THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut National Polytechnique de Toulouse

Présentée et soutenue par

Adrien THIBAUD

Le 2 septembre 2021

Répartition de flux dans les réseaux de contenu, application à un contexte satellite

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Emmanuel CHAPUT

Jury

M. Razvan STANICA, Rapporteur
M. Toufik AHMED, Rapporteur
M. Julien FASSON, Examinateur
M. Fabrice ARNAL, Examinateur
M. Thierry TURLETTI, Examinateur
Mme Véronique VèQUE, Examinatrice
M. Emmanuel CHAPUT, Directeur de thèse

M. Emmanuel Dubois, invité
M. Renaud Sallantin, invité

Résumé

Avec l'apparition de service de vidéo à la demande tel que Netflix, l'utilisation du streaming a explosé ces dernières années. L'important volume de données engendré oblige les opérateurs réseaux à définir et utiliser de nouvelles solutions. Ces solutions même si elles restent fondées sur la pile IP, cherchent à contourner la communication point à point entre deux hôtes (CDN, P2P, ...). Dans cette thèse, nous nous intéressons à une nouvelle approche, l'Information Centric Networking, qui cherche à déconstruire le modèle d'IP en se focalisant sur le contenu recherché. L'utilisateur indique ainsi au réseau qu'il souhaite obtenir une donnée et le réseau s'occupe de lui rapatrier ce contenu. Avec la pile IP, un hôte peut envoyer un message à un autre hôte. Il faut monter au niveau applicatif pour qu'un utilisateur puisse récupérer un contenu. Parmi les nombreuses architectures proposées dans la littérature, Named Data Networking (NDN) nous semble l'architecture la plus mature.

Pour que NDN soit une réelle opportunité pour Internet, il lui faut offrir une meilleure Qualité d'Expérience (QoE) aux utilisateurs tout en utilisant efficacement les capacités des réseaux. C'est le cœur de cette thèse : proposer une solution à NDN pour gérer au mieux la satisfaction des utilisateurs. Pour des contenus tels que de la vidéo, le débit est crucial. C'est pourquoi nous avons pris le parti de maximiser ce dernier pour maximiser la QoE. Les nouvelles opportunités offertes par les NDN, telles que le multi-chemin et la mise en cache, nous ont permis de redéfinir la notion de flux dans ce paradigme. Avec cette définition et la possibilité d'effectuer du traitement sur chaque nœud du réseau, nous avons décidé de voir le problème classique du contrôle de congestion comme la recherche d'une répartition équitable des flux. Pour que la QoE des utilisateurs soit optimale, cette répartition devra répondre au mieux aux demandes. Cependant, comme les ressources du réseau ne sont pas infinies, des compromis doivent être faits. Pour cela, nous avons décidé d'utiliser le critère d'équité Max-Min qui permet d'obtenir un équilibre de Pareto où l'augmentation d'un flux ne peut se faire qu'au détriment d'un autre flux moins privilégié.

L'objectif de cette thèse a ensuite été de proposer une solution au problème nouvellement formulé. Nous avons donc conçu Cooperative Congestion Control, une solution distribuée ayant pour but de répartir les flux équitablement sur le réseau. Elle se base sur une coopération de chacun des nœuds où les besoins des utilisateurs sont transmis jusqu'aux fournisseurs de contenu et où les contraintes du réseau sont ré-évaluées localement et transmises jusqu'aux utilisateurs. L'architecture de notre solution est générique et est composée de plusieurs algorithmes. Nous proposons quelques implantations de ceux-ci et montrons que même si un équilibre de Pareto est obtenu, seule une équité locale est atteinte. En effet, par manque d'information, les décisions prises par les nœuds sont limitées. Nous avons aussi éprouvé notre solution sur des topologies comprenant des liens satellites (proposant de hauts délais). Grâce à l'émission des Interests régulée par notre solution, nous montrons que ces hauts délais, et contrairement aux solutions de l'état de l'art, n'ont que très peu d'impacts sur les performances de CCC.

Abstract

With the emergence of video-on-demand services such as Netflix, the use of streaming has exploded in recent years. The large volume of data generated forces network operators to define and use new solutions. These solutions, even if they remain based on the IP stack, try to bypass the point-to-point communication between two hosts (CDN, P2P, ...). In this thesis, we are interested in a new approach, Information Centric Networking, which seeks to deconstruct the IP model by focusing on the desired content. The user indicates to the network that he wishes to obtain a data and the network takes care of retrieving this content. Among the many architectures proposed in the literature, Named Data Networking (NDN) seems to us to be the most mature architecture.

For NDN to be a real opportunity for the Internet, it must offer a better Quality of Experience (QoE) to users while efficiently using network capacities. This is the core of this thesis : proposing a solution to NDN to manage user satisfaction. For content such as video, throughput is crucial. This is why we have decided to maximize the throughput to maximize the QoE. The new opportunities offered by NDNs, such as multipathing and caching, have allowed us to redefine the notion of flow in this paradigm. With this definition and the ability to perform processing on every node in the network, we decided to view the classic congestion control problem as finding a fair distribution of flows. In order for the users' QoE to be optimal, this distribution will have to best meet the demands. However, since the network resources are not infinite, tradeoffs must be made. For this purpose, we decided to use the Max-Min fairness criterion which allows us to obtain a Pareto equilibrium where the increase of a flow can only be done at the expense of another less privileged flow.

The objective of this thesis was then to propose a solution to the newly formulated problem. We thus designed Cooperative Congestion Control, a distributed solution aiming at distributing the flows fairly on the network. It is based on a cooperation of each node where the users' needs are transmitted to the content providers and the network constraints are re-evaluated locally and transmitted to the users. The architecture of our solution is generic and is composed of several algorithms. We propose some implementations

of these and show that even if a Pareto equilibrium is obtained, only local fairness is achieved. Indeed, due to lack of information, the decisions made by the nodes are limited. We also tested our solution on topologies including satellite links (thus offering high delays). Thanks to the emission of Interests regulated by our solution, we show that these high delays, and contrary to state-of-the-art solutions, have very little impact on the performance of CCC.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

AIMD	*Additive Increase - Multiplicative Decrease*
AQM	*Active Queue Management*
BR	*Best Route*
CCC	*Cooperative Congestion Control*
CCN	*Content-Centric Networking*
CDN	*Content Delivery Network*
CS	*Content Store*
DAG	*Directed Acyclic Graph*
DASH	*Dynamic Adaptive Streaming over HTTP*
DONA	*a Data-Oriented Network Architecture*
DRF	*Dynamic Request Forwarding*
DTN	*Disruption/Delay Tolerant Networking*
FIB	*Forwarding Information Base*
FID	*Fowarding ID*
FIT	*Flow Information Table*
FPF	*Fast Pipeline Filling*
FS	*Forwarding Strategy*
GNRS	*Global Name Resolution Service*
HTTP	*HyperText Transfer Protocol*
ICN	*Information Centric Networking*
ICNRG	*Information-Centric Networking Research Group*
ICP	*Interest Control Protocol*
IFIL	*Input Face Info List*

IRTF	*Internet Research Task Force*
LID	*Link ID*
LIPSIN	*Line Speed Publish/Subscribe Inter-networking*
LPM	*Longest Prefix Match*
MTU	*Maximum Transmission Unit*
NACK	*Negative-Acknowledgement*
NDN	*Named Data Networking*
NetInf	*Network of Information*
NLSR	*Named-data Link State Routing Protocol*
OFIL	*Output Face Info List*
P2P	Réseaux pair-à-pair
PCON	*Practical Congestion cONtrol scheme*
PCON-CS	*PCON - Consumer Side*
PCON-FS	*PCON - Forwarding Strategy*
PEP	*Performance-Enhancing Proxies*
PI	*Pending Interests*
PIT	*Pending Interest Table*
POINT	*iP Over IcN - the betTer ip*
PSIRP	*Publish-Subscribe Internet Routing Paradigm*
PURSUIT	*Publish-Subscribe Internet Technology*
QoE	Qualité d'Expérience
QoS	Qualité de Service
RIFE	*architectuRe for an Internet For Everybody*
RTO	*Retransmission Timeout*
RTT	*Round Trip Time*
TCP	*Transmission Control Protocol*
TLV	*Type-Length-Value*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
XIA	*eXpressive Internet Architecture*

Introduction

Contexte et problématique de la thèse

En 2020, le service Youtube représentait à lui seul plus d'un milliard d'heures de vidéo diffusées sur un jour tandis que les utilisateurs de Netflix consommaient plus de 550 millions d'heures de vidéos (source DOMO [1]). Selon les mêmes sources, en une minute de temps une moyenne de 500 heures de vidéos (contre 6 heures en 2007 [2]) est mis en ligne sur la plateforme Youtube et ce n'est qu'une petite partie des 1.7 mégaoctets de données que crée chaque utilisateur d'Internet en une seconde. Ces quelques chiffres soulignent l'importance du contenu comme sa croissance dans Internet.

Pourtant le protocole à la base du réseau mondial, IP, n'a pas tant évolué que cela alors qu'il ne fournit aucune garantie de service. Comment offrir aux utilisateurs le contenu qu'ils souhaitent avec une Qualité d'Expérience (QoE) satisfaisante ? Ce problème n'est à l'évidence pas récent puisque, sans sa prise en compte, Internet n'aurait pas eu le succès actuel. Tout service de contenu est alors confronté à ces deux points : trouver un contenu dans un monde centré sur la localisation des équipements (adresse IP) d'une part, et d'autre part garantir une certaine qualité dans un monde qui en offre peu voire aucune.

Les nombreuses solutions répondent à des besoins des utilisateurs à une époque donnée du "net" : proxies-caches, miroirs, pair-à-pair (P2P) ou encore Content Delivery Network (CDN). Malgré leurs différences, toutes contournent la structure d'Internet en construisant un réseau au-dessus d'IP, un overlay. Aujourd'hui, les adresses IP utilisées ne reflètent plus la position réelle du serveur, alors pourquoi continuer à les utiliser ? Fort de ce constat, au début des années 2010, des projets ambitieux se positionnent comme une alternative aux solutions actuelles : construire un réseau centré sur la donnée. Il s'agit des Information Centric Networks (ICN).

1. Data never sleeps - `https://www.domo.com`
2. `https://tubularlabs.com/blog/hours-minute-uploaded-youtube/`

L'enjeu premier des ICNs est de proposer une architecture suffisamment mature pour être capable de remplacer tout ou partie de l'architecture actuelle d'Internet. Parmi elle, Named Data Networking (NDN) nous paraît la plus à même de proposer aujourd'hui un déploiement à large échelle. Toutefois il est vite apparu que de reconstruire un "nouvel Internet" rencontrerait les mêmes problèmes que l'ancien : adressage, routage, interconnexion, congestion... Ces difficultés ne doivent cependant pas occulter la question de la QoE.

L'objectif de notre travail est donc de proposer une solution protocolaire assurant de la QoE dans NDN. La QoE englobe beaucoup de caractéristiques et s'avère subjective. Pour les services de vidéo à la demande, la qualité de la vidéo est le critère principal de satisfaction des utilisateurs. Il faut aussi que la vidéo soit lue de manière continue et sans saccade. Ce service applicatif étant le principal service pouvant bénéficier de ce changement de paradigme, nous avons ainsi choisi une approche simple : lier l'évaluation de cette QoE directement au débit obtenu par l'utilisateur. Pour construire notre solution, nous avons procédé par étapes qui sont retranscrites dans le manuscrit sous la forme de chapitres. Le cas pratique des réseaux satellite a toujours été dans un coin de nos têtes (le satellite a historiquement un rôle fort dans la distribution de contenus vidéo avec la télévision par satellite) et a orienté certaines décisions de conception de notre solution. Il sera notamment étudié et évalué lors du dernier chapitre de ce mémoire.

Organisation du manuscrit

Le premier chapitre propose un état de l'art à différents niveaux (Chapitre 1). Tout d'abord, nous proposons une vue d'ensemble des différents projets ICNs (Section 1.2), leurs principaux choix ainsi que leur état d'avancement. C'est grâce à ce travail que nous avons choisi NDN comme objet d'étude. Nous le présentons plus en détail dans un second temps (Section 1.3). Une fois ces éléments présentés, un état de l'art est proposé sur les différentes solutions de la littérature pour la gestion des ressources dans NDN (Section 1.4), élément clé d'une bonne QoE pour les utilisateurs finals. Nous y présentons en particulier les différentes politiques de routage et les mécanismes de contrôle de congestion qui ne sont pas sans rappeler les solutions du monde IP. Nous terminons ce chapitre par une évaluation des solutions les plus matures (Section 1.4.5) via notre implantation sous ndnSIM. Il s'agit là d'une première contribution de notre thèse [5].

En se fondant sur cette première étude, le deuxième chapitre définit précisément le problème à traiter (Chapitre 2). Comme nous l'avons précisé auparavant, nous avons décidé de prendre comme métrique de QoE, le débit de l'application utilisateur, cela nécessite

de gérer les différents flux au travers du réseau. NDN offre de nouvelles opportunités avec du multi-source, du multi-chemin, du multi-consommateur, ou encore du "caching". Dans ce contexte, le terme "flux" peut prendre différents sens. Nous avons donc analysé les différents cas (Section 2.2) et défini la notion de flux dans le cadre de cette thèse (Section 2.3). A partir de cette notion, nous formulons le problème. Il s'agit pour nous de maximiser le débit final offert aux applications de l'utilisateur. Nous avons opté pour une approche équitable en utilisant le critère d'équité Max-Min. Bien sûr d'autres hypothèses pourraient être envisagées mais à utilisateur de classe de services identique, l'objectif retenu a été de contenter le plus grand nombre sans sacrifier des utilisateurs. Pour la formalisation, nous utilisons N problèmes d'optimisation successifs (où N est le nombre de flux) se basant sur le *Maximum Multi-Commodity Flow problem* et ayant pour but de trouver le prochain minimum parmi les flux dont le débit peut encore être augmenté. Cette modélisation nous permet d'obtenir la répartition théorique respectant nos critères des débits de chacun de nos flux. Ces valeurs serviront de point de comparaison avec les résultats de nos implantations.

Une fois le cadre étudié et le problème défini, nous construisons notre solution, Cooperative Congestion Control (CCC) (Chapitre 3). L'idée de CCC se fonde sur différents constats. Tout d'abord, l'architecture de NDN met de l'intelligence dans chaque nœud du réseau tandis que les requêtes sont faites aux extrémités. Une architecture distribuée pour la gestion des flux nous semble alors la mieux adaptée. Cette gestion passe par l'estimation des capacités attribuées à chaque flux. Classiquement dans Internet, l'action s'effectue sur les données pour faire réagir TCP. Dans NDN, nous avons une autre opportunité : les Interests. Les solutions de la littérature se basent souvent sur le nombre d'Interests en attente pour un préfixe dans chaque nœud. Nous lui avons préféré le rythme des Interests pour avoir un lien naturel avec le débit de donnée associé et ainsi être proactif face à la congestion. CCC sera donc une solution dite pace-based et sa signalisation sera légère et incorporée dans les messages de NDN (Section 3.2). Nous présentons dans ce chapitre la conception de CCC en l'illustrant sur un nœud de l'architecture (Section 3.3). Nous avons voulu que CCC soit une architecture modulaire fondée sur la coopération entre les nœuds, la supervision des files d'émission et la répartition en différents algorithmes. Nous avons choisi sciemment des algorithmes simples pour pouvoir faire une étude comparative avec l'évaluation proposée dans le Chapitre 1. Notre solution apparaît clairement supérieure en termes d'utilisation des capacités du réseau par les flux qu'en temps de convergence, et cela malgré des algorithmes simples. Cette contribution a fait l'objet d'un article [6].

Il s'avère cependant que cette première implantation ne prend pas en compte toutes les

topologies qu'un réseau NDN pourrait proposer. Comme NDN propose du multi-chemin, ces chemins peuvent très bien se rejoindre en certains points du réseau. Ce nœud possède alors plusieurs interfaces pour joindre un utilisateur. Dans le Chapitre 4, nous proposons de généraliser CCC à toute topologie. Notre démarche reste la même (Section 4.2) et introduit de nouveaux algorithmes pour rendre le traitement dans le sens des données symétrique à celui des Interests (Section 4.3). Une implantation sous ndnSIM est proposée et évaluée au travers de différentes topologies (Section 4.4). Comme auparavant, CCC montre de très bonnes performances.

Dans le dernier chapitre de cette thèse, nous nous tournons vers l'utilisation de NDN et de notre solution, CCC, dans un contexte satellitaire. Nous avions pu montrer l'apport des CDNs au satellite dans une étude préliminaire à cette thèse [7]. Si la mise en cache est un outil classique des Performances-Enhancing Proxies (PEPs) pour satellite, alors NDN pourrait être une solution pour s'affranchir de ces derniers. Cependant les spécificités du satellite conduisent souvent à un mauvais comportement contrôle de congestion proposé par TCP et ses variantes. C'est pourquoi, nous avons étudié CCC dans ce contexte. Après avoir introduit des liens satellites géostationnaire dans notre topologie, nous avons étudié leur impact sur la répartition des différents flux en les comparant aux solutions précédemment étudiées. Cette étude montre que CCC est aussi performant dans un contexte satellite que dans un contexte terrestre, offrant des performances bien supérieures à ses concurrents [8].

Publications

Article de revue

— A. Thibaud, J. Fasson, F. Arnal, D. Pradas, E. Dubois, and E. Chaput, "QoE enhancements on satellite networks through the use of caches," *International Journal of Satellite Communications and Networking*, vol. 36, no. 6, pp. 553–565, 2018

Conférences Internationales

— A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "An Analysis of NDN Congestion Control Challenges," in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 18–24, 2019

— A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Cooperative Congestion Control in NDN," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020

— A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Reactivity Enhancement of Cooperative Congestion Control for Satellite Networks," in *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 135–141, 2020

Conférences Nationales

— A. Thibaud, J. Fasson, F. Arnal, D. Pradas, E. Dubois, and E. Chaput, "Évaluation de l'impact de caches pour de la vidéo adaptative par satellite," in *Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, (Roscoff, France), May 2018

— A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Cooperative Congestion Control dans NDN," in *Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, (Lyon, France), Sept. 2020

Chapitre 1

État de l'art des architectures ICNs

Sommaire

Ce chapitre a pour objectif de présenter ce qu'est un Information Centric Network, un réseau centré sur le contenu. Nous y décrivons ses principales caractéristique et faisons un état de l'art des architectures ICN. La deuxième moitié du chapitre est, quant à elle, un état de l'art des solutions de transport dans NDN, une architecture ICN populaire.

1.1 Introduction

Les ICNs ambitionnent de remplacer la pile IP. En effet, ils ont vocation à être plus adaptés aux applications actuelles telles que le "streaming" vidéo. Comme leur nom l'indique, les ICNs sont des réseaux centrés sur l'information (aussi appelé donnée, contenu, objet, ou encore NDO pour Named Data Object). Ils sont donc différents des couches réseau comme IP qui est un host-centric network, à savoir un réseau centré sur l'emplacement des clients et serveurs. En effet, dans IP un utilisateur communique avec un serveur bien précis (identifié par son adresse IP) pour récupérer sa donnée. Dans les ICNs, l'utilisateur exprime simplement au réseau qu'il souhaite avoir une donnée spécifique et c'est le réseau qui est chargé d'aller la récupérer sur un ou plusieurs serveurs pour ensuite la délivrer aux utilisateurs intéressés. De nombreuses architectures ICNs ont été définies. Suivant le type d'architecture, l'intelligence peut être distribuée entre tous les nœuds, entre certains nœuds ou centralisée sur un seul nœud. La première fonctionnalité dont nous allons discuter est le nommage de cette donnée. C'est une problématique aussi importante que l'adressage IP, puisque celle-ci est l'équivalent du nommage des hôtes dans les réseaux IP. Puis nous expliquerons comment font ces nouvelles architectures pour assurer aux utilisateurs l'intégrité de la donnée. En effet, comme les utilisateurs s'adressent au réseau pour obtenir un contenu donné, les méthodes classiques client/serveur d'Internet ne sont pas applicables pour garantir qu'il s'agit bien du contenu demandé. Le plus naturel semble alors d'expliquer ensuite en quoi cela peut créer de nouvelles opportunités pour le *caching* et quelles sont les techniques de cache utilisées dans les ICNs. Cependant connaître le fonctionnement du relayage des messages ICNs est nécessaire pour bien saisir comment le *caching* peut fonctionner. C'est pourquoi nous verrons comment le réseau s'occupe du relayage des différents paquets ICNs puis comment il obtient les informations de routage nécessaires à ce relayage avant d'expliquer les méthodes de *caching*. Finalement, nous verrons en quoi certains types de relayage des ICNs permettent de gérer naturellement le multi-homing des utilisateurs, le multicast ou encore la mobilité.

La figure 1.1 présente un exemple d'utilisation d'une architecture ICN et illustre quelques-unes des fonctionnalités mentionnées. Notre utilisateur demande une donnée sans s'adresser à un serveur particulier. Le réseau (nous avons pris l'exemple d'une

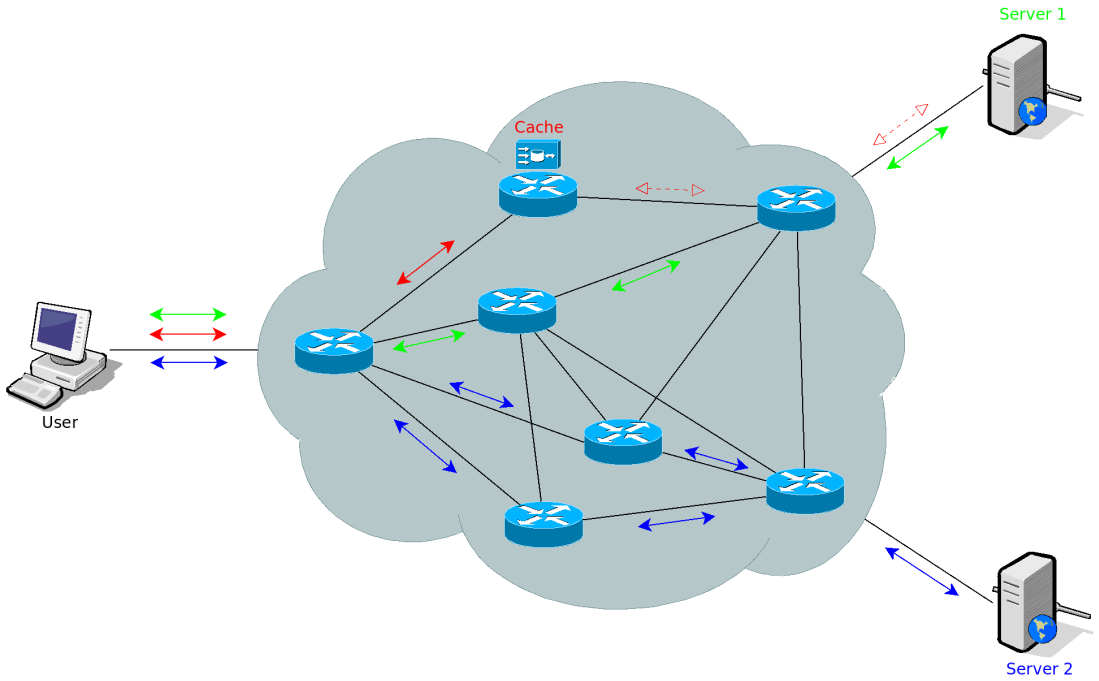


FIGURE 1.1 Vue d'ensemble

architecture où l'intelligence est distribuée sur chaque nœud et où nous n'illustrons que les décisions du premier nœud), qui sait que la donnée est disponible sur les serveurs 1 et 2, va orienter les différentes requêtes vers ces deux sources. Il va ainsi utiliser deux chemins différents pour joindre le serveur 2 (les deux chemins bleus) et deux autres chemins pour joindre le serveur 1 (les chemins rouge et vert). Nous pouvons remarquer qu'un cache sur le chemin rouge va répondre aux requêtes et donc ne pas les relayer jusqu'au serveur 1. Cette technique de cache, qui se nomment *on-path caching*, est une des techniques possibles pour les ICNs et sera décrite plus en détail et avec les autres techniques dans la partie 1.2.4.

1.2 Information Centric Networking

Van Jacobson, notamment connu pour avoir participé à l'élaboration du contrôle de congestion dans TCP, a donné une première présentation sur les ICNs en 2006 au Google Tech Talk (*"A new way to look at networking"*). Cela aboutira finalement sur le Content-Centric Networking (CCN) [11] en 2009 et Named Data Networking (NDN) [1, 12] en 2010. Les deux architectures étant très similaires (un effort de convergence est

en cours entre les deux communautés [1]), nous y ferons référence simplement sous le nom NDN.

