1109/IEEESTD.2011.6012487.

IEEE. Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.1a: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpan). *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pages 1–700, June 2005. doi: 10.1109/IEEESTD.2005.96290.

IEEE. Ieee draft trial-use standard for wireless access in vehicular environments (wave) - resource manager. *IEEE Std P1609.1/D17, Jul 2006*, pages 1–66, April 2019.

PrithviRaj Narendra, Simon Duquennoy, and Thiemo Voigt. Ble and ieee 802.15.4 in the iot: Evaluation and interoperability considerations. In Benny Mandler, Johann

Marquez-Barja, Miguel Elias Mitre Campista, Dagmar Cagáňová, Hakima Chaouchi, Sherali Zeadally, Mohamad Badra, Stefano Giordano, Maria Fazio, Andrey Somov, and Radu-Laurentiu Vieriu, editors, *Internet of Things. IoT Infrastructures*, pages 427–438, Cham, 2016. Springer International Publishing. ISBN 978-3-319-47075-7.

Thomas Watteyne, Maria Rita Palattella, and Luigi Alfredo Grieco. Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement. RFC 7554, May 2015. URL `https://rfc-editor.org/rfc/rfc7554.txt`.

Pascal Thubert. An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4. Internet-Draft draft-ietf-6tisch-architecture-20, Internet Engineering Task Force, March 2019. URL `https://datatracker.ietf.org/doc/html/draft-ietf-6tisch-architecture-20`. Work in Progress.

Matti Siekkinen, Markus Hiienkari, Jukka K. Nurminen, and Johanna Nieminen. *How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4*, pages 232–237. 2012. ISBN 978-1-4673-0682-9. doi: 10.1109/WCNCW.2012.6215496.

Arjun Attam and Ilya Moiseenkoy. "NDNBlue: NDN over bluetooth". Technical Report NDN-0015, November 2013. URL `http://named-data.net/techreports.html`.

Hauke Petersen, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. Ndn meets ble: A transparent gateway for opening ndn-over-ble networks to your smartphone. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, pages 175–176, New York, NY, USA, 2019a. ACM. ISBN 978-1-4503-6970-1. doi: 10.1145/3357150.3357411. URL `http://doi.acm.org/10.1145/3357150.3357411`.

Hauke Petersen, Peter Kietzmann, Cenk Gündoğnan, Thomas C. Schmidt, and Matthias Wählisch. Bluetooth mesh under the microscope: How much icn is inside? In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, ICN '19, pages 134–140, New York, NY, USA, 2019b. ACM. ISBN 978-1-4503-6970-1. doi: 10.1145/3357150.3357398. URL `http://doi.acm.org/10.1145/3357150.3357398`.

Van Jackobson. Compressing TCP/IP Headers for Low-Speed Serial Links. RFC 1144, February 1990. URL `https://rfc-editor.org/rfc/rfc1144.txt`.

Cenk Gündoğran, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. Icn-lowpan: Header compression for the constrained iot. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, ICN '18, pages 184–185, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5959-7. doi: 10.1145/3267955.3269006. URL `http://doi.acm.org/10.1145/3267955.3269006`.

Cenk Gündogan, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. Icn-lowpan - named-data networking for low power iot networks. *2019 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2018b.

Oliver Hahm, Emmanuel Baccelli, Thomas C. Schmidt, Matthias Wählisch, Cédric Adjih, and Laurent Massoulié. Low-power internet of things with ndn &#38; cooperative caching. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, pages 98–108, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5122-5. doi: 10.1145/3125719.3125732. URL `http://doi.acm.org/10.1145/3125719.3125732`.

X. Xu, H. Zhang, T. Li, and L. Zhang. Achieving resilient data availability in wireless sensor networks. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, May 2018. doi: 10.1109/ICCW.2018.8403581.

C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. Fit iot-lab: A large scale open experimental iot testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464, Dec 2015. doi: 10.1109/WF-IoT.2015.7389098.

Wikipedia. "Raspberry Pi". URL `https://en.wikipedia.org/wiki/Raspberry_Pi`.

Digi International Inc. XBee/XBee-PRO DigiMesh 2.4 OEM RF modules. Technical report, Digi International, 2008. URL `http://www.digi.com/products/xbee-rf-solutions/modules/xbee-802-15-4`.

Cody-Kenny Brendan, Guerin David, Ennis Desmond, Simon Carbajo Ricardo, Huggard Meriel, and Mc Goldrick Ciaran. Performance evaluation of the 6lowpan protocol on micaz and telosb motes. In *Proceedings of the 4th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, PM2HW2N '09, pages 25–30, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-621-2. doi: 10.1145/1641913.1641917. URL http://doi.acm.org/10.1145/1641913.1641917.

G. Gardasevic, S. Mijovic, A. Stajkic, and C. Buratti. On the performance of 6lowpan through experimentation. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 696–701, Aug 2015. doi: 10.1109/IWCMC.2015.7289168.

C. Gibson, P. Bermell-Garcia, K. Chan, B. Ko, A. Afanasyev, and L. Zhang. Opportunities and challenges for named data networking to increase the agility of military coalitions. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6, Aug 2017. doi: 10.1109/UIC-ATC.2017.8397416.

OMNet++. OMNeT++ discrete event simulator. URL https://www.omnetpp.org/.