En 2007, la National Science Fondation et British Telecom ont financé une étude qui donnera naissance à l'architecture DONA (a Data-Oriented Network Architecture) [13]. À partir de 2007, l'Union Européenne a lancé son septième programme-cadre (FP7). Le FP7 a notamment financé Publish-Subscribe Internet Routing Paradigm (PSIRP) [14] en 2008 qui évoluera en Publish-Subscribe Internet Technology (PURSUIT) en 2010 [15]. Il a aussi financé les projets 4WARD (en 2008) et SAIL (en 2010) qui créeront et feront évoluer l'architecture Network of Information (NetInf) [16]. En 2011, la NSF FIA (Future Internet Architecture) a aussi financé le projet MobilityFirst [17, 18]. Enfin, nous pouvons évoquer ici une dernière architecture, XIA pour eXpressive Internet Architecture [19]. Elle a été financée par la NSF aussi en 2011. De plus, à partir de 2014, le FP7 se termine et Horizon 2020 prend sa suite (programmes-cadre de l'Union Européenne). Il a financé notamment RIFE (architectuRe for an Internet For Everybody) [20] et POINT (iP Over IcN - the betTer ip) [21] à partir de 2015, qui sont tous deux la continuité de PSIRP et PURSUIT. L'Agence Spatiale Européenne (ESA) a aussi financé une étude appelé φsat [22] via son programme ARTES (ESA's programme of Advanced Research in Telecommunications Systems). RIFE et φsat étudient l'apport des ICNs pour les réseaux satellite et utilisent l'architecture PURSUIT.

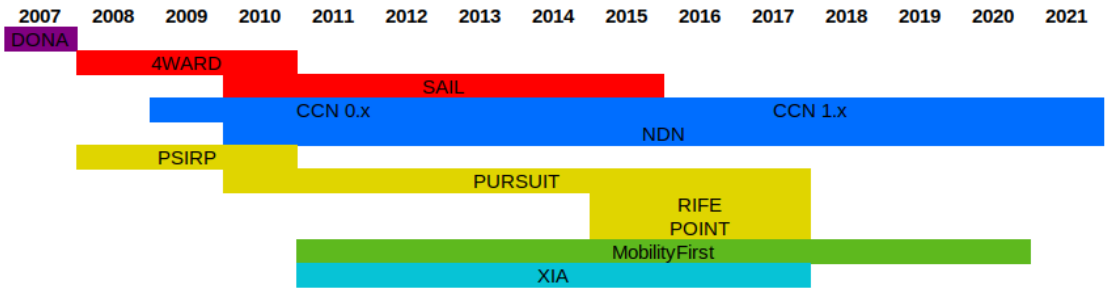


FIGURE 1.2 Frise temporelle des architectures ICN

La figure 1.2 récapitule les différents projets et architectures. Bien qu'elle soit parmi les premières architectures ICNs définies, DONA n'a jamais été implantée ni approfondie. Il reste cependant un exemple constamment cité par les autres architectures ou états de l'art sur les ICNs. L'architecture XIA se propose de ne pas s'intéresser qu'aux solutions "host-centric" ou "information-centric" mais d'être capable d'incorporer de futures idées, comme les "service-centric" par exemple. Elle est cependant l'architecture la moins mature avec notamment une interrogation sur le relayage des données. MobilityFirst,

1. `https://trac.ietf.org/trac/irtf/wiki/icnrg/convergence`

comme son nom l'indique, propose de s'occuper particulièrement de la gestion de la mobilité. L'architecture PURSUIT qui est utilisée dans les nouveaux projets RIFE et POINT continue d'évoluer et propose une gestion centralisée des décisions. À l'inverse, NDN propose quant à elle une gestion complètement décentralisée des décisions. Ce sont les deux architectures les plus matures d'ICNs à ce jour. Notons tout de même que MobilityFirst et XIA ont encore des publications en 2016 et 2017 (et jusqu'en 2020 pour MobilityFirst) mais que cela reste marginal par rapport à NDN et PURSUIT (via les projets RIFE et POINT). NetInf, de son côté, proposait de tout faire et, même si elle a reçu beaucoup d'engouement, il nous semble que c'est aujourd'hui terminé puisque ces principaux auteurs travaillent maintenant sur l'architecture NDN.

1.2.1 Nommage des objets

Comme indiqué précédemment, le nommage des objets est un point crucial de ce nouveau paradigme. Deux grandes familles de solutions se dégagent : le nommage hiérarchique et le nommage plat (et auto-certifié).
Le nommage hiérarchique permet de faire de l'agrégation de routes dans les tables de routage des nœuds ICN et favorise donc le passage à l'échelle. De plus, ces noms sont souvent compréhensibles par l'humain. Cette méthode nécessite cependant la présence d'une autorité de nommage. NDN utilise ce type de nommage.
Le nommage plat comporte, quant à lui, l'avantage d'authentifier la donnée et de ne pas nécessiter une autorité de nommage. Par exemple, NetInf utilise le *named information URI scheme* [23]. Le nom est composé du hash de la donnée et de la fonction de hashage utilisée. MobilityFirst utilise aussi un nommage plat. D'ailleurs, cette architecture ne se contente pas de nommer uniquement la donnée mais aussi les hôtes (pour les hôtes, le nom est leur clef publique, ce qui requiert donc une infrastructure à clés publiques). XIA va encore plus loin et va nommer les services aussi.
Les autres architectures, PURSUIT et DONA utilisent un peu de chaque méthode. Ils proposent tous les deux un nommage à plat composé de deux parties qui permet de faire de l'agrégation de routes tout en conservant l'authentification de la donnée et l'indépendance à une autorité de nommage. Cependant leur hiérarchisation étant limitée (seulement deux parties), l'agrégation de routes l'est elle aussi.

1.2.2 Sécurité

Un des points primordiaux des ICNs est qu'un contenu soit indépendant de qui le demande et de qui le fournit. Cela permet de pouvoir réutiliser un contenu pour plusieurs

utilisateurs et donc de procurer plus d'opportunités aux fonctions de caching. Il faut cependant assurer à un utilisateur que la donnée reçue est bien celle qu'il a demandée. C'est pourquoi l'intégrité de la donnée est un point essentiel.
Deux techniques sont classiquement utilisées et dépendent du nommage utilisé :

— Le nom de la donnée est, ou contient, son propre hash comme dans PURSUIT, NetInf, MobilityFirst et XIA. Cette technique pose problème dans le cas où le contenu est généré dynamiquement. En effet, le nom dépend donc du contenu et ne pourra être déterminé que lorsque le contenu sera connu entièrement.
— Une autre solution consiste à utiliser une infrastructure à clefs publiques et à signer la donnée. Cette méthode est donc plus contraignante mais possède des avantages sur le nommage, puisqu'une donnée peut avoir un nom arbitraire. Les différents acteurs qui ont choisi de l'utiliser sont cependant assez confiants puisqu'ils expliquent que c'est une contrainte déjà présent actuellement et qu'elle est bien gérée et non limitante. NDN et DONA utilisent cette méthode. À noter que MobilityFirst et XIA l'utilisent aussi pour les communications avec un hôte ou un service.

1.2.3 Relayage et Routage

Les ICNs redéfinissent les fonctions de routage et de relayage. Nous allons d'abord présenter les différentes techniques de relayage utilisées dans les ICNs puis nous expliquerons comment le routage est effectué.

Chaque architecture ICN possède, au moins, deux types de messages : les données et les requêtes pour ces données (aussi appelées *Interest*, *Find*, *Subscribe* ou encore *Get*). Les requêtes peuvent être faites de manière préventive, comme dans PURSUIT où les messages *Subscribe* peuvent être émis alors que la donnée n'est pas encore disponible, ou en direct, comme pour les *Interest* dans NDN. Ces requêtes sont de petites tailles alors que la donnée associée peut au contraire avoir une taille importante. C'est pourquoi la plupart des architectures n'adoptent pas le même comportement pour le relayage de la donnée et de la requête.

1.2.3.1 Relayage de la requête de donnée

Pour la requête, toutes les architectures utilisent un relayage **en fonction du nom**. Il y a d'abord la technique la plus directe : dans NDN, chaque nœud possède une table où pour chaque nom est indiqué la ou les interfaces via lesquelles la donnée est disponible.

Cette technique a l'avantage d'être simple et de permettre aux routeurs de proposer un relayage plus intelligent (puisqu'ils connaissent potentiellement plusieurs interfaces où la donnée est disponible). Elle possède cependant un problème de passage à l'échelle qui est en partie résolu par l'agrégation de routes.
DONA et PURSUIT utilisent une technique similaire. La différence est que seulement certains nœuds particuliers (*Resolution Handler* pour DONA, *Rendez-vous Node* pour PURSUIT) possèdent une table comme celle-ci. Les routeurs classiques transfèrent ainsi les requêtes au nœud particulier le plus proche. Celui-ci va ensuite les transférer au nœud particulier le plus proche de la donnée, qui va à son tour les transférer à l'entité qui publie la donnée. Le problème de passage à l'échelle se retrouve ici résolu (en dimensionnant correctement ces nœuds). C'était une nécessité puisque ces deux architectures n'ont pas la même puissance d'agrégation de route que NDN. Ils introduisent cependant des points individuels de défaillance (qui peuvent être comblés par de la redondance) et perdent l'intelligence dans le réseau qu'avait introduit NDN (en réalité, pour PURSUIT, c'est une entité centrale qui calcule la route pour la donnée donc c'est elle qui possède cette intelligence).
Une dernière technique, utilisée dans MobilityFirst, consiste à utiliser d'abord **une résolution du nom** en adresse IP puis à relayer les paquets avec IP. L'utilisation d'IP favorise la mise en place de cette solution mais finalement renie un des principes des ICNs qui est d'être indépendant de la localisation. La résolution de nom a cependant l'avantage de faciliter la gestion de la mobilité (cf 1.2.5).
Finalement, NetInf n'a pas su choisir et propose ces deux méthodes : les nœuds sont capables de router en fonction du nom mais s'ils ne connaissent pas de chemin, ils peuvent demander des pistes de routage (sous forme d'adresse IP par exemple) à leur service de résolution de nom, qui va leur permettre d'atteindre la source ou de s'en rapprocher.

1.2.3.2 Relayage de la donnée

Pour le relayage de la donnée, il y a trois principales familles de solutions :

— DONA et MobilityFirst utilisent **IP** (note : MobilityFirst relaye ses requêtes et ses données de la même manière). L'avantage, comme expliqué précédemment, est la simplicité de la mise en place.

— La deuxième famille de solutions consiste à utiliser **le chemin inverse de la requête**. NDN utilise cette technique en gardant des informations sur le passage des requêtes sur chaque routeur. L'inconvénient est le passage à l'échelle mais cette technique permet à chaque routeur de gérer complètement les flux de données qui le traversent (une requête implique une donnée). NetInf et DONA (ces derniers

prévoient plusieurs techniques pour relayer la donnée) utilisent aussi cette technique mais en inscrivant les identifiants des nœuds dans les paquets (remplissage lors de la traversée de la requête et utilisation pour le relayage de la donnée). Les routeurs sont ainsi sans état mais l'overhead dans le réseau est plus importante. Un autre inconvénient de cette technique est lorsque la topologie est dynamique et que le chemin n'est plus disponible.

— Finalement, la dernière technique est de faire du **routage par la source** et d'inscrire la route dans le paquet. Cette technique est utilisée dans PURSUIT. Comme expliqué précédemment, c'est une entité centrale, le *Topology Manager*, qui va calculer la meilleure route entre la source de la donnée et les utilisateurs qui la demandent. La route prend la forme d'un *Fowarding ID* (FID) qui correspond au ET logique de tous les ID des liens (LID). Un nœud qui reçoit un message va faire un OU logique du FID avec ses LID et si le résultat est le LID, il relayera le message sur le lien. Cette technique s'appelle Line Speed Publish/Subscribe Inter-networking (LIPSIN) [24] et utilise ce que l'on appelle des filtres de Bloom. La possibilité de faux positifs entraînent un risque de bande passante gâchée, potentiellement coûteux pour les liens satellites notamment (défaut relevé par RIFE et φsat).

XIA semble s'orienter vers du routage basé sur le nom (pour la requête comme pour la donnée). Leur architecture n'est pas encore fixée et de nouvelles recherches seraient nécessaires pour répondre à cette question.

1.2.3.3 Routage

Pour effectuer le relayage, les nœuds ont besoin d'information de routage. La plupart des architectures utilisent des messages spécifiques pour indiquer l'existence et l'emplacement d'une donnée (nommés *Publish* ou encore *Register*). Dans DONA et PURSUIT, ces messages sont propagés entre les nœuds particuliers (les autres ne se servent pas de ces informations). Par souci de passage à l'échelle, ces nœuds peuvent d'ailleurs avoir une architecture en *Multi-level Distributed Hash Table* [25, 26]. Dans MobilityFirst, ces messages sont aussi propagés dans leur GNRS (pour *Global Name Resolution Service*). NDN définit un protocole de routage à états de lien pour le routage intra-domaine, Named-data Link State Routing Protocol (NLSR) [27, 28]. Pour un nœud et un préfixe, ce protocole permet de calculer la meilleure route pour chaque interface du nœud. Le nœud a ainsi la liste des interfaces où les données correspondantes au préfixe sont disponibles. Pour le routage inter-domaine, il n'y a pas encore de solution globalement adoptée par la communauté. NetInf supporte de nombreux protocoles de routage, en particulier OSPF.

1.2.4 Caching

Dans les ICNs, les nœuds sont capables de mettre en cache des contenus pour les réutiliser lorsqu'une nouvelle requête arrive. Ceci est rendu possible par le fait qu'une donnée est intrinsèquement intègre et que son authenticité ne dépende pas du serveur avec qui l'utilisateur communique (ou de la connexion entre les deux entités).
Il existe deux techniques différentes de caching : le *on-path caching* et le *off-path caching*. Ce sont les noms qui sont utilisés dans la littérature mais, comme nous allons le voir, il faut plutôt comprendre *transparent caching* et *explicit caching*, respectivement.

— Le *off-path caching* correspond au fait que le nœud qui met en cache un contenu va l'annoncer au réseau (d'où le "explicit"). Des requêtes pourront donc lui être directement destinées. La décision de la mise en cache peut directement provenir du fournisseur de contenu. C'est ce que font déjà les CDNs aujourd'hui mais l'avantage des ICNs est que les nœuds peuvent le faire sans accord avec le fournisseur de contenu officiel dans le but d'améliorer l'utilisation du réseau. Des accords commerciaux peuvent toujours être mis en place, comme actuellement (via les plus grands fournisseurs de ce service comme Akamai ou Amazon par exemple), dans le but de promouvoir l'accès à certaines données.

— Le *on-path caching* correspond, quant à lui, au fait que des nœuds dans le réseau aient décidé de mettre en cache un contenu (car ils estiment que c'est bénéfique pour les performances du réseau) sans indiquer au réseau qu'ils servent ce contenu (d'où le "transparent"). L'intérêt, par rapport au *off-path caching*, est de ne pas produire des messages de signalisation et de pouvoir plus rapidement remplacer ce contenu par un autre contenu jugé plus populaire. De nombreux algorithmes permettent de choisir quel contenu un nœud va mettre en cache et pour combien de temps [29–31]. C'est un sujet de recherche actif et prometteur pour l'amélioration de la qualité d'expérience des utilisateurs.

Ces deux techniques de mise en cache sont donc possibles sans accord du fournisseur de contenu grâce au fait que la donnée soit intrinsèquement sécurisée (cf 1.2.2).

Toutes les architectures sont capables de faire du *off-path caching*. Pour le *on-path caching*, c'est plus compliqué car, suivant l'architecture, une requête de donnée n'est pas traitée de la même manière par tous les nœuds du réseau. Dans NDN et NetInf, tous les nœuds s'occupent de relayer les requêtes et donc, chacun peut décider s'il met en cache un contenu ou non. Pour DONA, ce sont les *Resolution Handlers* qui ont la possibilité de mettre en cache un contenu, tandis qu'à l'inverse, dans MobilityFirst, ce sont les nœuds classiques (hors GNRS) qui en sont capables. Les auteurs de PURSUIT

expliquent que, comme la donnée ne suit pas forcément le même chemin que la requête, elle ne repasse pas forcément par les nœuds spéciaux qui se sont occupés de relayer la requête et qui seront potentiellement à nouveau chargés de relayer les prochaines. Si ces nœuds-là ne reçoivent jamais la donnée, ils sont incapables de la mettre en cache. La solution préconisée est que les *Rendez-vous Nodes* de PURSUIT, au lieu de simplement, relayer la requête vont indiquer qu'ils souhaitent avoir la donnée aussi. Ils peuvent ainsi la mettre en cache et la réutiliser au besoin.

1.2.5 Mobilité

La mobilité est un problème de plus en plus présent de nos jours. Il y a, en effet, de plus en plus d'utilisateurs qui demandent une donnée depuis une position mais qui, avant que la donnée n'arrive, se sont déplacés. La donnée n'atteint donc pas l'utilisateur et celui-ci n'est pas satisfait. Les ICNs semblent très prometteurs vis-à-vis de ce problème du fait qu'ils sont indépendants de la localisation. Il faut quand même différentier deux types de mobilité : la mobilité de l'utilisateur et la mobilité du contenu.

1.2.5.1 Mobilité de l'utilisateur

Le *on-path caching* peut permettre d'améliorer la récupération d'un contenu par un utilisateur mobile. En effet, la donnée va aller vers l'emplacement où était l'utilisateur lors de la demande et va potentiellement être mise en cache par les nœuds sur le chemin. Lorsque l'utilisateur va redemander la donnée d'une localisation proche, elle sera obtenue potentiellement via un de ces nœuds et le délai de livraison sera donc raccourci. Toutes les architectures sont donc éligibles pour cette amélioration.
Des mécanismes spécifiques peuvent être mis en place suivant l'architecture. Par exemple, avec MobilityFirst, l'utilisateur peut mettre à jour sa position dans le réseau auprès des GNRS et un nœud peut redemander la position de l'utilisateur en cours de transfert. Quand la donnée arrive à l'ancienne position de l'utilisateur, le nœud peut ainsi récupérer la nouvelle position et continuer la transmission sans que l'utilisateur n'ait à redemander la donnée. PURSUIT propose de multicaster la donnée dans une zone (car ils supposent que l'utilisateur ne s'est pas déplacé très loin de sa position d'origine) ou même d'utiliser des techniques de "mobility prediction". XIA utilise, lui, des adresses structurées en DAG (*Directed Acyclic Graph*). C'est-à-dire que si un nœud ne comprend pas le premier adressage, il pourra utiliser le second qui sera plus explicite. Et surtout, l'adresse DAG peut indiquer un nœud de rendez-vous qui est en charge de savoir où se trouve l'utilisateur à tous moments.

1.2.5.2 Mobilité du contenu

Par mobilité du contenu, nous voulons dire mobilité du serveur qui propose le contenu. Une première amélioration notable est la présence des caches, aussi bien *off-path* que *on-path*. En effet, les *off-path caching* multiplient le nombre de sources potentielles et réduisent le risque que la requête soit dirigée vers la source originelle qui se serait déplacée. Le *on-path caching* permet d'atteindre la donnée avant d'arriver à la source donc, même si elle s'est déplacée, la donnée est atteinte. De plus, les mécanismes spécifiques de MobilityFirst et XIA décrit dans la partie précédente s'appliquent ici aussi.

1.2.6 Multi-homing et Multicast

Le multi-homing correspond au fait qu'un utilisateur soit connecté à plusieurs fournisseurs d'accès à internet. Dans NDN, la couche réseau sait gérer le fait d'avoir plusieurs routes (ou "next hops") vers une donnée. Les utilisateurs ont aussi une couche réseau et les différentes interfaces qu'ils possèdent (4G et Wi-Fi par exemple) sont naturellement toutes deux utilisées. Dans PURSUIT, l'entité responsable du calcul de la route de la donnée peut décider d'utiliser l'interface 4G ou Wi-Fi selon son estimation du meilleur chemin (il n'est plus borné par l'interface utilisée lors de l'envoi de la requête). Pour MobilityFirst, le système de résolution de nom peut aussi renvoyer les différentes adresses d'un utilisateur. Les autres architectures ne présentent pas de particularité avantageant la gestion de plusieurs interfaces.

Le multicast correspond à l'émission d'une donnée vers un groupe d'utilisateurs. Dans NDN, les nœuds font de l'agrégation de requête lorsque plusieurs utilisateurs demandent la même donnée au même moment. Lorsque la donnée arrive sur le nœud qui a fait l'agrégation, il la multicaste sur toutes les interfaces qui l'ont demandée.
Le routage dans PURSUIT permet naturellement de faire du multicast. Le FID correspond, en effet, au ET logique de tous les liens entre la source et les utilisateurs qui ont demandé la donnée. Le nœud où les chemins entre les différents utilisateurs se séparent va relayer la donnée sur les différents liens car ils vont individuellement satisfaire le filtre de Bloom.

1.3 Named Data Networking

NDN [1, 12] est l'architecture qui concentre le plus de travaux de recherche. Même s'il se veut général, l'ICNRG [32], le groupe de recherche de l'IRTF sur les ICNs se préoccupe principalement de NDN. C'est pourquoi, nous considérons que c'est l'architecture la plus mature et avons concentré nos efforts de recherche dessus. Nous allons donc décrire un peu

plus en détail le fonctionnement de cette architecture, pour une meilleure compréhension de la suite du document. Les utilisateurs d'un réseau NDN s'appellent les consommateurs et les fournisseurs de contenu s'appellent les producteurs (l'équivalent des clients-serveurs dans Internet).

1.3.1 Nommage

Comme expliqué dans la partie 1.2.1, le nommage dans NDN est un nommage hiérarchique. Concrètement, le nom des paquets est divisé en trois segments : un pour le routage, un pour l'application et un pour le protocole. Le segment dédié au protocole permet à l'utilisateur de demander une version particulière de la donnée et un "chunk" (segment) particulier. Les paquets dans NDN ont en effet une taille maximale de 8800 octets. Lorsqu'un contenu est de taille supérieure, il est obligé qu'il soit segmenté en plusieurs paquets Data. Le segment dédié à l'application permet à un serveur de proposer différentes données sous le même préfixe (qui sera donc le segment pour le routage). La Figure 1.3 donne trois exemples de noms de paquets Data pour trois contenus différents. La partie en rouge représente la partie pour le routage, la partie verte la partie pour

/paris/plan/metro/ligne1/v=3/c=24
/paris/plan/bus/ligne6/v=1/c=2
/toulouse/plan/metro/lignea/v=1/c=1

FIGURE 1.3 Exemple de noms dans NDN

l'application et la partie bleu pour le protocole. Les deux premiers noms sont donc fournis par la même application mais ne représente pas le même contenu. Les nœuds dans le réseau ne différencient pas a priori ces différentes parties du nom. Ils utilisent la technique de "Longest Prefix Match" (LPM) comme dans IP [33] et choisissent simplement l'entrée qui correspond le mieux (avec le préfixe le plus long).

1.3.2 Format et Traitement des messages

Dans NDN, une requête pour une donnée est appelée Interest et la donnée associée Data. Une propriété essentielle des réseaux NDN est que l'émission d'un Interest par un nœud induit au maximum l'émission d'une Data par ce même nœud. Cette propriété s'appelle le *flow balance*. Les paquets NDN utilisent l'encodage Type-Length-Value [34] (TLV), ce qui leur permet d'avoir de nombreux champs optionnels. Pour un Interest, le champ "Name" est obligatoire. Le champ "Nonce" est lui aussi obligatoire si le paquet est

transmis sur un lien du réseau. Il est utilisé pour détecter si l'Interest fait des boucles dans le réseau. Pour une Data, les champs "Name", "Content" et "Signature" sont obligatoires. Ces deux types de messages sont illustrés dans la figure suivante [1] :

(a) Paquet Interest (b) Paquet Data

FIGURE 1.4 Format des paquets dans NDN

Une Data est donc un segment de la donnée applicative et peut, elle-même, être fragmentée par la couche inférieure si la MTU n'est pas suffisamment élevée. Dans la partie 1.2.3, nous expliquions que chaque nœud NDN doit être capable de faire suivre un Interest vers les consommateurs comme de mémoriser les Interests en attente de Data. Pour se faire, le nœud possède deux tables distinctes :

— La Forwarding Information Base (FIB) : cette table contient, pour chaque préfixe, une liste d'interface où la donnée est disponible. Elle permet donc de savoir où relayer les Interests arrivant sur le nœud.

— La Pending Interest Table (PIT) : c'est dans cette table que le nœud va inscrire qu'une ou plusieurs demandes ont été faites pour une donnée (avec la provenance de ces demandes) et qui n'ont pas encore reçu la Data en retour.

De plus, le nœud possède aussi un Content Store (CS) qui est capable de stocker une donnée. Il peut ainsi la resservir si elle est demandée à nouveau (cf 1.2.4).
La figure 1.5 résume parfaitement le traitement d'un Interest et d'une Data par un nœud NDN.

Lorsqu'un Interest arrive, le CS est consulté en premier. Si la Data est disponible, elle est directement émise. Sinon, la PIT est consultée pour savoir si une requête similaire n'a pas déjà été faite récemment. Si c'est le cas, il y a déjà une entrée (appelé *Pending Interest*) dans la PIT pour cet Interest et seule l'interface d'où provient le nouvel Interest est ajoutée (l'Interest n'est pas réémis car il l'a déjà été lors de la création de l'entrée). Ce processus s'appelle l'agrégation d'Interests et va permettre au nœud de faire du multicast. S'il n'existe pas déjà une entrée pour cet Interest, celle-ci est créée et la FIB est consultée pour savoir où le contenu est disponible et quelles interfaces peuvent être utilisées. La FIB donne une liste d'interfaces possibles, c'est le rôle de la *forwarding strategy* (FS) de

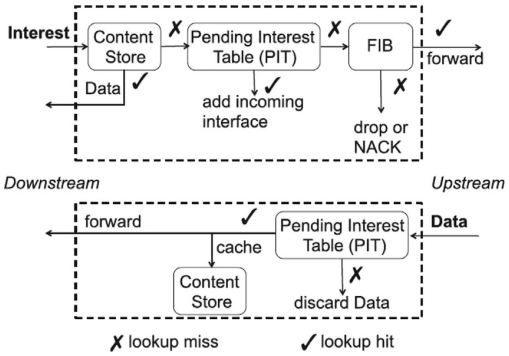


FIGURE 1.5 Processus de relayage sur un nœud NDN [1]

décider laquelle le nœud va utiliser pour cet Interest. Le design de la *forwarding strategy* n'est pas fixé dans NDN et de nombreuses contributions proposent la leur. Nous verrons dans la section 1.4.1.2 quelques-unes d'entre elles. Lorsqu'une Data arrive, c'est la PIT qui est consultée et toutes les interfaces où est arrivé un Interest pour ce paquet sont utilisées pour retransmettre la Data. C'est grâce à ce mécanisme que le multicast prend forme dans NDN (cf 1.2.6). Lorsque le nœud a relayé la Data sur toutes ces interfaces, il peut supprimer le *Pending Interest* associé dans la PIT. En fonction de la politique de cache, la Data peut aussi être stockée dans le CS.

1.3.3 Comparaison entre NDN et les architectures actuelles

1.3.3.1 Introduction

Le réseau IP est fondé sur la communication entre deux machines à travers une interconnexion de réseaux, tandis que le réseau NDN permet à un utilisateur de récupérer un contenu. Cette différence de point de vue met en valeur certains types de communication au détriment d'autres. Contrairement à NDN, le réseau IP est efficace pour mettre en relation deux hôtes. Malgré leur prédisposition pour tel ou tel type de services ou applications, NDN et IP ont tous deux essayé de se doter de moyen leur permettant de fournir tout type de service. Dans Internet, les solutions sont construites au-dessus d'IP : il existe plusieurs architectures en overlay qui permettent de faire de la livraison de contenu d'une manière plus efficace que nativement. Dans cette section, nous allons donc comparer NDN avec ces architectures en overlay mais aussi explorer les modifications nécessaires à NDN pour offrir des services de communications plus performants. Notre but ici est de mettre en valeur les avantages de l'utilisation de NDN pour les services de livraison de contenu mais sans occulter les problèmes posés par les autres services. Nous présenterons les principales solutions qui ont été proposées pour pallier ces difficultés.

1.3.3.2 Communications hôte à hôte

Contrairement au réseau IP, NDN ne favorise pas naturellement les services où un utilisateur souhaite communiquer avec un autre utilisateur. Plusieurs mécanismes et applications ont cependant été conçus pour combler ce manque.
La première dont nous allons parler s'appelle NDN-RTC (pour Real-time videoconferencing over Named Data Networking) [35]. Cette application permet à ses utilisateurs de faire de la vidéo-conférence en temps réel. Pour cela, chaque utilisateur sera producteur et consommateur. La partie producteur de l'application sera en charge de récupérer, encoder et nommer la donnée (vidéo et audio). Elle proposera plusieurs encodages pour que les utilisateurs choisissent celui convenant le mieux à leur bande passante. La partie consommateur sera, quant à elle en charge du choix de l'encodage utilisé (en tenant compte des mesures sur les conditions du réseau).
Une autre solution consiste à utiliser le protocole ChronoSync [36] qui permet de synchroniser des données sur NDN. Il est notamment utilisé pour les applications ChronoChat (qui est une application de discussion à plusieurs) ou ChronoShare (qui est une application de partage de fichier en pair-à-pair) et peut être utilisé pour de futures applications comme un équivalent de Dropbox ou de Google Docs. Le principe est assez simple, une première personne va créer le préfixe du partage et l'annoncer au réseau (toutes les données du partage auront donc ce préfixe). Les autres participants vont apprendre ce préfixe d'une manière indépendante et eux aussi l'annoncer au réseau. Les utilisateurs de ChnonoSync vont calculer un hash de l'état du partage (ce qu'ils possèdent) et l'appelle *state digest*. Ils vont ensuite envoyer des *Interests* le contenant qui seront multicastés entre tous les participants. Il faut donc modifier l'architecture de NDN pour que ces deux propriétés soient ajoutées. Pour le multicast d'*Interest*, c'est déjà possible mais il faut un moyen pour que les nœuds sachent s'il faut le faire ou pas (un champ type par exemple). Pour l'inclusion de méta-données dans les *Interests*, les mêmes auteurs proposent des solutions dans un autre papier (voir 1.3.3.3). Ces *Interests* permettent d'avoir une sorte de connexion entre les participants. Quand un des participants remarque qu'il est en avance sur l'état du partage (son digest est différent de celui qu'il reçoit), il envoie une réponse aux *Interests* (qui sera donc multicastée aux participants) contenant le nom de la nouvelle donnée (voire la donnée elle-même). Quand ils reçoivent le nom de la nouvelle donnée, les autres participants la demandent, leurs *Interests* seront agrégés par les nœuds et elle sera ainsi propagé en multicast. Finalement, le protocole ChronoSync est une première solution pour ne plus être bloqué par le comportement "pull-based" de NDN.

1.3.3.3 NDN et HTTP

Le principe des ICNs est largement inspiré par le protocole HTTP qui domine aujourd'hui Internet. En effet, NDN peut être, de manière schématique, décrit comme du HTTP au niveau réseau. Les *Interest* de NDN sont assimilables à des requêtes HTTP GET et les URL sont très proches des noms des données (avec leur structure hiérarchique notamment). Cependant le protocole HTTP ne propose pas que la méthode GET, il y a notamment les méthodes POST et PUT qui permettent d'envoyer des données au serveur. Les requêtes HTTP GET peuvent aussi contenir des métadonnées, alors que les Interests ne le peuvent pas. NDN est ce qu'on appelle "pull-based", c'est-à-dire que les données sont récupérées lorsqu'elles sont explicitement demandées. Tel quelle, l'architecture ne permet pas aux utilisateurs d'envoyer des données à un serveur. Il faudrait que celui-ci les demande explicitement sauf qu'il n'a aucun moyen de savoir que l'utilisateur veut les lui envoyer.
Même s'il ne propose pas nativement des communications push-based, NDN propose d'autres avantages. Comme nous l'avons expliqué dans la section 1.2.6, NDN profite d'un système de multicast qui n'est pas disponible en utilisant le protocole HTTP. La mise en cache des données est directement disponible dans la couche réseau NDN alors que pour HTTP il faut utiliser des CDNs (cf 1.3.3.6). Une autre différence notable est que le contrôle de congestion et la fiabilisation des données sont effectués par TCP pour HTTP et est donc de bout en bout, tandis que pour NDN, ils peuvent être faits de bout en bout mais aussi au saut par saut (cf 1.4.1 et 1.4.2). Il faut noter que les *Interest* ne peuvent récupérer des *Data* de taille 8800 octets (entête inclus) alors que les HTTP GET peuvent récupérer des fichiers d'une taille non limitée. Cela vient notamment du fait que le protocole HTTP est un protocole de couche applicative tandis que NDN est un protocole de couche réseau. Les auteurs de [37] explorent plusieurs solutions pour rendre un service RESTful sur NDN. C'est-à-dire qu'un utilisateur de NDN soit capable de récupérer une donnée mais aussi d'envoyer, de mettre à jour ou de supprimer une donnée sur un serveur (s'il en a les droits bien évidemment). L'utilisateur doit même être capable d'envoyer des métadonnées lors d'une récupération de contenu. Les premières solutions envisagées sont d'inclure les métadonnées (pour de la récupération de contenu, pour indiquer la volonté d'envoyer, de mettre à jour ou supprimer une donnée) directement dans le nom de la donnée (nous pourrions comparer cela aux "query string" des URL). Ces solutions posent de nombreux problèmes (taille des PIT qui augmentent, le nom complet est reporté dans la *Data* donc moins de place pour la vraie donnée, ...). Le deuxième type de solution est d'ajouter des champs dans les *Interests*. Le champ *Application Data Field* permet à un utilisateur de mettre des données dans ses *Interests* tandis que le champ

DataLocator Field lui permet d'indiquer sa volonté d'envoyer une donnée et le moyen d'y accèder. Le premier champ est utile quand la donnée à envoyer est petite. En effet, dans les *Forwarding Strategies*, une hypothèse forte est que la taille des *Interests* est très inférieure à celles des *Data*. Donc utiliser ce champ pour envoyer des données importantes aurait de mauvaises conséquences sur ces algorithmes et les performances globales du système. Aucune taille limite n'est définie dans le papier mais le problème est mentionné. Le deuxième champ est de toute façon nécessaire pour les données plus grandes que la taille d'une *Data*. Un autre problème se pose pour ces deux solutions : comment est-ce qu'un nœud va faire de l'agrégation d'*Interests* ? En effet, si deux *Interests* arrivent sur un nœud mais n'ont pas les mêmes champs et/ou valeurs dans ces champs, comment le nœud choisi lesquels il conserve et lesquels il abandonne ? Il pourrait potentiellement les agréger aussi mais cela ne peut être fait à l'infini puisque la taille d'un *Interest* est limitée.

1.3.3.4 NDN et DTN

Les DTN (pour Disruption/Delay Tolerant Networking) sont un type de réseau qui s'accommode bien si un lien a un fort délai (satellite géostationnaire voire pire) ou même si la connexion est fréquemment rompue. Les ICNs ont aussi pour ambition de répondre à cette problématique. Cependant, leur architecture, telle qu'elle est, n'est pas capable de prendre en charge un lien où la connexion est régulièrement interrompue. NDN, par exemple, va envoyer des NACK quand un *Interest* arrive sur le nœud au moment où le lien vers la donnée est rompue. Des propositions ont été faites pour combiner ICN et DTN [38, 39], et la solution la plus prometteuse reste de garder la couche ICN intacte et de la reposer sur une couche DTN [40]. Pour la couche ICN, le lien "disruption tolerant" serait considéré constamment actif, et c'est la couche DTN qui se chargerait de stocker la donnée le temps que le lien soit véritablement actif.

1.3.3.5 NDN et DASH

Dynamic Adaptive Streaming over HTTP (DASH) est un protocole de streaming adaptatif [41]. La vidéo est disponible sous plusieurs qualités et en fonction du débit mesuré, le client va demander la qualité qui va le mieux. Ce mécanisme adaptatif n'est pas implanté nativement dans NDN mais le reste du fonctionnement est très proche. Les points communs sont les suivants :

— La donnée (vidéo) est découpée en "chunk".
— Récupération de la donnée basée sur la demande ("pull-based communication").

— Utilisation de manifeste pour décrire la donnée.

Il est alors très facile de définir un protocole qui serait en charge de la partie adaptative de DASH pour NDN. DASC (pour Dynamic Adaptive Streaming over CCN) [42] a déjà été développé et testé sur CCN. Les applications utilisant ce protocole vont en plus bénéficier des avantages de NDN (multihoming, caching, ...). Il faut cependant noter que le contrôle de congestion ne va pas se faire au même niveau. Avec DASH, c'est TCP qui s'en occupe et qui le réalise "chunk" par "chunk" tandis qu'avec NDN, le contrôle de congestion est réalisé sur l'ensemble de la donnée applicative. Il y a d'ailleurs une limite à la taille du chunk (d'environ 8800 octets) qui va peut-être obliger DASC à avoir des chunks d'une durée maximale (en fonction de la qualité maximale proposée). Un autre problème qui peut arriver aussi avec NDN (mais qui ne lui est pas propre, c'est déjà le cas lors de l'utilisation de CDN, cf. notre papier [7]) est que l'utilisation de cache va potentiellement fausser la mesure du débit à un moment donnée et entraîner des oscillations importantes du délai de récupération des "chunks" voire des gels d'image.

1.3.3.6 NDN et CDN

Les *Content Delivery Networks* consistent à dupliquer une donnée dans de nombreux endroits du réseau pour en assurer la disponibilité. Un producteur de contenu qui souhaite que ses données soient rapidement disponibles pour ses utilisateurs va utiliser des CDNs pour que ses données soient dupliquées en bordure du réseau où elles seront plus proches des utilisateurs. Le CDN se charge ensuite de diriger les requêtes des utilisateurs vers le nœud du réseau le plus proche de l'utilisateur pour réduire le délai et augmenter le débit. Une des méthodes pour diriger les demandes des utilisateurs consiste à personnaliser la réponse du serveur DNS pour indiquer les serveurs voisins. Les différents fournisseurs de services CDNs ne sont pas forcément compatibles technologiquement (malgré les efforts de la CDNi initiative [43]) et interopérables. Cela demande des accords spécifiques entre les deux fournisseurs de service, qui ne sont souvent pas d'égaux à égaux. Pour le NDN, le "caching" est inclus dans le design. Les fournisseurs de services CDNs n'ont qu'à annoncer la donnée et le réseau va s'occuper de rediriger les utilisateurs vers leur nœud s'il estime qu'il est effectivement celui qui fournira le meilleur service. De plus, les opérateurs ne peuvent pas facilement mettre en place leur CDN dans les réseaux IP. En effet, si la donnée est récupérée de manière sécurisée (HTTPS par exemple), l'opérateur ne va pas être capable de mettre en cache la donnée s'il n'a pas au préalable établi un accord avec le producteur de la donnée. Dans NDN, la donnée est intrinsèquement sécurisée et les opérateurs peuvent les mettre en cache sans avoir à demander des clefs privées/publiques aux nombreux fournisseurs de contenu (ce qui poserait en plus des problèmes de sécurité,

un problème de passage à l'échelle important). En plus de tout cela, chaque nœud NDN possède un *Content Store* qui lui permet d'agir comme un nœud CDN transparent. Une première étude de comparaison entre CDN et NDN est faite dans [44]. Même si l'étude est conduite dans des cas spécifiques, les auteurs soulignent cependant que l'overhead de NDN est plus importante que celle de CDN. Ils concluent finalement que NDN est plus performant et simplifie énormément la mise en place de CDN.

1.3.3.7 NDN et P2P

Les ambitions de NDN et des réseaux pair-à-pair (P2P) sont très similaires. Tous deux se proposent de partager du contenu sur leur réseau. Le P2P est cependant un réseau en overlay sur IP et doit s'accommoder des communications "host-centric" pour échanger ses contenus. Les auteurs de [45] présentent le protocole BitTorrent, les points d'architecture d'NDN facilitant l'approche P2P par rapport à IP. Leur protocole nTorrent est une adaptation de BitTorrent à NDN.
Le premier problème pour un utilisateur d'un réseau P2P est de découvrir les autres utilisateurs du réseau et détenteurs de la donnée recherchée. Dans BitTorrent, cette découverte se fait à l'aide d'un "tracker" centralisé, d'une DHT (Distributed Hash Table) distribuée entre les "peers" ou du protocole PEX (Peer EXchange). Chacune de ces méthodes nécessite cependant une information extérieure (l'adresse du "tracker" ou d'un premier "peer"). Dans NDN, un "peer" ne découvre pas les autres "peers". C'est en effet le réseau qui achemine la demande au "peer" le plus proche ou disponible (suivant la métrique utilisée par la *Forwarding Strategy*). Il n'y a donc, ici, pas besoin d'information d'initialisation pour récupérer une donnée (hormis son nom bien entendu).
Il faut ensuite sélectionner le meilleur "peer" capable de fournir le contenu. Dans BitTorrent, l'utilisateur n'a pas d'information sur la condition du réseau. Il est donc obligé de tester les "peers" et d'évaluer la bande passante disponible. Une fois cela fait, il va choisir les "peers" qui disposent de la meilleure bande passante. Dans NDN, c'est la *Forwarding Strategy* qui a ce rôle. Aucun mécanisme particulier n'est nécessaire en plus pour la sélection des "peers".
Comme la donnée est proposée par de nombreux utilisateurs, il faut un moyen pour qu'un nouvel utilisateur puisse vérifier que la donnée est intègre (c'est-à-dire que c'est bien la donnée qu'il a demandée et qu'elle n'a pas été modifiée par un utilisateur malveillant). Dans BitTorrent, le fichier .torrent qui décrit les différents fichiers du torrent et leur découpage fourni aussi le hash de chacune des pièces du torrent. L'utilisateur peut ainsi facilement vérifier que la pièce reçue est bien celle produite par le serveur d'origine. Dans NDN, l'intégrité des données est garantie par son design.

Les deux derniers problèmes des réseaux P2P sont liés. Le premier est la raréfaction d'une partie d'une donnée. En effet, si dans l'ensemble du réseau P2P, la donnée est complète, la pièce 5 est par exemple disponible sur 50% des "peers" tandis que la pièce 6 ne l'est que sur 10%. Le protocole BitTorrent met en place un mécanisme pour que les "peers" demandent en priorité les pièces qui sont peu disponibles dans le réseau pour augmenter la disponibilité du contenu. Le deuxième problème est le fait qu'un "peer" peut à tout moment quitter le réseau P2P (potentiellement avec une ou plusieurs pièces rares et ainsi contribuer encore plus à la raréfaction de la donnée). Le protocole BitTorrent met en place un mécanisme de "prêter pour rendu" qui favorise les "peers" qui contribuent à la disponibilité des contenus et qui ne profitent pas seulement du réseau P2P. Dans NDN, ces deux problèmes ne sont pas réellement résolus. Aucun mécanisme dans le design ne permet cela. Cependant, le *in-path caching* va favoriser la disponibilité des contenus (mais sans vraiment la garantir).
Les auteurs présentent donc nTorrent, une application P2P sur NDN. Ils y définissent la structure du fichier .torrent. Ce fichier contient les noms des différents fichiers qui constituent le torrent. Chacun de ces noms permet de récupérer les manifestes des fichiers correspondants. Ces manifestes décrivent, à leur tour, le découpage des fichiers en pièces (ou paquets NDN ici) avec ce qu'ils appellent le "full name" (le nom du paquet auquel ils rajoutent le hash de celui-ci et qui permettrait une vérification de l'intégrité de la donnée plus rapide pour les routeurs). L'utilisation du "full name" est très similaire au protocole BitTorrent qui lui aussi inclus le hash de ses pièces dans son fichier .torrent. Comme expliquer précédemment, ce n'est pas essentiel ici puisque NDN garanti déjà l'intégrité de ses paquets. Les auteurs soutiennent que Le *in-path caching* contribue à réduire le problème de raréfaction du contenu. Dans nTorrent, les utilisateurs vont demander les morceaux de fichier de manière séquentielle. Les auteurs expliquent que cela va favoriser le *in-path caching* dans le cas de téléchargements simultanés et donc réduire la raréfaction du contenu. Même si l'impact sur le "caching" n'est pas évident, cette récupération séquentielle a l'avantage de faciliter la proposition du contenu par les "peers". En effet, le P2P prévoit aussi que les "peers" puissent fournir le contenu qu'ils ont déjà aux autres utilisateurs. Dans NDN, il n'y a ni "tracker", ni DHT, ni PEX et les utilisateurs doivent directement s'annoncer au réseau pour pouvoir recevoir les requêtes des autres "peers". Seulement par souci de passage à l'échelle des routeurs, ils ne peuvent pas avoir une entrée FIB pour chaque morceau de chaque fichier. Les "peers" ne vont donc annoncer que les fichiers voire le torrent dans sa totalité. Plusieurs solutions sont possibles ici : s'annoncer après avoir reçu la totalité du fichier/torrent, avant de l'avoir reçu et après en avoir reçu un pourcentage suffisant. Plus tôt un "peer" s'annonce,

plus tôt il participe au réseau P2P, seulement les autres "peers" vont potentiellement lui demander des morceaux qu'il ne possède pas encore. Il faut donc faire un compromis et la récupération séquentielle des données permet aux nouveaux utilisateurs de forcément trouver la donnée.

1.3.3.8 Conclusion

Dans cette section, nous avons comparé NDN aux architectures du monde IP pour la livraison de contenu, comme le P2P et les CDN. Afin d'élargir la comparaison, nous avons aussi considéré d'autres types d'applications comme de la vidéo-conférence, de l'upload de données vers un serveur ou encore de la synchronisation de données vers un groupe d'utilisateurs. Même si NDN est conçu pour faire de la livraison de contenu et ne propose pas nativement des mécanismes facilitant d'autres services, des solutions sont envisagées pour combler ce manque. Nous avons pu constater dans cette section, que suivant le service recherché, les Interests ne doivent pas être traités de la même manière par les nœuds du réseau. Par exemple, pour un service de synchronisation de contenu, les Interests ont besoin d'être multicastés vers l'ensemble des participants alors que pour la récupération de contenu classique, ils ont simplement besoin d'atteindre une seule source. Les nœuds NDN ont donc besoin de pouvoir différentier à quel service correspondent les Interests qu'ils transmettent pour y appliquer les algorithmes de relayage qui conviennent le mieux.
Les performances des applications décrites dans cette section sont d'ailleurs très dépendantes des mécanismes de transport et de gestion de QoS qui sont proposés dans les réseaux NDN aujourd'hui (que ce soit sur les nœuds extrémités, utilisateurs et fournisseurs, ou sur les nœuds sur le chemin). La section suivante propose donc un état de l'art de ces mécanismes. Nous y expliquerons notamment comment est fait le contrôle de congestion, la gestion des pertes ou plus généralement comment un nœud dans le réseau choisit l'interface où il va transmettre l'Interest qu'il est en train de traiter (ce type d'algorithme est appelé *Forwarding Strategy*). À nouveau, suivant le service recherché, les nœuds vont utiliser telle ou telle *Forwarding Strategy* pour atteindre les objectifs spécifiques du service.

1.4 État de l'art des solutions de transport dans NDN

Comme expliqué dans la partie précédente, la motivation des ICNs est d'utiliser le réseau d'une manière plus adaptée aux besoins et aux demandes des utilisateurs pour proposer de meilleurs services. Cela peut aussi permettre d'optimiser les coûts d'utilisation

du réseau (notamment parce qu'une donnée peut être intelligemment stockée dans le réseau et être réutilisée plus tard).
Il y a cependant des problématiques de transport et de QoS qui ne sont pas résolues par l'architecture de NDN telle qu'elle est décrite en partie 1.3. Le premier problème important que nous allons étudier est la congestion. Elle peut réduire drastiquement les performances d'un réseau si elle est mal gérée. Dans Internet, c'est généralement le protocole TCP qui s'en charge et il adopte une approche de bout en bout. Le problème de la congestion se trouve pourtant au goulot d'étranglement du réseau. Mais comme les routeurs IP ne sont pas capables et n'ont pas vocation à différentier les flux, le contrôle de congestion dans TCP se fait de bout en bout. Dans NDN, nous avons l'opportunité de gérer la congestion au saut par saut et donc directement là où le problème se produit.
La gestion des pertes est aussi une des problématiques à réévaluer dans le contexte de NDN. Là encore, c'est le protocole TCP qui s'en charge dans les réseaux IP et, pour les mêmes raisons, il adopte à nouveau une gestion bout en bout. Comme pour le contrôle de congestion, dans NDN, gérer ce problème sur chaque nœud est une option envisageable.
Le contrôle de flux est aussi une des prérogatives de TCP dans les réseaux IP. Dans NDN, il faut commencer par redéfinir ce qu'est le contrôle de flux. En effet, un utilisateur ne communique plus avec un émetteur précis mais avec le réseau. Les sources des données qu'il reçoit peuvent être multiples.
La gestion de la QoS représente aussi le fait de pouvoir garantir un service aux utilisateurs. Certain flux sont très sensibles aux forts délais et peu à une faible bande passante, tandis que pour d'autres, c'est l'inverse. Si les nœuds dans NDN sont capables de les différentier, ils peuvent appliquer une politique sur mesure pour chacun d'eux et les favoriser par rapport aux paramètres dont ils sont sensibles.
Dans cette partie, nous allons présenter les principales solutions existantes concernant la gestion de la QoS au sens large puis nous essayerons de faire un récapitulatif et de trouver les axes dans lesquels la recherche n'a pas encore apporté de réponse satisfaisante.