INET. "INET Framework". URL https://inet.omnetpp.org/.

Spyridon Mastorakis, Kevin Chan, Bongjun Ko, Alexander Afanasyev, and Lixia Zhang. Experimentation with fuzzy interest forwarding in named data networking. *CoRR*, abs/1802.03072, 2018. URL http://arxiv.org/abs/1802.03072.

Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications*, 50(Supplement C):148 – 158, 2015. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2014.06.007. URL http://www.sciencedirect.com/science/article/pii/S1084804514001404.

L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE*

Computer and Communications Societies. The Future is Now (Cat. No.99CH36320), volume 1, pages 126–134 vol.1, March 1999. doi: 10.1109/INFCOM.1999.749260.

Samir R. Das, Charles E. Perkins, and Elizabeth M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, July 2003. URL `https://rfc-editor.org/rfc/rfc3561.txt`.

Y. T. Yu, R. B. Dilmaghani, S. Calo, M. Y. Sanadidi, and M. Gerla. Interest propagation in named data manets. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, pages 1118–1122, Jan 2013. doi: 10.1109/ICCNC.2013.6504249.

Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 671–678, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=2987189.2987274`.

Y. Zhang and M. Fromherz. Constrained flooding: a robust and efficient routing framework for wireless sensor networks. In *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, volume 1, pages 6 pp.–, April 2006. doi: 10.1109/AINA.2006.132.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

Shailesh Kumar. Confidence-based dual reinforcement q-routing: an on-line adaptive network routing algorithm. Master's thesis, The University of Texas at Austin, 1998.

Amar Abane, Paul Muhlethaler, Samia Bouzefrane, Mehammed Daoui, and abdella battou. Towards evaluating named data networking for the iot: A framework for omnet++. 09 2018.

Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio, Dario Rossi, and Giuseppe Rossini. Inform: A dynamic interest forwarding mechanism for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric*

Networking, ICN '13, pages 9–14, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2179-2. doi: 10.1145/2491224.2491227. URL http://doi.acm.org/10.1145/2491224.2491227.

B. Fu, L. Qian, Y. Zhu, and L. Wang. Reinforcement learning-based algorithm for efficient and adaptive forwarding in named data networking. In *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 1–6, Oct 2017. doi: 10.1109/ICCChina.2017.8330354.

Olumide Akinwande. Interest forwarding in named data networking using reinforcement learning. *Sensors*, 18(10), 2018. ISSN 1424-8220. doi: 10.3390/s18103354. URL https://www.mdpi.com/1424-8220/18/10/3354.

Yi Zhang, Bo Bai, Kuai Xu, and Kai Lei. Ifs-rl: An intelligent forwarding strategy based on reinforcement learning in named-data networking. In *Proceedings of the 2018 Workshop on Network Meets AI & ML*, NetAI'18, pages 54–59, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5911-5. doi: 10.1145/3229543.3229547. URL http://doi.acm.org/10.1145/3229543.3229547.

Lu ZHAO, Guang wei BAI, Hang SHEN, and Zhen min TANG. Priority-based ieee 802.15.4 csma/ca mechanism for wsns. *The Journal of China Universities of Posts and Telecommunications*, 20(1):47 – 53, 2013. ISSN 1005-8885. doi: https://doi.org/10.1016/S1005-8885(13)60006-0. URL http://www.sciencedirect.com/science/article/pii/S1005888513600060.

Amar ABANE

A Realistic Named Data Networking Architecture for the Internet of Things

Résumé :

L'Internet des objets (IdO) utilise l'interconnexion de milliards de petits appareils informatiques, appelés «Objets», pour fournir un accès à des services et à des informations partout dans le monde. Cependant, la suite de protocoles IP a été conçue il y a plusieurs décennies dans un but totalement différent, et les fonctionnalités de l'IoT soulignent désormais les limites de l'IP. En parallèle aux efforts d'adaptation de l'IP à l'IdO, des architectures alternatives basées sur les réseaux orientés information promettent de satisfaire nativement les applications Internet émergentes. L'une de ces architectures est appelée réseau de données nommées (NDN). Nos objectifs à travers le travail rapporté dans ce manuscrit peuvent être résumés en deux aspects. Le premier objectif est de montrer que NDN est adapté à la prise en charge des systèmes IdO. Le deuxième objectif est la conception de deux solutions de communication legères pour les réseaux sans fil contraints avec NDN.

Mots clés :

Réseaux de données nommées, Réseaux orientés information, Internet des Objets, IEEE 802.15.4, Réseaux sans fil.

Abstract :

The Internet of Things (IoT) uses the interconnection of billions of small computing devices, called "Things", to provide access to services and information all over the world. However, the IP protocol suite has been designed decades ago for a completely different purpose, and IoT features now highlight the limitations of IP. While adapting IP for the IoT might be seen as cutting corners, alternative architectures based on the Information Centric Networking (ICN) paradigm promise to natively satisfy emerging Internet applications. One of these architectures is Named Data Networking (NDN). Our objectives through the work reported in this manuscript can be summarized in two aspects. The first objective is to show that NDN is suitable to support IoT networking. The second objective is the design of two solutions for lightweight forwarding in constrained wireless networks.

Keywords :

NDN, ICN, IoT, IEEE 802.15.4, Broadcast, Wireless Networks