1.4.1 Contrôle de congestion

La congestion du réseau est un point très important à contrôler pour assurer de bons services et une satisfaction des utilisateurs. Dans les réseaux IP, c'est TCP qui s'en occupe depuis de nombreuses années. Il adopte une gestion de la congestion de bout en bout et contrôlée par l'émetteur des données. Deux grandes familles de solutions ont été proposées pour NDN : la famille qui gère la congestion de bout en bout et la famille qui gère la congestion au saut par saut. En effet, NDN apporte de l'intelligence dans le réseau et de nouvelles méthodes peuvent permettre de prévenir la congestion sur chaque

nœud. Ces deux familles peuvent être complémentaires et les solutions qui en découlent sont appelées des solutions hybrides.

1.4.1.1 Bout en bout

Le mécanisme de contrôle de congestion de TCP étant celui qui est le plus utilisé actuellement, toute nouvelle solution doit se comparer aux performances de TCP. C'est pourquoi les premières solutions de gestion de la congestion dans NDN ne sont qu'une adaptation de celui-ci. Cependant et contrairement aux réseaux IP, NDN est ce que nous appelons *receiver-driven*. C'est-à-dire que c'est le demandeur (ou le récepteur) qui est en charge de diriger la communication. Dans TCP, c'est l'émetteur qui gère le contrôle de congestion via sa fenêtre de congestion et les acquittements que le récepteur envoie.
Comme il n'y a pas d'acquittement dans NDN, les solutions "TCP-like" tels que ICP [46] (Interest Control Protocol) ou ICTP (Information-Centric Transport Protocol) [47] utilisent les *Interests* dans le rôle des données et les *Data* dans le rôle des acquittements. La fenêtre de congestion est ainsi gérée, non pas par l'émetteur, mais par le demandeur de la donnée. C'est absolument nécessaire, car un des principes des ICNs est que les données peuvent provenir de plusieurs sources, voire de caches dans le réseau. La multiplicité des émetteurs compromet donc une gestion *sender-driven* comme dans TCP.
Les auteurs d'ICTP [47] déclarent que leur protocole implante les mêmes algorithmes que TCP et ses évolutions (à savoir : *slow start*, *congestion avoidance*, *fast retransmit* et *fast recovery*). Ils n'expliquent pas vraiment comment et si pour certains de ces algorithmes, l'adaptation semble naturelle, pour le *fast retransmit* ce n'est pas évident. En effet, le *fast retransmit* dans TCP correspond au fait que l'émetteur va réémettre une donnée s'il reçoit trois acquittements dupliqués sans atteindre la fin du time-out. Dans NDN, les données correspondraient à un SACK (Selective ACKnowledgment) de TCP puisque chaque donnée peut avoir emprunté un chemin différent, plus court ou plus lent que les autres, et peut de plus provenir d'une source différente. En effet, l'arrivée des données 5, 6 et 7 ne signifie en aucun cas que la donnée 4 a été perdue. Le *fast retransmit* de TCP ne peut donc pas s'adapter naturellement à NDN. ICP n'implante ni le *slow start* (les raisons ne sont pas évidentes) ni le *fast retransmit*. Le cœur de ces deux protocoles restent alors l'utilisation d'une fenêtre d'émission gérée par l'équivalent du *congestion avoidance* en AIMD.
En plus du contrôle de congestion, ces deux algorithmes visent à rendre les communications :

— Fiable : ils sont en charge de renvoyer des *Interests* lorsqu'ils détectent une perte.
— Efficace : ils augmentent le débit d'émission tout en évitant la congestion.

— Équitable : comme TCP, lorsque plusieurs flux sont en compétition, ils vont se partager équitablement la bande passante.

ICP a été largement adopté par la communauté et, comme TCP pour IP, est utilisé dans la majeure partie des communications.
Cela n'a pas empêché les auteurs d'ICP de continuer leurs recherches dans ce domaine. Dans [48], ils présentent un nouvel algorithme de contrôle de congestion de bout en bout, que nous appellerons ici MPICP (pour Multi-Path Interest Control Protocol). MPICP utilise aussi une fenêtre de congestion basée sur un mécanisme AIMD. Cependant, ce protocole va prendre en compte les délais des différentes routes d'où proviennent les données. De par son design, NDN ne permet pas aux utilisateurs (ni aux nœuds du réseau) de connaître l'identité de la source ou les nœuds parcourus sur le chemin. C'est pourquoi MPICP va modifier ce design, ajouter un champ "route label" dans les paquets *Data* et permettre aux nœuds sur le chemin d'inscrire leur identifiant à la suite dans ce champ (leur identifiant pouvant être leur adresse MAC). Ainsi lorsqu'ils récupèrent des *Datas*, les utilisateurs peuvent les classer en fonction de leur route label et ainsi mesurer les délais des différents chemins. À la réception de chaque donnée, le délai moyen de la route associée sera mis à jour. De plus, le consommateur va calculer une probabilité de réduire la fenêtre de congestion par route. MPICP permet donc au consommateur de connaître le délai de chacune des routes empruntées et définit une probabilité de réduire la fenêtre de congestion pour chacune de ces routes. Plus le délai va augmenter et plus la fenêtre de congestion aura de chance de diminuer. Cependant, l'utilisateur ne choisit pas quelle sera la route utilisée et bien qu'il sache qu'une route soit congestionnée, sa seule action est de réduire le débit d'émission global des *Interests*. Toutefois, les auteurs de MPICP ne définissent pas seulement ce protocole dans ce papier. Ils définissent aussi une *Forwarding Strategy*, décrite plus en détail dans la partie 1.4.4, qui va agir directement dans les nœuds du réseau.
Une solution similaire est définie dans [49]. Path-specified Transport Protocol (ou PTP) utilise aussi un champ "route label". Une première partie de découverte du réseau requiert que le consommateur envoie ses Interests sans ce champ. Ce sont ensuite les nœuds sur le réseau qui le créent en inscrivant leur identifiant à la suite lors du passage des paquets Data (de la même manière que pour MPICP). La différence est que le consommateur utilise ces labels de route pour forcer l'acheminement des Interests sur les chemins spécifiés. De plus, le consommateur possède ici une fenêtre de congestion par chemin. En conclusion de cette sous-partie, nous avons deux solutions qui sont une adaptation du protocole TCP à NDN et deux autres qui essayent de tirer parti de la multiplicité de chemins utilisés dans le réseau pour atteindre une donnée.

1.4.1.2 Saut par saut

Avec NDN, les nœuds dans le réseau possèdent potentiellement plusieurs chemins pour accéder à une donnée. De plus, ils ont la capacité de surveiller les performances de leurs interfaces (RTT, taux de perte, lien congestionné, etc ...) de manière générale mais aussi plus précisément, pour chaque préfixe. Par exemple, un nœud peut surveiller spécialement les performances du préfixe */cnes/etude/icn/* dans le but de fournir un traitement sur mesure aux paquets qui y correspondent. Cela permet à des mécanismes spécifiques aux NDNs de voir le jour. Dans une première sous partie, nous discuterons des solutions qui font du shaping d'*Interest* (ou régulation d'*Interest*). Ensuite dans une deuxième sous partie, nous parlerons des méthodes qui indiquent explicitement aux demandeurs les débits disponibles. Finalement, dans une troisième partie, nous expliquerons le principe des *Forwarding Strategy* et en décrirons les solutions qui se préoccupent du contrôle de congestion.

Shaping Dans cette première sous partie, nous présentons des méthodes qui utilisent le shaping d'*Interest*. Nous définissons d'abord la notion de conversation (introduite dans [2]) : c'est un flux de paires *Interest*/*Data* pour un même préfixe. L'équivalent dans le monde TCP/IP serait les flux TCP (identifié par les adresses IP source et destination et les ports source et destination). Il faut cependant noter qu'ici une conversation n'est pas limitée à une seule source, ni à un seul consommateur.
La première solution provient des mêmes auteurs qu'ICP et s'appelle HR-ICP [50] (Hop-by-hop and Receiver-driven Interest Control Protocol). Ils vont en réalité utiliser ICP pour avoir un contrôle de congestion de bout en bout et ajouter une gestion sur chaque nœud pour obtenir HR-ICP. Chaque nœud va instancier une file d'attente virtuelle pour chacune de ses interfaces et pour chacun des préfixes (ou conversations) actifs et y associer un crédit. Ce crédit sera initialisé au débit équitable du lien et sera décrémenté pour chaque *Interest* transmis. Lorsque le crédit est nul, les *Interests* de cette conversation ne sont plus transmis mais stockés dans la file virtuelle dans le but d'être transmis plus tard (quand la congestion sera terminée). Ce mécanisme a plusieurs avantages :

- Si un utilisateur n'utilise pas ICP mais émet des *Interests* à un débit constant important, les nœuds sur le réseau vont réduire le débit initial au débit qu'ils estiment être équitables pour l'ensemble des conversations. Ainsi, une conversation en CBR ne va pas pénaliser les autres conversations comme cela peut être le cas avec les flux TCP et UDP.
- Le fait de mettre en tampon les *Interests* plutôt que les *Datas* permet d'en stocker plus (les *Interests* étant plus petits que les *Datas*).
- La congestion est détectée plus rapidement. Il faut cependant noter que même si

la congestion est détectée par les nœuds et que les *Interests* sont mis en tampon ou supprimés (par manque de place dans le tampon), les utilisateurs ne détectent la congestion que lorsque leur timer expire.

La deuxième solution proposant du shaping d'*Interest* s'appelle HoBHIS [2]. Son but est de s'assurer que les files de transmission ne soient jamais complètement remplies et que chaque conversation occupe une part égale dans ces files. Pour cela, ils vont calculer un "shaping time" par conversation à chaque réception d'une *Data*. Ce "shaping time" est ensuite utilisé pour savoir combien de temps un *Interest* doit attendre sur le nœud avant de pouvoir être émis. Dans les faits, ils vont calculer un "shaping rate" en fonction de la capacité de l'interface d'émission de la *Data* auquel ils ajoutent le débit que la file peut absorber, c'est-à-dire la taille restante allouée à la conversation divisée par le RTT local (entre le nœud et la source) mesuré. La formule prend donc cette forme :

$$\gamma(t) = C(t) + h \cdot \frac{r - e(t)}{A(t)} \tag{1.1}$$

où C(t) est la capacité de l'interface de sortie des données, r est le seuil de la file d'attente des données, e(t) est le nombre de paquets de donnée dans cette file, A(t) est le délai mesuré pour la conversation et h est un paramètre pour contrôler le dynamisme du système. La figure 1.6 représente le système considéré dans HoBHIS avec toutes ses variables. Cette technique diffère de HR-ICP. En effet, HoBHIS ne fait pas de pré-allocation de

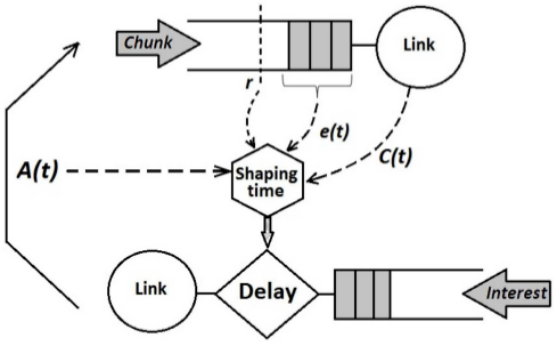


FIGURE 1.6 Représentation du système dans HoBHIS [2]

ressources comme peut le faire HR-ICP avec ses crédits. HoBHIS va en réalité réguler l'envoi des *Interests* pour que les données n'arrivent pas à un débit trop important pour l'interface qui devra les retransmettre et sa file d'attente.

Les deux solutions présentées ne considèrent cependant que du trafic unidirectionnel. IHIS [51] (an Improved Hop-by-hop Interest Shaper), des mêmes auteurs que HoBHIS,

étudie l'interdépendance entre les *Interests* et les *Datas* dans des communications bidirectionnelles. Ils prennent en considération que les *Interests* jouent aussi un rôle dans la congestion et en subissent les conséquences (si un *Interest* est perdu, la donnée ne sera jamais reçue par le demandeur). Ils considèrent donc la quantité de donnée qu'un débit entrant d'*Interests* va provoquer, et cela pour chaque interface pour calculer la nouvelle formule du "shaping rate". Après un travail d'optimisation, ils obtiennent la formule suivante :

$$min\_s_1 + (max\_s_1 - min\_s_1) \cdot (1 - \frac{obs\_s_2}{expmin\_s_2})^2 \tag{1.2}$$

où s_1 est le débit mis en forme sortant et s_2 est le débit mis en forme entrant (obs_s_2 étant le débit mesuré et $expmin\_s_2$ étant le débit optimal attendu). Bien que les trafics bidirectionnels soient pris en compte, cette formule ne distingue plus les différentes conversations. L'équité n'est donc plus assurée. De plus, ils considèrent que le ratio entre la taille d'un *Interest* et de sa *Data* associée est connu et fixe. Ce qui n'est pas le cas puisque la taille d'une donnée est choisie par le créateur de cette donnée (elle est cependant limitée à 8800 octets) et que la taille d'un *Interest* dépend notamment de la taille du nom de la donnée.

Notifications explicites Le problème des techniques de "shaping" d'*Interests* est qu'elles ne permettent pas de réduire directement le volume d'émission d'*Interests* des utilisateurs. C'est pourquoi les auteurs de HR-ICP utilisent le protocole ICP en complémentarité. Les auteurs de HoBHIS ont eux aussi détecté ce problème et ont proposé une solution qu'ils nomment EIR [52] (Explicit Interest Rate). Ils utilisent toujours HoBHIS sur leurs nœuds avec son *shaping rate* mais en plus de cela, les nœuds vont calculer un *tolerance rate* qui sera envoyé aux utilisateurs, via des messages de contrôle spécifiques, pour leur indiquer le débit d'émission à utiliser sur ce chemin. Ces messages de contrôle seront modifiés à la volée par les différents routeurs du chemin. Leur *tolerance rate* n'est pas égal à leur *shaping rate*. En effet, les auteurs expliquent que cela pourrait provoquer des oscillations dans la file des *Interests* dues au fait que le délai entre le nœud congestionné et l'utilisateur soit inconnu. Pour contrer cela, leur *tolerance rate* sera le *shaping rate* pondéré par le pourcentage de places restantes dans la file des *Interests*. La formule est donc la suivante :

$$\gamma^T = \gamma(t) \cdot (1 - \frac{e^I(t)}{B^I}) \tag{1.3}$$

où $e^I(t)$ est le nombre d'*Interest* dans la file et B^I est la taille de cette file. Avec cette formule, lorsque il y a peu de place dans la file d'*Interest*, le *tolerance rate* est faible par rapport au "shaping rate". Cela permet de ne pas faire déborder la file des *Interests* dans le cas où le délai entre ce nœud et l'utilisateur est faible. Ils évitent ainsi bien le problème décrit, mais ils réduisent les performances dans tous les cas où le problème ne se produit pas. L'utilisateur va donc recevoir le *tolerance rate* à intervalle régulier et va pouvoir ajuster son débit d'émission. EIR est donc un protocole qui va indiquer les débits pour éviter la congestion au saut par saut mais c'est finalement l'utilisateur qui va utiliser ce débit, de bout en bout. L'utilisateur ne va plus avoir une fenêtre de congestion qui évoluerait via une méthode AIMD comme dans ICP ou TCP mais va avoir un contrôle basé sur le débit. Il va réduire ou augmenter son débit en concordance avec le *tolerance rate* reçu. Cette méthode implique cependant l'envoi de messages de contrôle spécifiques qu'il faut gérer différemment des autres messages déjà définis dans NDN. Au vu de l'architecture de NDN, cela peut être compliqué et le plus simple est de simplement "piggybacker" le *tolerance rate* dans les messages de donnée, un peu comme cela est fait dans XCP [53], qui est d'ailleurs un protocole très proche dans le monde d'IP. Nous pouvons soulever deux autres difficultés liées à cette solution :

— Le *tolerance rate* reçu par l'utilisateur est le débit disponible pour la conversation sur un chemin. L'utilisateur récupère la donnée sur potentiellement plusieurs chemins, bien qu'il n'en soit pas conscient. Utiliser ce débit-là ne correspond donc pas à la réalité du réseau.

— Le *tolerance rate* calculé sur les routeurs correspond à une conversation, qui est potentiellement utilisée par plusieurs consommateurs. Ces multiples utilisateurs vont donc recevoir ce *tolerance rate*, émettent leurs *Interests* à ce débit et s'ils sont désynchronisés (l'un demande les *Datas* 1, 2, 3 ... tandis qu'un autre a de l'avance et demande les *Datas* 101, 102, 103, ...), les routeurs sur le chemin ne vont pas être capables d'agréger les deux flux. Le débit sera finalement double et les nœuds seront congestionnés.

Malgré ses difficultés, cette solution propose un nouveau point de vue et met en lumière des nouveaux problèmes.

Forwarding Strategy Dans cette sous-partie, nous présentons des solutions de la famille des *Forwarding Strategy* qui s'occupent du contrôle de congestion. Nous verrons dans la section 1.4.4 d'autres *forwarding strategy* dont le but n'est pas le contrôle de congestion. Commençons par définir une *Forwarding Strategy*. C'est un algorithme qui est implanté dans les nœuds NDN et qui leur permet de choisir où transmettre un *Interest*

parmi les possibilités données par la FIB (pour un préfixe donné, la FIB renvoie en effet une liste de "next hops"). Suivant la solution, le but peut être d'optimiser un ou plusieurs critères. Ici, nous nous intéressons uniquement aux solutions qui se proposent d'éviter la congestion.
Les prémices des méthodes de *Forwarding Strategy* sont posées dans [3]. Dans cet article, les auteurs vont compléter l'architecture de NDN. Comme nous pouvons le voir dans la

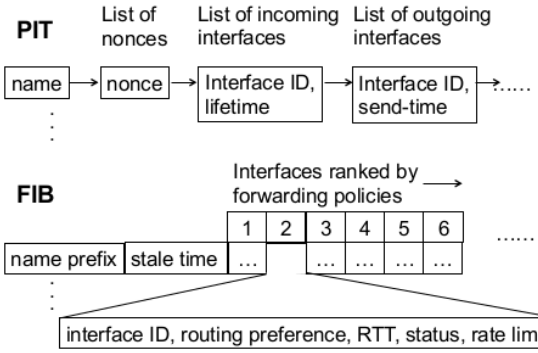


FIGURE 1.7 Contenu d'une entrée PIT et FIB [3]

Figure 1.7, à chaque interface d'une entrée FIB sont associés de multiples paramètres qui vont aider à la sélection de l'interface utilisée pour la transmission d'un *Interest*. Ces paramètres sont mis à jour lors de la réception de données. De plus, les auteurs vont définir les messages NACK (qui ont été implantés dans NDN) et qui permettent aux routeurs d'indiquer qu'ils n'ont plus de chemin vers un préfixe ou que les chemins le traversant sont congestionnés. L'algorithme décrit dans cet article, que nous appelons ici Color-based Forwarding (CbF), classe les différentes interfaces en fonction de ces multiples paramètres. En plus de ce classement, ils vont avoir trois types d'interfaces :

- les interfaces "vertes" qui sont celles qui fonctionnent ;
- les interfaces "jaunes" qui sont celles qui peuvent fonctionner (mais dont l'algorithme n'est pas sûr) ;
- les interfaces "rouges" qui sont celles qui ne fonctionnent pas.

Une interface verte passe en jaune si une entrée PIT expire ou si elle n'est pas utilisée pendant un certain temps. Une interface passe en rouge si le nœud a reçu un NACK indiquant que le chemin n'est plus disponible. Ils vont donc utiliser la meilleure interface verte si disponible, sinon la meilleure jaune. S'il n'y a ni interface verte, ni interface jaune disponible, le nœud va envoyer un message NACK. Pour ce qui est du contrôle de congestion, cet algorithme définit deux *interest rate limit*. Le premier *interest rate limit*

est défini par interface et vaut :

$$L_i = \alpha \cdot \frac{C_i}{\overline{S}_i} \tag{1.4}$$

où C_i est la capacité de l'interface, $\overline{S}_i$ est la taille moyenne d'une donnée et α est un paramètre permettant de compenser les erreurs de calculs. Il est utilisé pour faire du shaping, un peu comme pour HoBHIS, sauf que lorsque le débit limite est atteint, l'algorithme ne va pas stocker les prochains *Interests* pour les transmettre plus tard mais va utiliser une autre interface. Si aucune autre interface n'est disponible, le nœud va émettre un NACK pour notifier une congestion au nœud lui ayant envoyé l'*Interest*. Le deuxième *interest rate limit* est, lui, défini par préfixe et par interface. C'est celui que nous pouvons voir dans la figure 1.7. Il permet de garder une trace du débit disponible jusqu'à la ou les sources disponibles via cette interface. Il est mis à jour lorsqu'une donnée ou un NACK de congestion arrive. Color-based Forwarding est une première approche intéressante qui, même si elle ne définit pas clairement d'algorithme de sélection du "next hop", définit les outils pour cela et met en valeur le potentiel des *Forwarding Strategy*.
Une autre solution, similaire mais plus approfondie, est définie dans [54] et se nomme FPF (pour Fast Pipeline Filling). Leur but est de remplir complètement tous les chemins disponibles, le plus rapidement possible. FPF va compter les *Pending Interests* (PI), les *Interests* transmis qui n'ont pas encore eu de réponse, pour chaque interface et calculer le nombre limite de PI que chaque interface peut avoir avant d'être congestionnée. La capacité d'une interface est calculée suivant la formule suivante :

$$C_i = 2 \cdot R_i \cdot D_i + L_i \tag{1.5}$$

où R_i est débit de l'interface, D_i est le délai de propagation moyen de l'interface et L_i est la taille du tampon de l'interface en nombre de *Datas*. Cette capacité se veut représentative du nombre maximal d'*Interests* que le nœud peut émettre avant de recevoir la première *Data* s'il ne veut pas créer de congestion. Lors de la transmission d'un *Interest*, parmi les interfaces proposées par la FIB qui n'ont pas atteint leur capacité, FPF va choisir celle qui possède le plus petit RTT moyen. En choisissant l'interface au plus petit RTT, FPF souhaite avoir une réponse rapidement et donc une libération de place plus rapide que sur une autre interface.
DMF [55] (a Dynamic Multipath Forwarding strategy) est très proche de FPF. Tout d'abord, les auteurs de DMF ont modifiés la formule 1.5 pour prendre en compte la taille des données et la capacité des nœuds à mettre en file d'attente. En effet, C_i et L_i sont en nombre d'*Interest*/*Data* tandis que $2 \cdot R_i \cdot D_i$ est un débit. C'est pourquoi ils vont définir

une première capacité, le *Bandwidth Delay Product* de la manière suivante :

$$BDP_i = \frac{BW_i \cdot RTT_i}{S_D} \tag{1.6}$$

où BW_i est la bande passante de l'interface, RTT_i est le délai moyen et S_D est la taille moyenne d'une *Data*. La capacité totale est donc définie ainsi :

$$C_i = BDP_i + B_i \tag{1.7}$$

où B_i est la taille du tampon de l'interface en nombre de *Datas*. DMF utilise ces deux capacités de la manière suivante :

- S'il y a des interfaces dont le nombre de PI est inférieur à BDP_i, l'*Interest* est transmis sur celle qui a le plus petit RTT (ils font comme FPF dans ce cas-là),
- sinon s'il y a des interfaces dont le nombre de PI est inférieur à C_i, l'*Interest* est transmis sur celle qui a le plus grand débit (c'est ici qu'ils diffèrent de FPF),
- sinon, l'*Interest* est supprimé et le nœud répond avec un NACK de congestion (comme FPF).

Les auteurs justifient cet algorithme par l'utilisation d'un protocole de contrôle de la congestion bout en bout tel que ICP et par l'analyse de l'évolution de la taille de la fenêtre de congestion. Ils expliquent que lorsque la fenêtre de congestion *cwnd* est inférieure à BDP_i, le débit de réception vaut $cwnd * S_D/RTT_i$ (dépendant du délai), tandis que quand cwnd est supérieur à BDP_i, le débit de réception vaut le débit du goulot d'étranglement (dépendant du débit). Bien sûr, lorsque les tampons des routeurs sont remplis, les paquets sont perdus.

FPF et DMF proposent un contrôle de congestion novateur en comparaison de ceux existant dans le monde d'IP. Ils permettent en effet d'utiliser un des chemins vers la donnée qui n'est pas congestionné et de supprimer l'*Interest* si un tel chemin n'existe pas. Ce qui permet de réduire la congestion globale dans le réseau.

QoS-FS [56] (pour Quality of Service - Forwarding Strategy), une autre solution axée sur la gestion de la QoS, propose de faire de la pré-allocation de ressources pour gérer la congestion. Les utilisateurs vont indiquer dans les *Interests* qu'ils envoient les paramètres de QoS dont ils ont besoin pour leur communication (délai et bande passante). Les nœuds vont monitorer le délai pour chaque préfixe sur chaque interface donnée par la FIB grâce à un apprentissage par renforcement (Q-learning). Lorsqu'ils transmettent un *Interest*, les nœuds vont réduire la bande passante disponible des interfaces concernées. Leur algorithme de sélection d'interface est le suivant :

- parmi les interfaces proposées par la FIB, ne garder que celles qui ont suffisamment

de bande passante disponible et dont le délai est inférieur à celui indiqué dans l'*Interest* ;
— choisir celle avec le plus petit délai ;
— si aucune interface ne convient, envoyer un NACK indiquant qu'il n'y a pas de chemin par l'interface d'où vient l'*Interest*.

Tant qu'il n'y a pas eu d'échange de données, les nœuds ne connaissent pas les délais pour un préfixe. C'est pourquoi QoS-FS utilise une phase d'exploration, qui permet de mesurer les premiers délais, et une phase d'exploitation, qui est celle décrite précédemment. Durant cette phase d'exploration, les nœuds vont sélectionner l'interface à utiliser de manière cyclique parmi les disponibles. Nous reviendrons sur les performances de la gestion de la QoS dans la partie 1.4.3 qui lui est dédiée. La pré-allocation des ressources, bien qu'elle assure qu'il n'y aura pas de congestion, a de nombreux désavantages. La bande passante est, en effet, bloquée jusqu'à ce que la donnée soit finalement arrivée sur le nœud. Le problème est d'autant plus flagrant lorsqu'il y a de gros délais, par exemple lors de l'utilisation d'un lien satellite sur le chemin. Il en résulte donc une sous-utilisation des ressources.

En conclusion, il y a deux styles pour le contrôle de la congestion dans les *Forwarding Strategy*. Le premier est de faire de la pré-allocation de ressources comme dans Color-based Forwarding avec ses *interest rate limit* et dans QoS-FS. Le deuxième est de calculer une valeur théorique d'*Interests* à émettre avant de créer de la congestion. Cette deuxième méthode fait un pari fort sur le délai et suppose notamment que le délai est le même pour chaque préfixe. Ils n'ont cependant pas le problème systématique de sous-utilisation des ressources que les méthodes utilisant la pré-allocation ont.

1.4.1.3 Hybride

Finalement, un dernier type de solution propose d'avoir à la fois un comportement de bout en bout et un comportement au saut par saut. Ce sont les solutions que nous appelons solutions hybrides. PCON [57] (pour Practical Congestion cONtrol scheme) est l'une d'entre elles. Pour la partie bout en bout, elle propose d'utiliser une fenêtre de congestion du côté du consommateur (comme pour ICP). Pour la partie saut par saut, elle propose à la fois d'utiliser une AQM (pour Active Queue Management) pour détecter si un phénomène de congestion se produit localement et une *forwarding strategy* pour faire de la répartition de charge et utiliser en priorité les chemins les moins chargés. L'AQM utilisée par PCON s'appelle CoDel [58]. Elle consiste à mesurer le temps de séjour des paquets dans la file d'émission. Si ce temps dépasse un certain seuil, l'interface associée est considérée congestionnée par PCON. Dans ce cas, PCON va explicitement

marquer les paquets Data utilisant l'interface congestionnée. Ces marques sont ensuite interprétées par la *forwarding strategy* de PCON, pour rediriger une partie du trafic sur d'autres chemins, et par le mécanisme de bout en bout. En effet, le consommateur va réduire la fenêtre de congestion lors qu'un timer expire (comme pour ICP) mais aussi quand il reçoit une Data marquée. PCON peut ainsi être divisé en trois sous-parties. La partie bout en bout avec la fenêtre de congestion chez le consommateur, que nous appelons PCON-CS (pour Consumer Side), la forwarding strategy que nous appelons PCON-FS et une dernière qui fait de la notification explicite de congestion suite au marquage des Data par une AQM (utilisée par les deux premières parties). PCON-CS et PCON-FS peuvent être utilisés indépendamment et couplés avec d'autres mécanismes de saut par saut et bout en bout, respectivement. Nous verrons justement des évaluations sur différentes combinaisons dans la partie 1.4.5.

1.4.1.4 Conclusion

Dans cette partie, nous avons décrit les deux familles de méthodes pour le contrôle de congestion : les méthodes bout en bout et les méthodes au saut par saut. Nous avons vu trois types de solutions pour le contrôle de congestion au saut par saut. Ces différents types ont tous pour but d'éviter la congestion mais n'utilisent pas les mêmes méthodes :

- les solutions de type "shaping" profitent du fait que les *Interests* soient moins volumineux que les *Datas* ;
- les solutions de type "notification explicite" agissent sur le débit des *Interests* de l'utilisateur ;
- les solutions de type "forwarding strategy" profitent de la multiplicité des chemins pour éviter ceux qui sont congestionnés.

Toutes ces solutions sont compatibles entre elles et une solution qui tirerait parti de chaque méthode est envisageable. De plus, chaque solution suppose qu'il y a, en parallèle, un contrôle de congestion de bout en bout, qui s'assure notamment de la fiabilisation de la communication.

C'est d'ailleurs une supposition qu'il faut potentiellement abandonner. En effet, le contrôle de congestion de bout en bout va réduire son débit d'émission lorsqu'il détecte une perte (via un timer) alors que cette perte est potentiellement provoquée par un autre protocole de contrôle de congestion qui s'effectue au saut par saut. De nouveaux mécanismes essayent de contourner ce problème. MPICP mesure le délai de chaque route par exemple, mais cela nécessite une modification du design de NDN. De plus, cela ne garantit pas que la réduction de la fenêtre de congestion soit en adéquation avec les conditions du réseau. Une autre solution, exploitée par EIR, est d'indiquer explicitement au récepteur

les débits disponibles.
Trois solutions semblent donc émerger :

— une combinaison de protocoles de contrôle de congestion de bout en bout et saut par saut avec le bout en bout qui est conscient et en phase avec le saut par saut ;
— la même combinaison mais dans ce cas c'est le saut par saut qui va explicitement indiquer au bout en bout comment se comporter (comme le fait PCON) ;
— simplement du saut par saut pour le contrôle de congestion, l'utilisateur va émettre en fonction de ses besoins sans dépasser la capacité de son lien.

1.4.2 Gestion des pertes

Dans le monde d'IP, c'est TCP qui s'occupe de fiabiliser les communications, notamment via des timers. Dans NDN, c'est aussi les solutions de bout en bout, "TCP-like", tels que ICP et ICTP qui s'en chargent. Si nous décidons de ne plus faire de contrôle de congestion de bout en bout, nous pouvons toujours avoir un protocole qui s'occupe du contrôle de flux et de la fiabilisation (renvoie des *Interests* après timeout).
Cependant, il reste imaginable que les nœuds dans le réseau soient aussi en charge de la fiabilisation. Ils possèdent déjà des timers pour leurs entrées PIT, qui leur sert à déterminer quand l'*Interest* ou la *Data* est considérée comme perdu et donc quand supprimer l'entrée PIT pour économiser de la place. Certaines *Forwarding Strategies* s'en servent notamment pour décider qu'une interface n'est plus capable de fournir les données d'un préfixe et mettent à jour son statut dans l'entrée FIB correspondante (Color-based Forwarding par exemple).
Les auteurs de [59] proposent une solution qui permet aux consommateurs d'indiquer s'ils souhaitent que la gestion des pertes soit faite sur chaque nœud ou de bout en bout (par eux-mêmes). Ils expliquent que suivant le type d'application, une gestion des pertes au saut par saut est nécessaire. Dans leur solution, l'utilisateur va ajouter un champ *Interest Retransmission Policy* dans son *Interest*. Si ce champ vaut 1, les nœuds sur le réseau vont devoir gérer la retransmission de l'*Interest* s'ils n'ont pas reçu de réponse avant la fin de leur timer. La solution n'indique cependant pas comment est choisi ce timer, ni quel comportement adopter lorsque la perte est détectée (retransmission sur la même interface, utilisation d'une autre interface si disponible ?). Les auteurs indiquent seulement que les nœuds du réseau ont une limite de 10 retransmissions avant d'abandonner et d'émettre un NACK en amont. Ils laissent la décision de comment retransmettre à la *Forwarding Strategy*. Les auteurs ajoutent aussi un champ *Network Retransmission Differentiation* aux paquets *Interests* pour que les nœuds puissent différentier les *Interests* initiaux des *Interest* retransmis. Les *Interests* retransmis sont, en effet, envoyés avec le même "Nonce",

mais avec une valeur différente de ce champ (qui est un entier et permet de savoir à quel numéro de retransmission cet *Interest* correspond). Une autre solution consisterait à simplement changer le "Nonce" pour que l'*Interest* ne soit pas ignoré car considéré comme dupliqué. Leur solution permet cependant au nœud de savoir qu'un Interest correspond à une retransmission ou non et un traitement spécifique peut être mis en place. Il peut, en effet, estimer qu'il y a effectivement une perte ou au contraire se dire que le nœud en amont n'a pas bien réglé son timer et décider d'attendre la fin de son propre timer avant de transmettre cet *Interest*.
Néanmoins, une gestion des pertes au saut par saut complexifierait les nœuds du réseau et le passage à l'échelle pourrait être compromis.

1.4.3 Gestion de la QoS

Une première solution, QoS-FS, a été décrite dans la partie 1.4.1.2. Dans QoS-FS, les utilisateurs vont indiquer leurs contraintes de QoS (délai et bande passante) dans les *Interests*. Les nœuds vont observer ces paramètres pour chaque préfixe et interface et sélectionner l'interface au plus petit délai parmi celles qui satisfont les contraintes indiquées. Surveiller le délai de chaque préfixe sur chaque interface (donnée par la FIB) peut se révéler long (phase d'exploration), inefficace (durée non négligeable) et coûteux (en termes d'espace mémoire sur un nœud).
AAF [4] (pour Application-Adaptive Forwarding Strategy) propose de ne superviser le délai moyen que par interface. Si cette proposition est très approximative, elle a toutefois l'avantage d'éviter une phase d'exploration et de réduire la taille d'une entrée FIB. AAF utilise aussi des classes de service déductibles du nom de la donnée. Il n'y a donc pas besoin de rajouter un overhead dans les *Interests* comme c'est le cas dans QoS-FS. Les différentes interfaces d'un nœud vont se voir assigner un poids en fonction de leur délai moyen. Les *Interests* étant affiliés aux meilleures classes de service vont être transmis sur les interfaces avec le meilleur poids (c'est-à-dire au plus court délai moyen). AAF met aussi en place un mécanisme pour que les *Interests* puissent être satisfaits dans les temps. À chaque classe va correspondre un délai maximum acceptable. Si le délai de l'interface choisie pour une classe est supérieur au délai maximum de la classe, AAF va permettre à cet *Interest* d'utiliser une interface prévue pour une meilleure classe. Pour cela, chaque classe va avoir un *interest releasing interval* qui correspond au temps que peut attendre un *Interest* dans les files d'attente avant de ne plus respecter sa contrainte de QoS. La classe qui ne peut pas respecter sa contrainte de QoS avec ses interfaces va utiliser toutes les interfaces disponibles en retardant les autres classes d'au maximum leur *interest releasing interval*. La figure 1.8 nous montre le cas où quatre classes de service

sont disponibles et où la deuxième n'a, temporairement, pas des interfaces capables de fournir le service demandé. AAF va donc retarder les *Interests* de la classe 1 et permettre

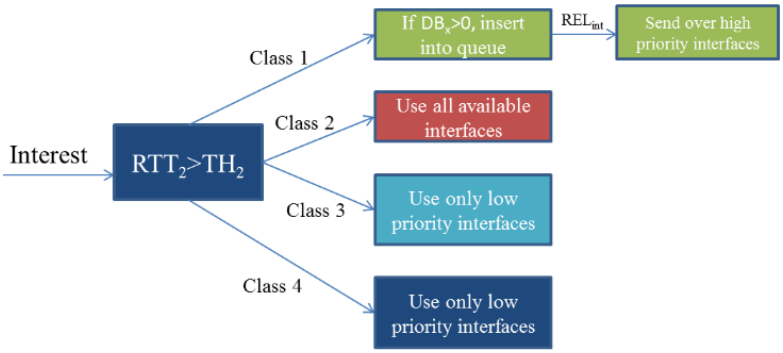


FIGURE 1.8 Cas où la classe 2 n'est pas satisfaite [4]

à ceux de la classe 2 d'utiliser toutes les interfaces possibles. Une fonctionnalité notable de cet algorithme est qu'il permet à des *Interests* d'une classe d'utiliser des interfaces d'une classe "inférieure". Cela peut s'expliquer par le fait que le délai moyen dépend seulement de l'interface et que pour le préfixe demandé, le délai est potentiellement plus court. AAF ne va pas supprimer l'*Interest* quand il pense qu'il n'est pas capable de fournir le service demandé, contrairement à QoS-FS. Il définit d'ailleurs l'*interest multicasting* pour maximiser ses chances d'avoir une réponse dans les temps. Cette technique va cependant favoriser les congestions, qui ne sont d'ailleurs pas prises en compte dans cet algorithme. Un autre problème de cette solution est que chaque nœud considère le délai de son point de vue (de lui jusqu'à la source) alors que la classe de service considère le délai complet (de l'utilisateur à la source). Dans le cas où un nœud déciderait de retarder des *Interests* d'une classe de service (pour permettre à une autre classe de service de respecter ses délais), les nœuds suivants ne sont pas avertis du délai cumulé sur les nœuds précédents. S'ils décident de retarder ces *Interests* à nouveau, les impératifs de la classe de service peuvent très bien ne pas être respectés.

Les deux algorithmes présentés prennent, donc, des chemins très différents pour la gestion de la QoS : classe de service déductible du nom de la donnée demandée pour l'une contre insertion explicite des contraintes de QoS dans les *Interests* pour l'autre ou encore partage des interfaces entre classe de service pour l'une contre utilisation de la meilleure interface qui respecte les contraintes pour l'autre. Pour autant, les deux solutions ne se basent que sur le délai comme critère déterminant de QoS (QoS-FS prend en compte le débit mais seulement pour s'assurer qu'il n'y aura pas de congestion). De nombreux services ne sont pas spécialement sensibles au délai et demande surtout un bon débit pour satisfaire les

utilisateurs (visualisation de vidéos ou écoute de musique par exemple). Une proposition prenant en compte ce fait est alors susceptible de fournir une gestion de la QoS plus efficace que les solutions présentées.
Les auteurs de [60] suivent cette voie. Ils définissent deux protocoles en fonction de la sensibilité du trafic au délai ou à la bande passante : ACFS (pour Ant Colony Forwarding Strategy) et RBIT (Routing for Bandwidth Intensive Traffic), respectivement.
ACFS utilise un algorithme de colonies de fourmis. Un champ "phéromone" est ajouté dans les entrées FIB pour chaque interface (φ_i pour l'interface i). Un nœud va transmettre un *Interest* sur l'interface qui possède la meilleure phéromone. De plus, toutes les autres interfaces auront une probabilité de transmettre aussi l'*Interest*. La probabilité est calculée de la manière suivante (pour l'interface i) :

$$p_i = \frac{\varphi_i^\alpha}{\sum_{j=1}^{n} \varphi_j^\alpha} \tag{1.8}$$

où α représente un facteur d'échelle et n le nombre total d'interfaces. Les phéromones évoluent ensuite quand une *Data* arrive :

— si c'est la première fois qu'elle arrive, la phéromone de son interface d'arrivée augmente de $\nabla_\varphi = 0.1$, la quantité de phéromones déposées ;
— sinon, la phéromone de son interface d'arrivée est multiplié par $\nabla_\varphi = 0.5$, le facteur d'évaporation de phéromones.

Cet algorithme va favoriser les interfaces qui ont un court délai puisque celles-ci vont plus régulièrement augmenter leurs phéromones. De plus, l'utilisation probabiliste des autres interfaces permet de mesurer régulièrement leur délai et de profiter ainsi rapidement d'une amélioration sur un chemin.
RBIT, quant à lui, est un algorithme qui va aider les flux sensibles au débit à obtenir la meilleure bande passante possible. Pour cela, les consommateurs initialisent la communication par le "multicast" d'un nouveau type de message, les info-*Interests*, vers chaque source de la donnée. En retour, les producteurs répondent avec des info-*Data* où sera indiquée la bande passante disponible ainsi que le nom de la source. Les nœuds sur le chemin agrègent toutes les réponses et mettent à jour la bande passante disponible si leur lien est le goulot d'étranglement. Quand l'utilisateur reçoit finalement l'info-data (tous les nœuds sur le chemin attendent d'avoir toutes les réponses de chaque source), il peut commencer à émettre ses *Interests*. Chaque nœud sur le chemin va utiliser l'interface qui propose le plus de bande passante. Cette technique ne nous semble pas efficace car :

— sa phase d'initialisation est faite à l'initiative de l'utilisateur. Les données ras-

semblées ne sont pas réutilisables par d'autres utilisateurs ;
— la phase d'initialisation consiste à contacter toutes les sources, ce qui peut se révéler très long avant de finalement pouvoir commencer la communication ;
— l'algorithme ne tient pas compte de la congestion, seulement de la capacité des liens. Dans le cas où il y a plusieurs utilisateurs, les liens choisis ne seront pas forcément les meilleurs à utiliser.

1.4.4 Forwarding Strategy

Nous avons déjà décrit plusieurs *Forwarding Strategy* dans les sections précédentes. Commençons ici par la solution qui nous paraît la plus simple : ASF [61] (pour Adaptive Smoothed RRT-based Forwarding). Dans cette solution, les nœuds maintiennent un *smoothed RTT*, calculé de la même manière que dans TCP, pour chaque interface. Il est recalculé à chaque réception de *Data*. Lors de la réception d'un *Interest*, le nœud choisi l'interface qui a le plus petit *smoothed RTT* pour transmettre l'*Interest*. De plus, les nœuds vont régulièrement explorer les autres chemins pour apprendre leur délai. Un peu comme le fait déjà ACFS, les autres interfaces non sélectionnées par l'algorithme ont une certaine probabilité de transmettre aussi l'*Interest*. L'exploration se fait de manière périodique et les probabilités des interfaces sont calculées comme suit :

$$P(i) = 2\frac{N+1-i}{N(N+1)} \tag{1.9}$$

où i est le rang de l'interface (le rang 1 pour l'interface au plus petit *smoothed RTT*) et N est le nombre d'interfaces. Une interface avec un haut rang (qui proposait de bonnes performances précédemment) a donc plus de chance d'être sélectionnée pour l'exploration et donc d'être à jour.
La prochaine solution, que nous noterons ici DRF [48] pour Dynamic Request Forwarding, propose de se baser sur le nombre de PI pour choisir l'interface où transmettre l'*Interest*. À chaque réception de *Data* et émission d'*Interest*, le nœud va mettre à jour un compteur de Pending Interest par interface et par préfixe. De la même manière, il va maintenir un poids pour chaque interface et chaque préfixe. Ce poids est calculé de la façon suivante :

$$weight(face, prefix) = \frac{1}{\overline{PI}(face, prefix)} \tag{1.10}$$

où $\overline{PI}(face, prefix)$ est la moyenne glissante du nombre de PI pour le préfixe et l'interface considérés. Ce poids est alors utilisé pour déterminer quelle interface utiliser pour transmettre un *Interest*. L'algorithme propose l'utilisation d'une fonction aléatoire

pondérée pour sélectionner l'interface à utiliser. De cette manière, moins une interface a de PI, plus elle a de chance d'être utilisée. Cet algorithme permet de bien répartir une conversation sur l'ensemble des interfaces où la source est disponible et donc d'utiliser de la même manière tous les chemins disponibles. Cependant, il ne tient pas compte de la capacité des interfaces, ce qui est étonnant puisqu'un lien avec un grand débit peut émettre plus rapidement des *Interests* et, de même, recevoir les *Datas* à un plus grand débit. Il n'y a pas de considération en ce qui concerne la congestion non plus. Le contrôle doit donc être fait de bout en bout.
Finalement, la dernière solution MDPF [62] (pour Maximizing Deviation based Probabilistic Forwarding) est surtout intéressante par sa manière de poser le problème. En effet, l'algorithme utilisé pour la sélection de l'interface est le suivant :

— le nœud construit une matrice de décision avec les interfaces disponibles données par la FIB et les différents paramètres à prendre en compte dans la sélection (n interfaces, m paramètres). Cela définit ce que les auteurs appellent un problème de prise de décision à multiples paramètres (MADM pour Multiple Attribute Decision Making) ;
— il calcule ensuite le vecteur de poids optimal en utilisant un algorithme de maximisation de la déviation (MD pour Maximizing Deviation) ;
— il calcule ce que les auteurs appellent le score de l'interface (multiplication du vecteur de poids optimal calculé précédemment par le vecteur de l'interface) ;
— il sélectionne finalement l'interface qui transmettra l'*Interest* grâce aux scores calculés à l'étape précédente. Pour cela, il utilise un algorithme de sélection probabiliste comme décrit pour DRF, et appelé *roulette wheel selection*. Plus une interface possède un grand score et plus elle a de chances d'être sélectionnée.

Cet algorithme ressemble beaucoup à celui de DRF mais avec plusieurs paramètres pour effectuer la décision. C'est d'ailleurs assez théorique car les auteurs du papier n'utilisent que le délai et le nombre de PI dans leur évaluation. Cet algorithme est une première version qui nous semble prometteuse. Il faut maintenant trouver les bons paramètres à utiliser lors de la sélection pour répondre à toutes les problématiques qui nous intéressent. De plus, l'étape 2 est potentiellement une étape longue à exécuter. Une solution serait de trouver ces paramètres par des expériences et de ne pas avoir à les calculer à chaque fois. Cela réduirait le temps de traitement pour chaque *Interest* et permettrait aux nœuds de gérer les *Interests* sans en ralentir le débit (ou très peu).

1.4.5 Évaluation des performances

Dans cette partie, nous présentons les principaux résultats de notre papier [5]. Le but de cette étude est de comprendre les enjeux du contrôle de congestion dans NDN. Nous avons ré-implanté certaines des solutions présentées précédemment sur le simulateur ndnSIM [63]. Celui-ci se base sur le simulateur de réseaux ns3 et implante le modèle de communication de NDN. En effet, il utilise directement les bibliothèques de NDN, ndn-cxx [2] et NFD [3]. Cela permet une portabilité théoriquement simple du code développé dans le simulateur vers un environnement réel. Nous avons ré-implanté ICP, PCON et FPF pour ce qui est du contrôle de congestion, mais aussi la *forwarding strategy* DRF pour du partage de charge. Les sources sont disponibles librement pour reproduire et utiliser nos résultats [4].

Pour notre premier scénario, nous utilisons la topologie illustrée sur la Figure 1.9. Le

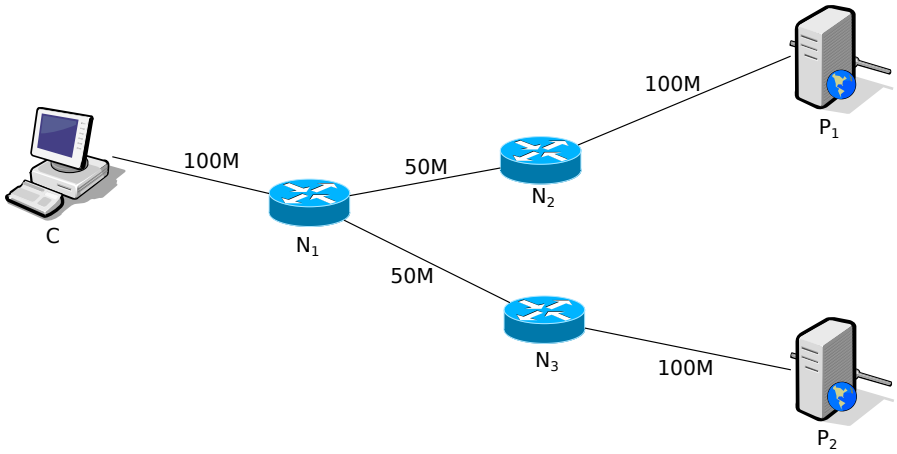


FIGURE 1.9 Topologie de test

consommateur C peut récupérer le contenu qu'il cherche depuis deux producteurs, P_1 et P_2. Le flux entre le consommateur C et ses deux producteurs dispose donc de deux chemins, un pour chaque producteur :

— un premier, reliant C à P_1 via les nœuds N_1 et N_2,

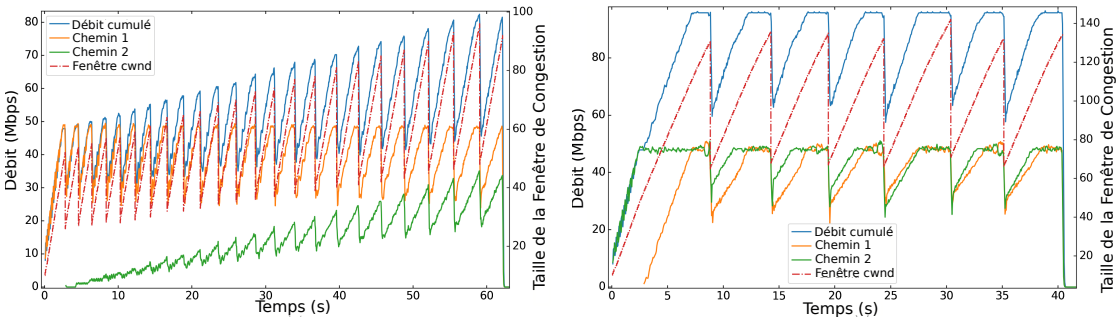
— et un second, reliant C à P_2 via les nœuds N_1 et N_3.

Dans le cas idéal, le consommateur utilise les deux chemins à sa disposition avec un débit de 50Mbps sur chacun et au total 100Mbps. La table 1.1 résume les débits, délais et temps de séjour moyens obtenus sur ce scénario avec chaque algorithme de bout en

2. https://github.com/named-data/ndn-cxx
3. https://github.com/named-data/NFD
4. https://gitlab.tesa.prd.fr/athibaud/guidelines-codebase

TABLE 1.1 Resultats du premier scenario.

	Algorithme de bout en bout	Forwarding Strategy		
		BR	PCON-FS	DRF
Débit (Mbps)	PCON-CS	40.2	52.9	64
	ICP	42	81.3	71.9
Délai (ms)	PCON-CS	64.3	69.4	74.6
	ICP	72.9	78	76.5
Temps de séjour (ms)	PCON-CS	2.1	2	2.4
	ICP	10.3	6.5	3.9



(a) PCON-CS avec PCON-FS (b) ICP avec PCON-FS

FIGURE 1.10 Débits et fenêtre de congestion pour deux combinaisons

bout, ICP ou PCON-CS, couplé à chaque *forwarding strategy*, PCON-FS, DRF, ou BR [5]. Comme le calcul original du RTO dans ICP a montré de faibles performances, nous avons utilisé la méthode de TCP, décrite dans [64]. C'est aussi cette méthode que nous avons utilisée pour le calcul du RTO dans PCON-CS. La Figure 1.10a montre les débits cumulé et sur chacun des deux chemins du consommateur avec les algorithmes PCON-CS et PCON-FS (la proposition PCON dans son intégralité). Elle montre aussi l'évolution de la taille de la fenêtre de congestion du consommateur. De la même manière, la Figure 1.10b correspond à l'utilisation du couple d'algorithmes ICP et PCON-FS.

En utilisant la *forwarding strategy* BR, les débits n'excèdent jamais les 50Mbps. La raison est simple et provient du fait que cette FS n'utilise qu'un seul des deux chemins. C'est un résultat attendu et qui ne nous surprend pas. Le comportement le plus surprenant de ces évaluations est celui du couple PCON-CS et PCON-FS. En effet, la proposition originelle de PCON montre une surréaction à un évènement de congestion. Comme nous le voyons sur la Figure 1.10a, au fil de l'augmentation de la fenêtre de

5. BR pour Best Route. C'est la FS de base dans ndnSIM. Elle renvoie toujours la même interface, celle qui est la mieux notée par l'algorithme de routage.

congestion, le chemin 1 est progressivement utilisé en intégralité. Puis, le nœud N_3 va commencer à marquer les paquets Data. Le nœud N_1 bascule alors une partie du trafic sur le chemin 2, vers N_2. Mais ces marques sont aussi interprétées par le consommateur qui réduit de moitié sa fenêtre de congestion. Après plusieurs itérations, le nœud N_1 devrait trouver le bon pourcentage de trafic à envoyer sur chaque chemin. Malheureusement, ce processus est relativement long et le consommateur a finalement réussi à récupérer le contenu entièrement mais à un débit moyen loin de l'optimal. À l'inverse, le couple ICP et PCON-FS montre de bonnes performances (cf Figure 1.10b). Une fois le premier chemin utilisé totalement, le nœud N_3 marque les Data mais ces marques ne sont plus interprétées que par le nœud N_1 qui opère le basculement. Le consommateur continue à augmenter normalement sa fenêtre de congestion. Ce n'est finalement que lorsque les deux chemins sont utilisés entièrement qu'une perte se produit et que le consommateur réduit sa fenêtre de congestion. Un résultat supplémentaire est qu'avec ICP, les délais de bout en bout sont plus longs qu'avec PCON-CS. En effet, PCON-CS réagit avant que les files ne soient remplies alors qu'ICP attend qu'une perte se produise pour réduire sa fenêtre de congestion. Finalement, le couple PCON-CS et DRF proposent un meilleur débit que le couple PCON-CS et PCON-FS car il n'y a pas les deux mécanismes qui réagissent aux marques comme décrit plus haut. Le couple ICP et DRF proposent quant à lui, de moins bonnes performances en termes de débit par rapport au couple ICP et PCON-FS. Cela vient de la méthode probabiliste de choix de l'interface à utiliser.

Un deuxième scénario où plusieurs flux se croisent et utilisent les mêmes liens met en valeur l'importance d'une forme d'équité entre ces flux. Nous y reviendrons plus en détail dans les chapitres suivants.

Finalement un autre scénario avec une topologie plus large corrobore nos précédents résultats [5].

1.5 Conclusion

Dans ce chapitre, nous avons fait un historique de toutes les architectures ICN qui ont été définies. Bien qu'elles reposent toutes sur le fait de mettre le contenu au centre du réseau, elles ne s'y prennent pas toutes de la même manière. Après avoir décrit comment chacune de ces architectures implantait les principales caractéristiques des ICNs (nommage du contenu, routage, sécurité, etc ...), nous nous sommes concentrés sur celle qui nous semble la plus mature : Named Data Networking. En effet, de nos jours, la grande majorité des contributions de recherche sur les ICNs se font sur cette architecture. Son succès par rapport à ses pairs vient à la fois de sa structure décentralisée, de sa modularité

via les *forwarding strategy* et de la simplicité pour tester de nouvelles contributions grâce au module de ns3, ndnSIM, qui l'implante avec fidélité. Les changements sur le routage apportés par NDN offrent de nouvelles opportunités pour le transport. Comme vu précédemment, les nœuds dans NDN ont potentiellement plusieurs choix d'interface pour la transmission des *Interests*. De plus, ils peuvent monitorer activement différents paramètres, comme le délai moyen d'une interface ou plus spécifiquement pour un préfixe donné. Différentes techniques utilisent ces informations dans le but d'éviter la congestion, de gérer de la QoS, fiabiliser les communications ou encore assurer un partage de charge sur les différents chemins. La Figure 1.11 rassemble tous les algorithmes décris dans ce chapitre et les classe graphiquement en fonction de leurs objectifs et méthodes. La Table 1.2 résume en plus le principe de chaque solution.

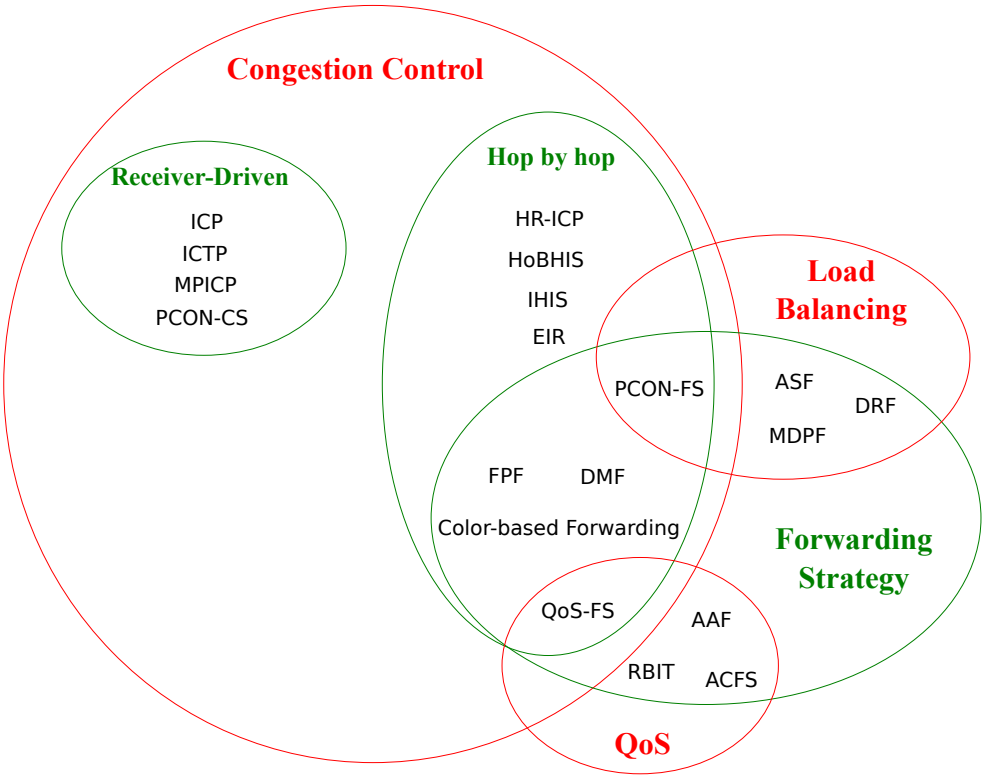


FIGURE 1.11 Catégorisation des protocoles présentés

Les techniques les plus citées et populaires reprennent malheureusement les codes du monde IP. Si cela est, bien sûr, un bon point de départ, nous prenons le parti dans cette thèse de changer de point de vue et de profiter de ce nouveau paradigme pour répondre à ces problématiques tout en utilisant au mieux les nouvelles opportunités fournies par NDN.

TABLE 1.2 Résumé des protocoles présentés.

Sigle	Lieu	Objectifs	Principe	Paramètres
ICP	Consommateur	Contrôle de congestion	Fenêtre de congestion	RTT
ICTP	Consommateur	Contrôle de congestion	Fenêtre de congestion	RTT
MPICP	Consommateur	Contrôle de congestion	Fenêtre de congestion	RTT et label de route
PTP	Consommateur	Contrôle de congestion	Fenêtre de congestion	RTT et label de route
HR-ICP	Nœud et Consommateur	Contrôle de congestion	Crédit par interface	RTT
EIR	Nœud et Consommateur	Contrôle de congestion	Notifications explicites	RTT
PCON	Nœud et Consommateur	Contrôle de congestion et Répartition de charge	Fenêtre de congestion et Notifications explicites	RTT et temps de séjour dans files d'attente
HoBHIS	Nœud	Contrôle de congestion	Interest Shaping	RTT
IHIS	Nœud	Contrôle de congestion	Interest Shaping	RTT
CbF	Nœud	Contrôle de congestion	Interest Rate Limit	Nombre de PI
FPF	Nœud	Contrôle de congestion	Interest Rate Limit	Nombre de PI et RTT
DMF	Nœud	Contrôle de congestion	Interest Rate Limit	Nombre de PI, RTT et bande passante
QoS-FS	Nœud	Contrôle de congestion et Gestion de la QoS	Paramètres de QoS dans Interest et monitoring des interfaces	RTT et bande passante
AAF	Nœud	Gestion de la QoS	Classes de Service et monitoring des interfaces	RTT
ACFS	Nœud	Gestion de la QoS	Colonies de Fourmis	Phéromones
RBIT	Nœud	Gestion de la QoS	Mesure de capacité par l'utilisateur	Bande passante
ASF	Nœud	Répartition de charge	Utiliser l'interface avec le plus petit délai	RTT
DRF	Nœud	Répartition de charge	Utiliser l'interface la moins utilisée	Nombre de PI
MDPF	Nœud	Répartition de charge	Calcul d'un score par interface à l'aide d'une matrice de décision	Nombre de PI et RTT

Chapitre 2

Définitions et Formulation du problème

Sommaire

2.1 Introduction

Lorsque l'on s'intéresse à la Qualité d'Expérience (QoE) des utilisateurs au niveau du réseau dans sa globalité, il se pose très vite la question de comment différencier les données qui transitent dans le réseau. Comme nous allons le voir dans la première section, la notion de flux est difficile à définir dans NDN. Dans un second temps nous analysons différents cas où le mot flux pourrait avoir du sens. Puis nous proposons dans une troisième partie une définition d'un flux en accord avec nos objectifs. Enfin une fois les notions clairement définies, nous formulons notre problématique que nous décrivons ensuite formellement au travers d'un modèle. Celui-ci sera utilisé dans la suite de nos travaux pour proposer des réponses aux différentes questions soulevées mais également pour déterminer les valeurs optimales aux métriques observées.

2.2 Problématique de la définition d'un flux

Dans le modèle *host-centric* d'IP, la notion de flux a du également être définie et l'est généralement comme étant une communication entre les applications d'un client et d'un serveur. Il est caractérisé par un *5-tuple* qui contient les adresses IP des deux hôtes (source et destination), le protocole de transport utilisé (TCP ou UDP) et les ports des deux applications communicantes. Ce *5-tuple* (Figure 2.1) permet d'identifier un flux d'une communication de manière unique dans le réseau et les cinq champs sont disponibles directement dans le paquet.

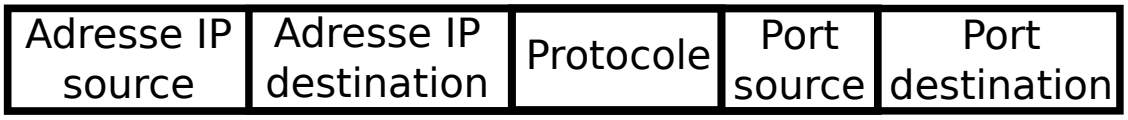


FIGURE 2.1 5-tuple permettant d'identifier un flux dans IP

Dans le modèle *content-centric* de NDN, le problème est plus complexe. D'une part, l'application d'un client peut ici récupérer le contenu souhaité depuis potentiellement plusieurs serveurs. Nous qualifierons ce comportement de *multi-source*. D'autre part, pour un serveur donné, plusieurs clients peuvent demander le même contenu en même temps, comportement que nous appelons *multi-destination*.

De plus, toutes les entités du réseau ne sont pas forcément conscientes de tous les acteurs d'un flux. Un consommateur, qui est l'une des extrémités du flux, ne connaît a priori ni le nombre, ni l'identité des producteurs qui le servent. Actuellement, un producteur n'a pas de traitement particulier à faire sur la gestion des flux et se contente simplement

de répondre aux Interests qui arrivent en envoyant le paquet Data correspondant. Mais dans de futures propositions, il pourrait avoir besoin de savoir lui aussi qui participe à un flux, ou a minima de les différentier, et dans quelle mesure. Finalement, du point de vue d'un nœud du réseau, où la *forwarding strategy* a potentiellement besoin de gérer les flux différemment, seul le préfixe du contenu demandé est connu. Le nœud n'est donc actuellement pas capable de savoir combien il y a de consommateurs qui demandent ce contenu, ni combien de producteurs le fournissent.

En outre, comme un contenu a besoin d'être morcelé en différents paquets Data, chacun de ces paquets (et les Interests correspondant) contient le nom complet du morceau. Le préfixe (ou nom du contenu) est alors déterminé grâce à la méthode *Longuest Prefix Match* (LPM), comme vu dans la section 1.3. Un problème supplémentaire survient alors puisque les entrées de la FIB peuvent avoir été agrégées (dans le but de réduire la taille de celle-ci). Le LPM peut donc renvoyer le même préfixe pour deux contenus différents (mais proche sémantiquement et potentiellement de la même provenance). Deux flux peuvent donc être identifiés comme un flux unique par un nœud mais comme deux flux distincts par un autre.

Pour résumer, plusieurs notions de flux sont donc envisageables dans NDN : unicité du consommateur, unicité du producteur, unicité du chemin, unicité du contenu (mais sujet à interprétation par les nœuds). Avant de proposer une définition pour un flux, il nous faut donc comprendre quels sont les avantages et inconvénients de chacune de ces notions et quels sont nos besoins. Dans la suite de ce chapitre, nous allons donc expliciter les cas envisagés avec des exemples simples puis définir un concept de flux qui réponde à nos besoins.

2.2.1 1 consommateur, 1 producteur et pas de multi-chemin

Le cas le plus simple est celui dans lequel un seul consommateur demande un contenu qui n'est servi que par un seul producteur et via un seul chemin (Figure 2.2). Dans ce cas-là, il n'y a pas d'ambiguïté et chaque flux peut être identifié de façon unique par un préfixe, cette dernière notion pouvant donc être utilisée pour définir celle de flux. Malheureusement, un système réel est sensiblement plus complexe.

FIGURE 2.2 Cas Simple

2.2.2 Multi-chemins

Dans ce cas de figure, nous avons toujours un consommateur et un producteur (Figure 2.3). Cependant le contenu est récupéré simultanément par plusieurs chemins entre ces deux acteurs. Deux définitions sont ici possibles :

— le flux est l'ensemble des données acheminées au travers d'un chemin donné ;

— le flux est l'ensemble des données acheminées sur l'ensemble des chemins empruntés.

Comme chacun des chemins participe au débit total reçu par l'utilisateur et donc à sa QoE, nous décidons d'utiliser la deuxième possibilité. En complément, nous appellerons sous-flux l'ensemble de données acheminé au travers d'un chemin unique. Un flux est donc un ensemble de sous-flux (réduit à un seul sous-flux dans le cas décrit en 2.2.1).

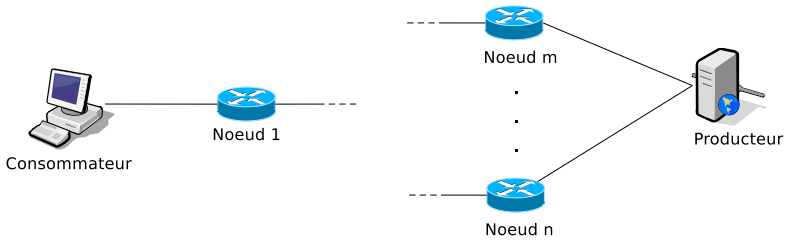


FIGURE 2.3 Cas Multi-Chemins

2.2.3 Multi-producteurs

Imaginons maintenant que plusieurs producteurs transmettent le même contenu vers un même consommateur (Figure 2.4). Deux définitions sont à nouveau possibles ici :

— chaque paire consommateur-producteur représente un flux distinct ;

— le flux est l'ensemble des données acheminées entre le consommateur et les différents producteurs.

Comme pour les multi-chemins, chaque producteur participe à améliorer le débit reçu par le consommateur. Nous décidons donc qu'un flux englobe tous les paquets concernant le même contenu, peu importe sa provenance, puisque chacun contribue à la QoE finale de l'utilisateur. Comme précédemment, un échange particulier avec un unique producteur pourra être appelé sous-flux.

2.2.4 Multi-consommateurs

Nous nous plaçons maintenant dans le cas où plusieurs consommateurs demandent le même contenu et sont servis par un unique producteur (Figure 2.5). Deux points de vue sont encore possibles :

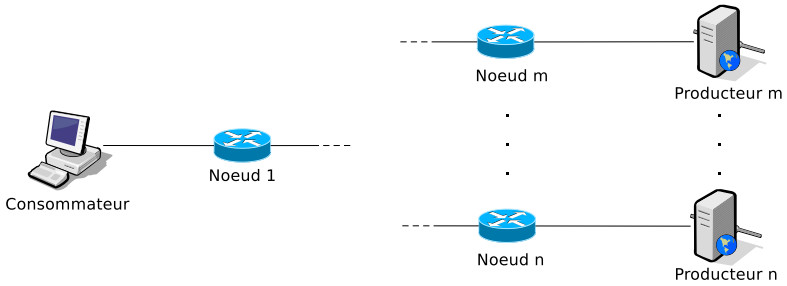


FIGURE 2.4 Cas Multi-producteurs

— chaque paire de consommateur-producteur représente un flux distinct ;
— le flux est l'ensemble des données acheminées entre les différents consommateurs et le producteur.

Contrairement aux cas précédents, nous décidons ici de différentier les flux de chaque consommateur. En effet, chacun des consommateurs possède ses besoins et contraintes propres. Nous décidons de prendre le point de vue des utilisateurs et la première solution est donc, ici, retenue.

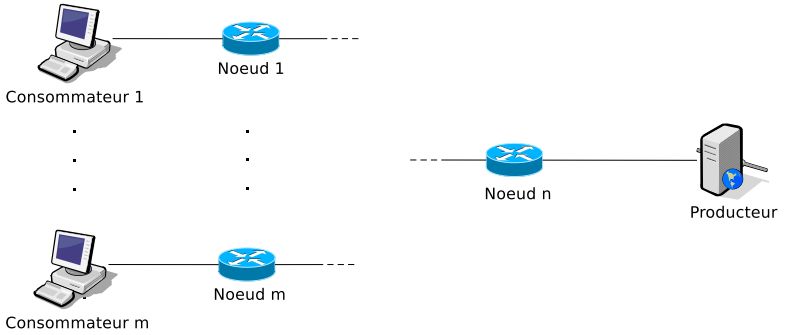


FIGURE 2.5 Cas Multi-consommateurs

2.2.5 Agrégation de flux

Dans ce dernier cas, nous mettons en lumière la difficulté actuelle pour les nœuds NDN de différentier deux contenus dont le nom est proche (Figure 2.6). Sur les nœuds, la taille de la FIB peut être un problème et, comme pour IP, de l'agrégation d'entrées est mise en place pour en limiter les effets. La méthode *Longuest Prefix Match* est originellement conçue pour trouver la meilleure entrée dans la FIB pour le relayage des Interests. Le préfixe obtenu est alors classiquement utilisé pour identifier et différentier les flux dans les méthodes de contrôle de congestion de la littérature (non canonique). Dans

l'exemple illustré par la Figure 2.6, le producteur A propose la donnée "/exemple/A" et le producteur B la donnée "/exemple/B". Le nœud 2 aura deux entrées distinctes pointant chacune vers l'interface emmenant au contenu recherché. Le nœud 1, cependant, a potentiellement agrégé les deux entrées puisqu'elles ont le préfixe "/exemple" en commun et pointent toutes deux sur l'interface emmenant au nœud 2. Dans ce cas là, le nœud 1 identifiera les demandes vers les deux contenus comme étant le même flux. Pour qu'il n'y ait pas de problème, nous souhaitons donc que la notion de flux que nous allons définir permette à chacun des nœuds du réseau de différentier les flux sans équivoque. Dans notre exemple, nous souhaitons que les nœuds 1 et 2 soient capables d'identifier de la même manière les flux, et de façon indépendante.

FIGURE 2.6 Cas deux contenus

2.3 Définition d'un flux

Sur la base de ces observations, nous proposons une définition de la notion de flux et de sous-flux qui sera utilisée tout au long de ce manuscrit.

2.3.1 Définitions

Pour résumer, pour notre notion de flux, nous voulons ne considérer qu'un seul consommateur car notre but va être d'améliorer sa QoE. Nous voulons aussi qu'un flux soit identifié de la même manière partout dans le réseau, pour qu'il n'y ai pas de problème lors de la gestion de ceux-ci. Nous acceptons donc qu'un flux emprunte plusieurs chemins et soit alimenté par plusieurs sources. Par conséquent, nous définissons un flux dans NDN de la manière suivante :

Définition : Dans NDN, un flux est une suite de paquets (Interests et Data) transmis entre un consommateur et un ou plusieurs producteurs et constituant un contenu unique. Il peut emprunter plusieurs chemins et doit pouvoir être identifié sans ambiguïté par chaque nœud du réseau qu'il emprunte.

Par souci de clarification, nous définissons aussi les termes "chemin" et "sous-flux" comme suit :

Définition : Dans NDN, un chemin représente un ensemble de nœuds qui sont reliés un à un et dont les deux extrémités sont un consommateur et un producteur.

Définition : Dans NDN, un sous-flux représente une partie d'un flux qui ne prend qu'un unique chemin.

Nous pouvons voir sur la Figure 2.7 qu'il y a trois flux, un premier concernant le contenu "/exemple/A" et le consommateur 1, en rouge, un deuxième concernant le contenu "/exemple/B" et le consommateur 1, en bleu et un dernier concernant à nouveau le contenu "exemple/B" mais avec le consommateur 2, en vert. Pour le premier flux, il y a deux chemins disponibles et donc deux sous-flux. Les deuxième et troisième flux ne sont composés que d'un seul sous-flux, car il n'y a qu'un chemin entre leur consommateur respectif et le producteur B.

2.3.2 Identification des flux

La définition que nous proposons pour la notion de flux permet donc une prise en compte des besoins de qualité de service de chaque consommateur par chacun des nœuds du réseau. Cependant, pour pouvoir mettre en place une solution capable d'assurer une telle qualité de service, il est nécessaire que cette identification des flux puisse être réalisée

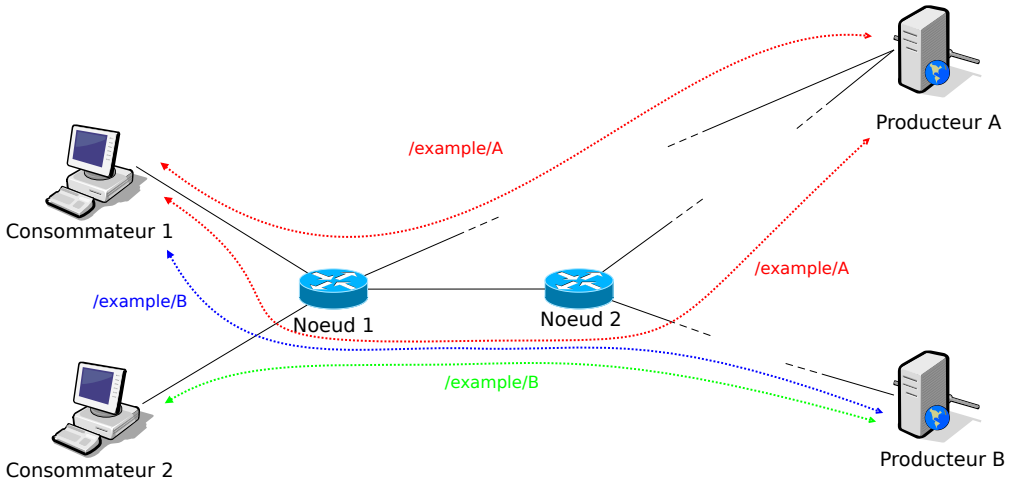
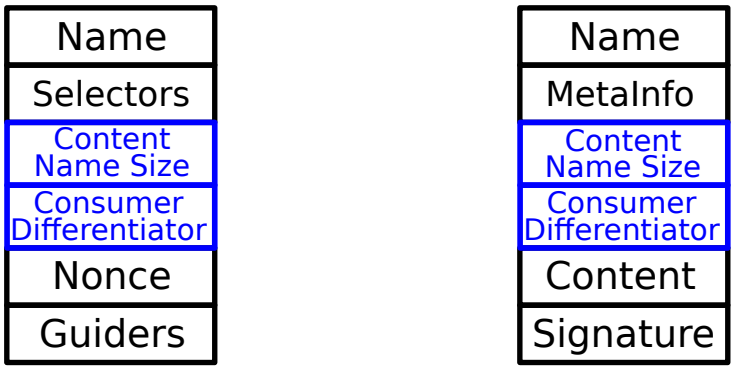


FIGURE 2.7 Trois flux pour quatre sous-flux

par les différents éléments du système. Tel quel, NDN ne permet pas cela. Avec le LPM, les nœuds sont seulement capables de trouver un préfixe du nom de contenu. Ils ne sont pas capables de savoir si celui-ci correspond au nom complet ou non. Nous proposons donc de rajouter un champ dans les Interests et les Data indiquant la taille du nom du contenu, appelé "ContentNameSize". Ce champ sera initialisé par le consommateur et restera inchangé jusqu'aux producteurs. Grâce à ce nouveau champ et au nom déjà contenu dans les paquets NDN, chaque nœud du réseau pourra donc identifier de manière unique et invariante le nom du contenu. Ils utiliseront toujours le LPM mais seulement pour la partie routage et relayage des Interests.

Il faut maintenant que les nœuds soient capables de différentier deux consommateurs. Tel quel, les acteurs de bout en bout (consommateurs ou producteurs) sont anonymes dans NDN. C'est un choix de conception qui ne nous semble pas raisonnable de remettre en question car nous sommes dans un modèle centré sur le contenu. Nous allons cependant devoir concéder un peu sur cet aspect sans que cela ne change fondamentalement les motivations des ICNs. En effet, nous proposons d'ajouter un nouveau champ dans les paquets NDN, appelé "ConsumerDifferentiator", qui a pour but de différentier plusieurs utilisateurs demandant le même contenu. Ce nouveau champ est généré aléatoirement par le consommateur pour chaque contenu demandé et ne permet toujours pas d'identifier et de tracer l'utilisateur sur le réseau. Nous conservons donc la propriété d'anonymat de NDN. Par exemple, si un consommateur demande deux contenus différents, il générera aléatoirement deux "ConsumerDifferentiator" différents. Une entité malveillante ne peut donc pas déduire du champ "ConsumerDifferentiator" l'identité du consommateur.



(a) Paquet Interest (b) Paquet Data

FIGURE 2.8 Format des paquets avec nos ajouts

En résumé, nous proposons de rajouter deux nouveaux champs dans les paquets Interests :

— le premier qui permet de déterminer le nom du contenu recherché (avec l'aide du nom du paquet Data demandé) ;
— le second qui permet de différentier plusieurs utilisateurs qui demanderaient le même contenu.

Avec ces deux modifications, les nœuds du réseau sont maintenant capables d'identifier et différentier les flux dans NDN tel que nous les avons définis et sans ambiguïté. De plus, ces ajouts se font très facilement grâce au schéma d'encodage TLV (pour Type-Length-Value) utilisé dans NDN et ne représentent qu'un faible surcoût sur la taille des entêtes des paquets.

2.4 Formulation du problème

2.4.1 Notations

La table 2.1 résume toutes les notations que nous allons utiliser dans la suite de ce chapitre. Nous les décrirons plus en détail par la suite.

Nous considérerons un système dans lequel circulent K flux, identifiés par un indice $k \in [\![1, K]\!]$. Nous nous intéresserons au débit F^k du flux k, qui peut être décomposé de la manière suivante :

$$F^k = \sum_{l=1}^{L^k} F^{k,l} \tag{2.1}$$

L'Équation (2.1) indique que le débit du flux k correspond à la somme des contributions

TABLE 2.1 Liste des Notations.

Notation	Description
K	Nombre de flux
F^k	Débit du flux k
C^k	Consommateur du flux k
L^k	Nombre de producteurs du flux k
P_l^k	Producteur l du flux k
$F^{k,l}$	Débit du flux k fourni par le producteur P_l^k
$f_{i,j}^k$	Débit du flux k entre les nœuds i et j
$M^{k,l}$	Nombre de multi-chemin pour le flux k fourni par le producteur P_l^k
$c_m^{k,l}$	Chemin m amenant au producteur l du flux k
M^k	Nombre total de chemins pour le flux k
$F^{k,l,m}$	Débit du sous-flux sur le chemin m amenant au producteur l du flux k
$u_{i,j}$	Capacité du lien entre les nœuds i et j
V	Ensemble des nœuds du réseau
E	Ensemble des liens du réseau
min_t^*	$t^{ème}$ plus petit débit dans la répartition équitable

de chaque producteur l. Dans l'exemple de la Figure 2.9, nous avons un flux demandé par le consommateur C^1, et servi par deux producteurs, P_1^1 et P_2^1. Nous avons donc, dans cet exemple, le nombre de flux $K = 1$, le nombre de producteur de ce flux $L^1 = 2$, le nombre de chemins pour le premier producteur $M^{1,1} = 2$ et le nombre de chemins pour le deuxième producteur $M^{1,2} = 1$. Il y a, au total, trois chemins et donc trois sous-flux :

— le chemin $c_1^{1,1}$, reliant le consommateur au producteur P_1^1 en passant par le nœud 1,

— le chemin $c_2^{1,1}$, reliant le consommateur au producteur P_1^1 en passant par les nœuds 1 et 2,

— le chemin $c_1^{1,2}$, reliant le consommateur au producteur P_2^1 en passant par les nœuds 1 et 2.

Le débit global du flux 1, celui reçu par le consommateur C^1, est la somme des débits des trois sous-flux ($F^1 = F^{1,1,1} + F^{1,1,2} + F^{1,2,1}$). Le débit d'un flux k fourni par le producteur P_l^k peut, lui aussi, être décomposé :

$$F^{k,l} = \sum_{m=1}^{M^{k,l}} F^{k,l,m} \tag{2.2}$$

Dans l'exemple de la Figure 2.9, le producteur P_1^1 alimente le flux par l'intermédiaire

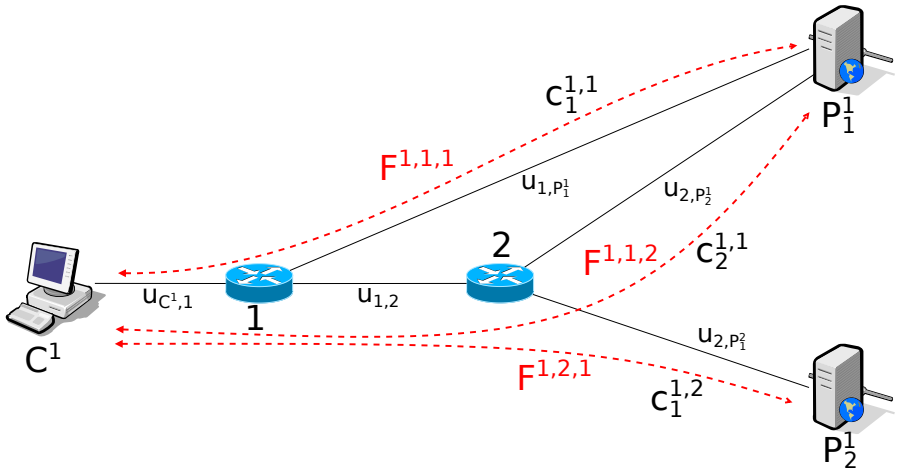


FIGURE 2.9 Exemple concret avec un flux

de deux chemins. Dans ce cas, l'Équation (2.2) est donc composée de deux éléments : $F^{1,1} = F^{1,1,1} + F^{1,1,2}$. Le producteur P_2^1 n'alimente le flux que par l'intermédiaire d'un chemin. Dans ce cas, l'Équation (2.2) est donc composée d'un seul élément : $F^{1,2} = F^{1,2,1}$. Le débit du flux k sur un lien est la somme des débits des sous-flux présents sur ce lien. Par exemple :

- sur le lien entre 2 et P_1^1, il n'y a qu'un seul sous-flux présent, donc $f^1_{2,P_1^1} = F^{1,1,2}$,
- sur le lien entre 1 et 2, il y a deux sous-flux présents, donc $f^1_{1,2} = F^{1,1,2} + F^{1,2,1}$,
- sur le lien entre C^1 et 1, il y a les trois sous-flux présents, donc $f^1_{C^1,1} = F^{1,1,1} + F^{1,1,2} + F^{1,2,1}$.

2.4.2 Formulation et Formalisation

Depuis le début, nous nous sommes toujours intéressés à la Qualité d'Expérience des utilisateurs finaux. Nous aurions pu préférer le point de vue des opérateurs de réseau, prioriser la réduction de leurs coûts et axer notre étude sur les aspects associés. Ce n'est donc pas le choix que nous avons fait et nous avons opté pour l'étude de l'amélioration de la QoE des utilisateurs dans NDN. Dans cette optique, nous avons choisi d'utiliser le critère du débit final obtenu par l'utilisateur pour quantifier sa QoE. Ce choix, bien que potentiellement simpliste, résulte du constat que le nouveau paradigme d'ICN aura a priori du mal à complètement remplacer l'actuel paradigme accès sur l'hôte. Nous sommes donc partis du principe que la couche NDN, pour pouvoir exister au côté d'IP, se doit d'exceller dans ses domaines de prédilections. La récupération d'un contenu via plusieurs sources et/ou chemins est donc le domaine où NDN doit avoir des performances

qui justifient son introduction. Comme de nombreux utilisateurs peuvent utiliser le réseau simultanément, et que celui-ci se doit d'être neutre (nous n'avons pas considéré de mettre en place des classes de service), nous nous sommes aussi intéressés à la notion d'équité entre ces utilisateurs. Même dans le cas où le réseau proposerait plusieurs classes de service, la notion d'équité serait malgré tout à considérer lors de la confrontation de plusieurs flux d'une même classe de service. C'est pourquoi nous avons décidé d'utiliser le critère d'équité Max-Min [65]. Cela nous permet de donner autant à chaque flux, mais si le réseau le permet, donner plus aux flux pouvant obtenir plus. Ce critère consiste à atteindre un équilibre de Pareto dans lequel l'augmentation du débit d'un flux provoquerait la diminution du débit d'un flux déjà plus petit que celui que nous augmentons. Chaque flux serait donc contraint par un goulot d'étranglement sur chacun de ses chemins. Finalement, notre problématique peut être formulée comme suit : répartition équitable des flux dans un réseau de contenu utilisant les multi-chemins.

Dans [6], nous formalisions notre problématique avec deux problèmes d'optimisation successifs. Le premier était le *Maximum Multi-Commodity Flow problem* [66] qui permet de maximiser la somme des débits des utilisateurs du réseau. Le second introduisait de l'équité et permettait de trouver le débit minimal que chaque utilisateur doit obtenir pour que la distribution soit équitable. Cependant, la solution au *Maximum Multi-Commodity Flow problem*, même s'il donne bien un équilibre de Pareto, n'est pas obligatoirement compatible avec celui donné par l'équité Max-Min. Le contre-exemple (Figure 2.10) [65] suivant le démontre facilement :

En supposant que les liens limitants sont les liens entre les Nœuds 1, 2 et 3, une répartition qui maximise l'utilisation du réseau est obtenue lorsque :

— le flux A dispose d'un débit $F^A = 0Mbps$,
— le flux B dispose d'un débit $F^B = 10Mbps$,
— et le flux C dispose d'un débit $F^C = 10Mbps$.

Ainsi, l'utilisation du réseau est de $20Mbps$ et est maximale. Cette distribution est évidemment non équitable puisque le flux A n'obtient pas de débit. La distribution équitable est la suivante : $F^A = F^B = F^C = 5Mbps$. Les deux liens sont toujours utilisés entièrement mais la somme des débits de tous les utilisateurs n'est ici pas maximale (seulement $15Mbps$ au total).

Nous cherchons donc bien une répartition qui soit un équilibre de Pareto (l'augmentation du débit d'un flux ne peut se faire qu'au détriment d'un autre flux), mais pas spécialement une de celles qui maximisent la somme des débits comme dans le *Maximum Multi-Commodity Flow problem*. Nous cherchons à ce que l'ensemble des chemins de nos flux soient saturés sur au moins un lien (ce qui garanti l'équilibre de Pareto et

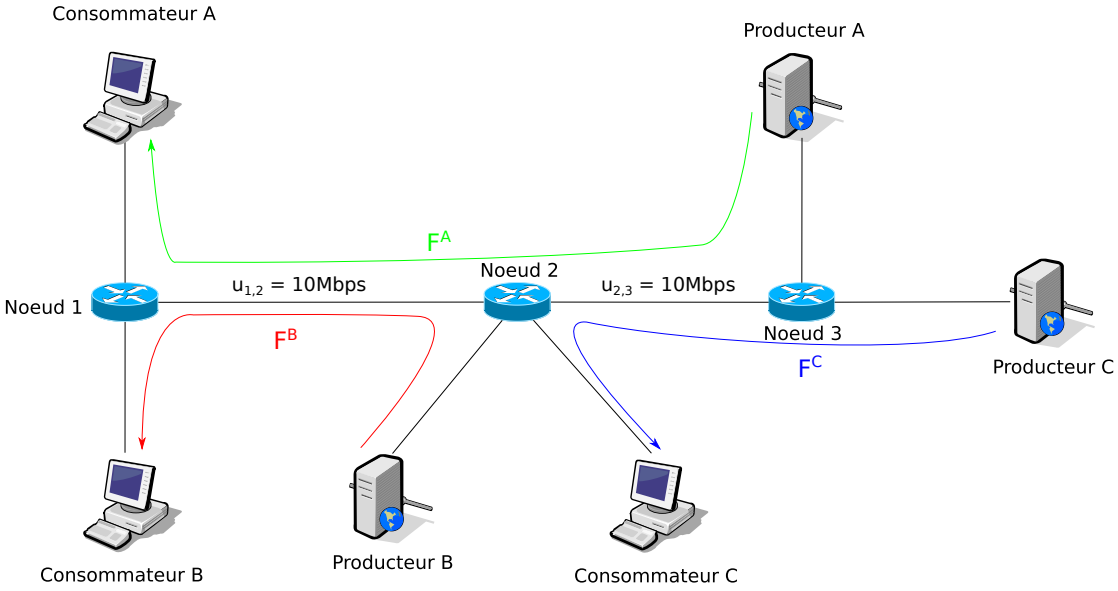


FIGURE 2.10 Exemple où maximiser l'utilisation du réseau est différent de l'équité Max-Min

une utilisation efficace du réseau) et que l'augmentation du débit d'un flux implique la diminution de celui d'un autre flux alors qu'il est déjà inférieur au flux augmenté.

De plus, dans notre formalisation de [6], notre deuxième problème d'optimisation nous permettait seulement de trouver le premier minimum de la distribution (le plus petit débit). Nous cherchons en réalité, pas seulement le minimum des débits mais tous les débits. C'est pourquoi, nous formalisons finalement ça en N problèmes d'optimisation successifs. Chacun a pour but de trouver le prochain minimum parmi les flux dont le débit peut encore être augmenté. C'est-à-dire que le $t^{ème}$ problème d'optimisation permet de trouver le $t^{ème}$ minimum sachant que les $t-1$ plus bas débits ont déjà été trouvés. Avec nos notations, le $t^{ème}$ problème d'optimisation peut être formulé avec l'objectif suivant :

$$\max min_t \tag{2.3}$$

et avec les contraintes suivantes :

$$\sum_{j \in V} f_{i,j}^k - \sum_{j \in V} f_{j,i}^k = \begin{cases} F^k & i = C^k \\ 0 & i \neq C^k, P_l^k \\ -F^{k,l} & i = P_l^k \end{cases}, \forall k = 1, \ldots, K, \forall l = 1, \ldots, L_k \tag{2.4}$$

$$\sum_k f_{i,j}^k \leq u_{i,j}, \forall (i,j) \in E \tag{2.5}$$

$$F^k = min^*_{g(k)}, \forall k \in K_{t-1} \tag{2.6}$$

$$F^k - min_t \geq 0, \forall k \in [\![1, K]\!] \setminus K_{t-1} \tag{2.7}$$

Les contraintes (2.4) représentent les contraintes de conservation de flux. Les contraintes (2.5) représentent le respect des capacités de chaque lien du réseau. Les contraintes (2.6) représentent les $t-1$ premiers minimums trouvés lors des précédentes itérations. La fonction $g(k)$ renvoie, pour le flux k, l'indice du minimum qui lui correspond. En d'autres termes, il classe les flux par débits croissants. K_{t-1} est l'ensemble des indices des flux possédant les $t-1$ plus bas débits trouvés lors des itérations précédentes. Les contraintes (2.7) permettent à min_t de représenter le minimum des débits des flux restants (il est, en effet, inférieur ou égal à tous ces débits).

Une fois le problème résolu, nous obtenons min^*_t qui représente le plus grand minimum que l'ensemble des flux considérés doit vérifier pour que la distribution soit équitable. Il nous suffit de résoudre ce problème d'optimisation successivement pour trouver tous les débits des flux et ainsi avoir la distribution qui correspond au critère d'équité Max-Min. Nous nous servons dans la suite de cette distribution optimale pour comparer les différentes solutions, notamment celle que nous allons introduire dans le prochain chapitre.

La Figure 2.11 reprend l'exemple précédent en modifiant le débit entre les nœuds 1 et 2, le faisant passer à $100Mbps$, et celui entre les nœuds 3 et Producteur C, le faisant passer à $2Mbps$. Une première itération de notre problème d'optimisation nous donne $min_1 = 2Mbps$, qui correspond au débit du flux du consommateur C (F^C). Lors de celle-ci, les contraintes (2.6) n'existent pas (aucun minimum n'a déjà été trouvé) et $K0$ est vide. Pour la seconde itération, les contraintes (2.6) ne sont constituées que d'une équation : $F^C = min_1$ et $K_1 = \{C\}$. Celle-ci nous donne $min_2 = 8Mbps$, qui correspond au débit du flux du consommateur A (F^A). Finalement, pour la troisième et dernière itération, les contraintes (2.6) sont constituées de deux équations : $F^C = min_1$ et $F^A = min_2$ et $K_2 = \{A, C\}$. Elle nous donne $min_3 = 92Mbps$, qui correspond au débit du flux du consommateur B (F^B).

2.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la définition d'un flux dans NDN. En effet, celui-ci n'est pas toujours clairement défini dans la littérature et les nombreuses interprétations possibles peuvent induire en erreur un lecteur et rendre le discours ambigu. Après avoir étudié les différents cas possibles, nous avons proposé une définition où un

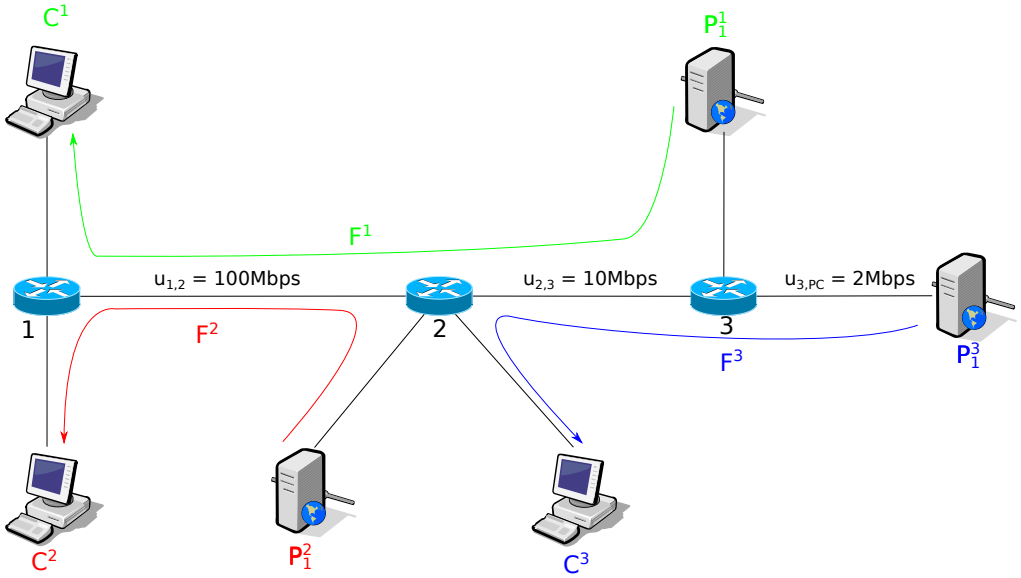


FIGURE 2.11 Exemple pour notre formulation

flux est caractérisé par le contenu demandé et le consommateur qui le demande. Nous avons ensuite proposé l'utilisation de deux nouveaux champs dans les paquets Interest et Data permettant de mettre en œuvre cette définition. Avec ces deux champs, chaque entité du réseau peut dorénavant différentier deux flux sans ambiguïté. Nous conservons avec succès la propriété d'anonymat de l'utilisateur, comme initialement prévu dans NDN. Avec une définition claire de ce qu'est un flux, nous avons ensuite défini notre problématique. Nous cherchons à définir une solution qui permettra de mettre en place une distribution maximale et équitable des différents flux sur le réseau. Si le but est atteint et pour un flux donné, il ne sera pas possible d'augmenter son débit sans diminuer celui d'un autre flux moins favorisé que lui. C'est-à-dire que le second flux a globalement déjà moins de débit que le flux que l'on considère. Nous avons ensuite exprimé formellement le problème au travers de deux problèmes d'optimisation. Le premier permet de trouver l'utilisation maximale du réseau (c'est-à-dire, la valeur maximale que peut atteindre la somme des débits des flux). Le second, quant à lui, permet de trouver le débit minimal que chaque flux doit obtenir pour que la répartition soit équitable. La répartition ainsi trouvée représente l'équilibre de Pareto utilisant le critère d'équité Max-Min. Dans le prochain chapitre, nous allons définir une solution dont les résultats auront pour but de se rapprocher au mieux de cette répartition théorique.

Chapitre 3

Cooperative Congestion Control dans des topologies arborescentes

Sommaire

3.1 Introduction

Dans ce chapitre, nous définissons une solution distribuée, appelée Cooperative Congestion Control (CCC), qui a pour but de répartir équitablement les flux tout en utilisant au mieux les capacités multi-chemin de NDN. Reprenons un scénario tel que celui illustré par la Figure 3.1. Notre objectif est donc de fournir aux clients (C^1, C^2, ...) une qualité de service satisfaisante avec une utilisation optimale de l'infrastructure réseau. Cette situation nous a naturellement conduits (dans le chapitre précédent) à modéliser un tel système comme un problème d'optimisation (de la qualité de service) sous contraintes (des performances du réseau). Si une telle approche permet d'obtenir des bornes théoriques intéressantes, elle ne définit pas une implantation capable d'atteindre ces performances, ni même de s'en approcher, dans un contexte dynamique.

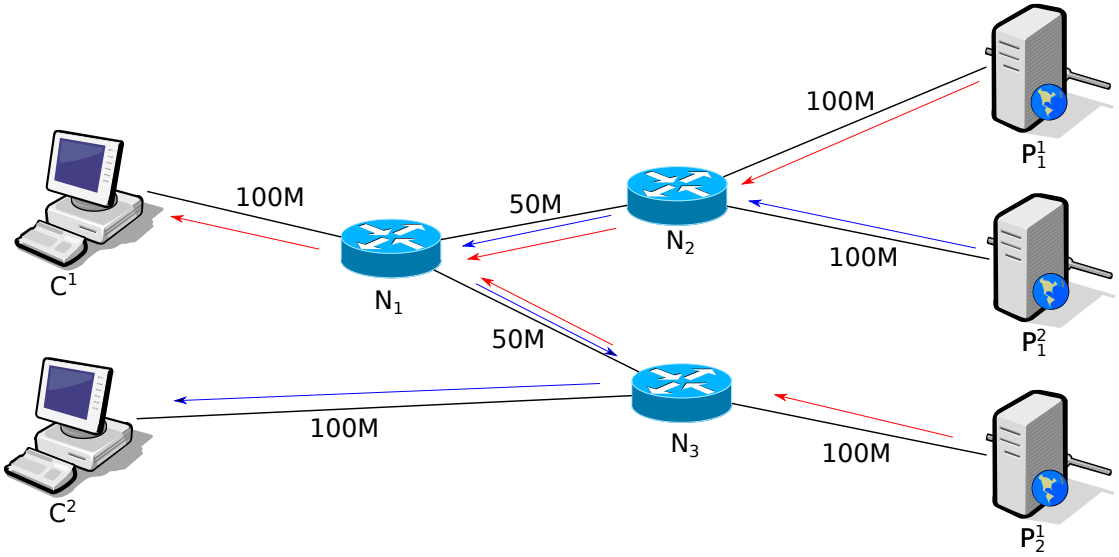


FIGURE 3.1 Topologie de test

Le but de ce chapitre est donc de proposer une architecture visant à répondre aux besoins applicatifs dans un tel contexte. Comme nous l'avons vu, des propositions existent dans la littérature. Nous pouvons distinguer celles que nous qualifions de "bout en bout" (fondées sur l'expérience de TCP) et celles qualifiées de "saut par saut" (héritières des outils d'ordonnancement dans le monde IP). Notre objectif est de profiter du meilleur de chaque approche, en proposant une solution qui englobe tous les éléments impliqués dans l'acheminement d'un flux de données.

L'expression du problème d'optimisation sous contraintes met clairement en évidence la dualité entre les objectifs de qualité d'expérience des applications et les contraintes

inhérentes à la capacité de l'infrastructure. Il est donc nécessaire que les nœuds du réseau échangent des informations leur permettant de prendre en compte ces différents paramètres. Nous proposons donc une signalisation au travers de laquelle des "objectifs" (en termes de débit) seront transmis dans le sens des Interests et des "contraintes" (sur ces mêmes débits) circuleront dans le sens contraire. Nous décrirons dans la section 3.2 cette extension du plan de contrôle.

Chaque nœud devra alors acheminer ces objectifs et ces contraintes en les adaptant conformément à une politique permettant de trouver une forme de compromis. Il devra également transmettre les données en respectant le cadre ainsi défini. Nous proposons pour cela un nombre limité d'algorithmes à implanter sur chaque nœud. Le choix des algorithmes eux-mêmes est ouvert et permet de définir une politique globale. Nous décrirons cette structure algorithmique dans la section 3.3.

La dernière section de ce chapitre sera dédiée à l'analyse des performances obtenues avec une implantation la plus simple possible des algorithmes définis.

3.2 La signalisation

Le but de cette section est donc de décrire les échanges effectués entre nœuds dans le plan de contrôle. Nous allons dans un premier temps présenter et justifier notre choix de décrire les objectifs et contraintes par le biais de "rythmes" de messages. Nous décrirons alors plus précisément ces échanges et leur notation en 3.2.2. Finalement, une brève description d'un protocole permettant ces échanges est donnée en 3.2.3.

3.2.1 Une solution "pace-based"

Le premier choix de conception que nous avons fait est de travailler avec le rythme des Interests. C'est la métrique utilisée par notre solution pour estimer l'usage d'un lien. Deux raisons nous ont poussés à utiliser cette métrique. La première est pour ne pas créer artificiellement de la congestion avec des burst de paquets Data. En effet, les paquets Interests sont beaucoup moins volumineux que les paquets Data. Avec la propriété de "flow balance" de NDN qui stipule qu'un Interest équivaut à une Data et le fait que les producteurs ne font a priori pas de contrôle de flux, une émission non contrôlée d'Interest, même si elle ne représente pas beaucoup de débit, peut créer une congestion dans le sens inverse avec les paquets de Data. Pour cette raison, chaque nœud du réseau va mettre en place du "pacing" d'Interest. Si le nœud estime que le flux A doit avoir 5Mbps de débit, il n'émettra pas les Interests à un rythme qui ferait excéder ce débit dans le sens retour.

La deuxième raison est que le rythme des Interests représente une mesure fidèle d'un débit. La taille d'un paquet de Data reste effectivement constante pour un flux. Connaissant celle-ci, il est très facile pour n'importe quelle entité d'en déduire le débit utilisé par le flux (en multipliant le rythme d'Interests par la taille des Data). D'autres métriques existent dans la littérature, comme le nombre d'Interests en attente (pour un nœud, un Interest qui a déjà été émis mais dont la Data associée n'est pas encore arrivée). Cette métrique, très utilisée dans les diverses "forwarding strategy" de la littérature, n'est en réalité pas fiable ni pour estimer un débit ni pour comparer l'utilisation d'un lien entre deux flux. En effet, pour un flux qui n'emprunte qu'un unique chemin, et en supposant les temps de traitement et d'attente dans les différentes files négligeables, le nombre d'Interest en attente ("pending interests") peut être estimé de la sorte :

$$pi\_num_{CONSUMER} = \frac{rate}{data\_size} \cdot rtt \tag{3.1}$$

Ce nombre d'Interests en attente dépend bien du débit mais aussi du temps d'aller-retour (rtt) entre le consommateur et le producteur. Dans le cas où il y a plusieurs chemins utilisés, l'Équation (3.1) représenterait la contribution d'un unique chemin et le $pi\_num_{CONSUMER}$ serait une somme sur chacun des chemins. Avec les notations du chapitre précédent (où la variable k représentant le flux n'est pas mentionnée par souci de clarté) :

$$pi\_num_{CONSUMER} = \sum_{l=1}^{L} \sum_{m=1}^{M^l} \frac{rate_{c_m^l}}{data\_size} \cdot rtt_{c_m^l} \tag{3.2}$$

On considère ici que le consommateur récupère son contenu depuis L producteurs et que pour chacun de ces producteurs, il y a M^l chemins, sachant que c_m^l est le $m$ième chemin du producteur l. La relation entre nombre d'Interests en attente et débit est encore moins évidente. Finalement, cette métrique n'est pas transposable d'un nœud à un autre, car il faut utiliser les temps d'aller-retour locaux. C'est à dire qu'un pi_num de 200 sur un nœud peut représenter un débit de 15Mbps alors que sur son voisin, qui est plus proche de la source, ce même débit sera représenté par un pi_num de 150. Il est donc, en plus, difficile pour les nœuds de se baser sur le nombre d'Interests en attente lors de leurs échanges et coopération. À l'inverse, la métrique du rythme d'Interest peut simplement être convertie en débit (en multipliant par la taille d'un paquet de Data) et est identique d'un nœud à son voisin.

3.2.2 Les échanges entre nœuds

Le deuxième choix de conception que nous avons fait est de faire coopérer les nœuds du réseau. Pour cela, ils vont s'échanger des informations sur les flux. La Figure 3.2 représente ces informations pour un flux donné, tandis que la Table 3.1 résume les notations introduites.

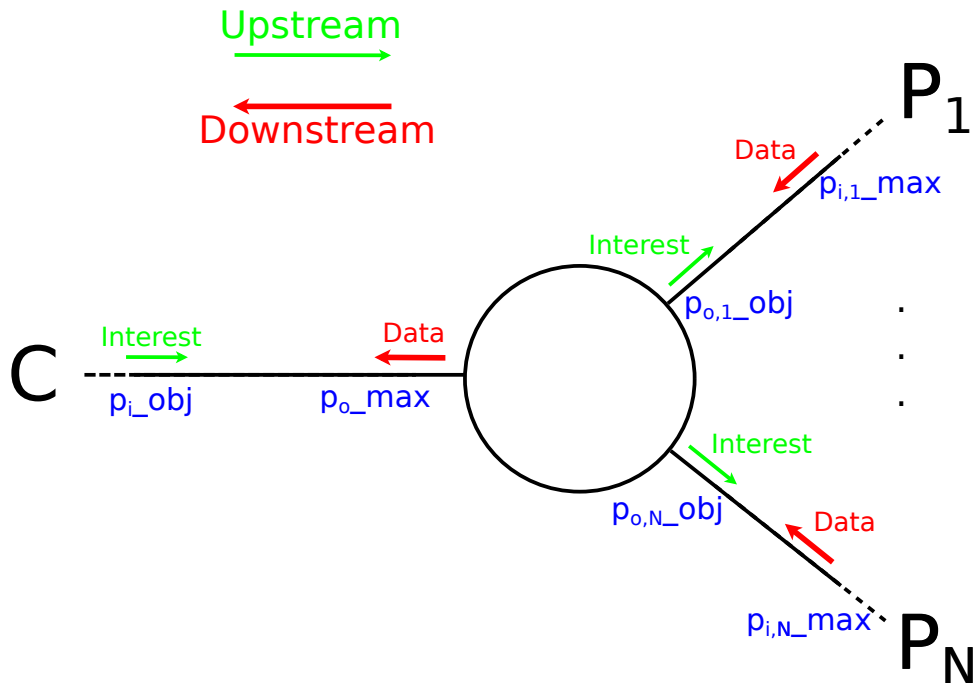


FIGURE 3.2 Signalisation de CCC du point de vue d'un nœud du réseau

La première partie de la coopération prend la forme d'objectifs exprimés par les consommateurs et transmis jusqu'aux producteurs (dans le sens montant ou "Upstream"). En prenant l'exemple de notre figure, le nœud reçoit un rythme objectif de son voisin p_i_obj ("input pace"). Le nœud indique, à son tour, l'objectif qu'il aimerait atteindre pour chacun de ses chemins en émettant des rythmes objectifs $p_{o,n}\_obj$ ("output pace").

La deuxième partie de la coopération prend la forme de contraintes exprimées depuis les producteurs et mises à jour sur chaque nœud du réseau (dans le sens descendant ou "Downstream"). Comme indiqué sur la Figure 3.2, le nœud reçoit de ses voisins des rythmes contraints $p_{i,n}\_max$. Une contrainte représente le rythme maximal auxquels doivent être émis les Interests de ce flux. À nouveau, le nœud indique à son voisin en aval la contraint qu'il devra respecter p_o_max.

TABLE 3.1 Liste des Notations de CCC.

Notation	Description
N	Nombre d'interfaces de sortie des Interests
p_i_obj	Rythme Objectif entrant
$p_i^k\_obj$	Rythme Objectif entrant pour le flux k
p_o_max	Rythme Contraint sortant
$p_o^k\_max$	Rythme Contraint sortant pour le flux k
$p_{o,n}\_obj$	Rythme Objectif sortant sur l'interface n
$p_{o,n}^k\_obj$	Rythme Objectif sortant sur l'interface n pour le flux k
$p_{i,n}\_max$	Rythme Contraint entrant sur l'interface n
$p_{i,n}^k\_max$	Rythme Contraint entrant sur l'interface n pour le flux k

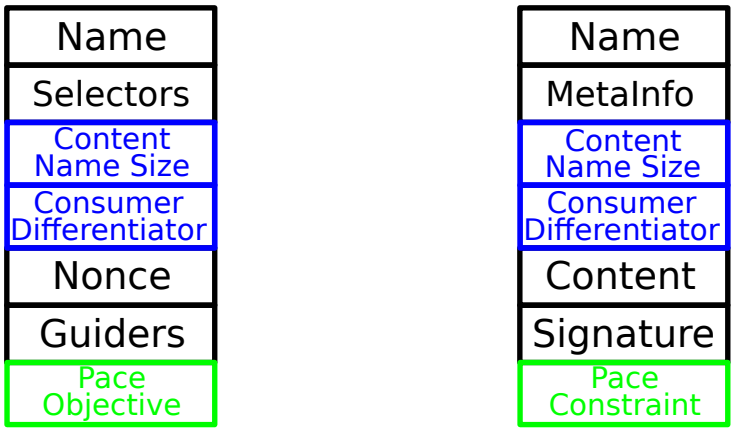
3.2.3 Le protocole

3.2.3.1 Format de la signalisation

Les paquets NDN sont encodés en utilisant le schéma TLV (pour Type-Length-Value). Si certains champs sont obligatoires, comme le nom du paquet par exemple, d'autres sont optionnels et spécifiques à des protocoles particuliers. Nous allons donc à nouveau profiter de cette particularité pour ajouter un champ dans le paquet Interest pour l'objectif et un champ dans le paquet Data pour la contrainte. Comme illustré sur la Figure 3.3, nous ajoutons le champ "Pace Objective" dans le paquet Interest qui permet au nœud en aval d'indiquer l'objectif du flux sur cette portion du réseau et le champ "Pace Constraint" dans le paquet Data qui permet au nœud en amont d'indiquer l'allocation maximale que le flux peut utiliser sur cette même portion. Ils sont tous deux indiqués en vert sur la figure. Pour mémoire, en bleu sont les deux champs ajoutés dans le chapitre précédent pour permettre aux nœuds d'identifier de manière unique les flux.

3.2.3.2 Structures de données

Dans la sous-section précédente, nous avons vu comment les informations sont transmises dans NDN. Maintenant, nous allons voir comment les nœuds conservent ces informations. Les nœuds ont déjà deux tables différentes, la Forwarding Information Table ou FIB et la Pending Interest Table ou PIT (cf 1.3). Cependant, aucune de ces deux tables ne convient pour notre besoin. En effet, la PIT contient les demandes en cours pour les paquets de Data alors que la FIB est une table de routage pour aiguiller les Interests en fonction de leur préfixe. La PIT a une granularité au paquet de Data tandis que la FIB

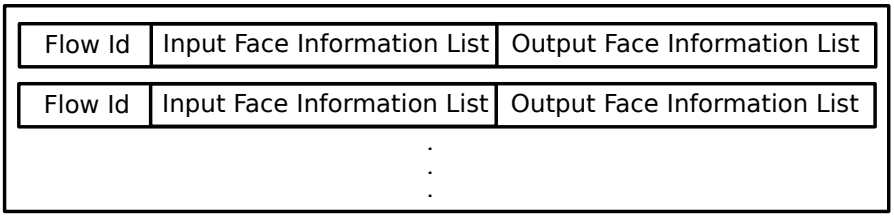


(a) Paquet Interest (b) Paquet Data

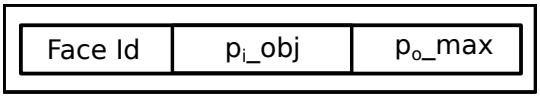
FIGURE 3.3 Format des paquets dans CCC

a la granularité d'un groupement de contenu (commençant par le même préfixe). C'est pourquoi nous introduisons la Flow Information Table ou FIT. La FIT est une table où l'entrée est un flux. Celui-ci est identifié par le nom du contenu (obtenu grâce au champ "Name" et "Content Name Size") et par le différenciateur de consommateur (champ "Consumer Differentiator"). Une entrée de cette table contient donc l'identifiant du flux ainsi que deux listes de données. La première liste contient les informations sur toutes les interfaces d'où viennent les requêtes et s'appelle l'Input Face Info List (IFIL). Une entrée de cette liste contient l'identifiant de l'interface, le rythme objectif entrant (p_i_obj) et le rythme contraint sortant (p_o_max). Dans notre cas d'étude de topologie en forme d'arbre, cette liste n'aura qu'une seule entrée, mais dans un souci d'évolutivité, nous utilisons dès maintenant une liste. La deuxième liste contient les informations sur toutes les interfaces d'où viennent les données et s'appelle l'Output Face Info List (OFIL). Une entrée de cette liste contient l'identifiant de l'interface (n), le rythme contraint entrant ($p_{i,n}\_max$) et le rythme objectif sortant ($p_{o,n}\_obj$). Une représentation des différentes structures de données est disponible sur la Figure 3.4. Lorsqu'un nouvel Interest arrive, le rythme objectif entrant (p_i_obj) est mise à jour dans l'entrée correspondante dans la IFIL. Quand un Interest est émis par le nœud sur l'interface n, le rythme objectif sortant ($p_{o,n}\_obj$) est inscrit dans le paquet. De la même manière, lorsqu'une Data arrive sur l'interface n, le rythme contraint entrant ($p_{i,n}\_max$) est mis à jour dans l'entrée correspondante dans la OFIL. Quand une Data est émise par le nœud, le rythme contraint sortant (p_o_max) est inscrit dans le paquet.

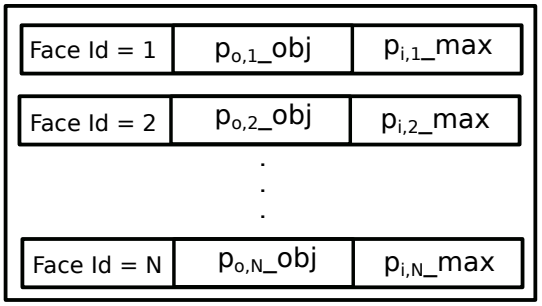
En plus de ces structures de données spécifiques à l'algorithmique de CCC, nous mettons aussi en place du *pacing* d'Interest comme expliqué en 3.2.1. Pour cela, nous



(a) Flow Information Table



(b) Input Face Information List



(c) Output Face Information List

FIGURE 3.4 Tables de CCC

utilisons une file d'Interest par flux, basée sur le principe FIFO (First In First Out). Lorsqu'un flux arrive sur le nœud, il sera inséré en bout de cette file. Pour ce flux, chaque interface présente dans sa OFIT a le droit d'émettre un Interest au plus au rythme $p_{i,n}\_max$ (pour l'interface d'indice n). Les interfaces présentes dans la OFIT vont donc venir piocher un Interest a chaque fois qu'elles ont le droit d'émettre. Si la file est vide à ce moment-là, elle retient quelles interfaces sont disponibles et leur fourni un Interest dès que possible.

3.3 Mise en œuvre sur un nœud

Maintenant que nous avons décrit les informations échangées entre nœuds, intéressons-nous à leur traitement. Comme nous l'avons dit, chaque nœud va mettre en œuvre une stratégie visant à réaliser un compromis entre objectifs et contraintes. Cette stratégie est implantée au travers de quelques algorithmes, chacun dédié à une activité spécifique. Nous allons décrire la structure globale d'une implantation de ces algorithmes puis nous préciserons le rôle exact de chaque algorithme. Nous montrerons enfin comment une stratégie spécifique peut être mise en œuvre par une conception cohérente de ces algorithmes.

Notons qu'une implantation, quels que soient les objectifs précis qu'elle vise, est

contrainte à respecter certaines propriétés. Nous définissons ainsi deux lois que devront respecter les nœuds lors du traitement des objectifs et contraintes d'un flux. La première, la loi de Conservation des Objectifs, stipule que le nœud doit émettre autant d'objectifs qu'il en a reçu pour un flux. Il n'est pas autorisé à créer ou supprimer de la demande.

Loi : Conservation des Objectifs :

$$\sum_{n=1}^{N} p_{o,n}\_obj = p_i\_obj \tag{3.3}$$

La deuxième, la loi d'Agrégation des Contraintes, indique que les nœuds doivent agréger les contraintes entrantes pour un flux. Ce n'est, ici, pas une égalité car le nœud n'est pas systématiquement capable de gérer l'agrégat de ses chemins et a donc le droit de pouvoir réduire celui-ci pour ne pas créer localement un phénomène de congestion.

Loi : Agrégation des Contraintes :

$$p_o\_max \leq \sum_{n=1}^{N} p_{i,n}\_max \tag{3.4}$$

3.3.1 Structure globale

Dans cette section, nous allons présenter la structure générale de notre solution et une première implantation des algorithmes principaux qui la compose. CCC est une solution distribuée où chaque nœud, consommateur et producteur va avoir un rôle à jouer. Les consommateurs vont gérer l'émission initiale des Interests et exprimer leur besoin applicatif. Les producteurs sont les entités les plus simples car ils devront simplement répondre à chaque Interest. C'est dans les nœuds que l'intelligence sera la plus grande car ils vont devoir gérer les objectifs et contraintes entrants et déterminer les objectifs et contraintes sortants. Chaque acteur se devra naturellement de respecter les contraintes qu'ils reçoivent lors de l'émission des Interests d'un flux. CCC s'articule autour de trois principes :

— la coopération,
— la supervision,
— la répartition.

Le principe de coopération est représenté par les échanges d'objectifs et de contraintes entre chaque nœud (décrit en section 3.2.2). Le principe de supervision consiste à assurer que tout se passe bien localement. De manière périodique, chaque nœud du réseau estime si un phénomène de congestion est en cours ou si au contraire, la capacité d'un lien est sous-utilisée.

Finalement, le principe de répartition est mis en œuvre par les algorithmes qui sont en charge de répartir les objectifs et les contraintes. Avec les informations récoltées par les deux principes précédents, les nœuds doivent maintenant décider de la répartition de la bande passante entre les flux et leurs différents chemins. Dans le sens montant, les nœuds décident de la distribution de l'objectif entrant sur l'ensemble des chemins disponibles. Dans le sens descendant, les nœuds diminuent ou augmentent les contraintes sortantes de chaque flux en fonction du retour de la supervision (et en respectant la loi d'Agrégation des Contraintes, Équation (3.4)). C'est aussi dans cette partie qu'est implantée la stratégie d'allocation (équité entre flux, économie des ressources, ...). Nous avons cherché à définir une solution générique et modulaire dont chaque bloc peut être implanté indépendamment des autres. Cela nous permet à la fois de proposer des améliorations via de nouvelles implantations mais aussi de pouvoir changer la stratégie d'allocation au gré de nos besoins. Les algorithmes sont activés par les arrivées de d'Interest (section 3.3.2) et de Data (section 3.3.3), mais aussi de façon périodique (section 3.3.4). Nous allons présenter des algorithmes simples, qui vont définir une stratégie d'allocation des ressources proche de l'équité Max-Min. Nous verrons en section 3.3.5 comment une stratégie différente peut être mise en place sans remettre en cause l'architecture.

3.3.2 Arrivée d'un Interest

La Figure 3.5 présente les mécanismes mis en place lors de l'arrivée d'un Interest. Lors de l'arrivée d'un Interest, le nœud va au préalable créer une entrée dans sa FIT s'il appartient à un nouveau flux et y mettre à jour l'objectif entrant nouvellement reçu. Notre premier algorithme, la distribution de l'objectif, entre alors en jeu et a pour rôle, comme son nom l'indique, de distribuer cet objectif sur l'ensemble des interfaces de sortie. L'Algorithme 3.1 est l'implantation dans CCC de cet algorithme et va simplement donner une part égale de l'objectif entrant à chacune des interfaces. L'Interest est ensuite inséré dans la file d'émission d'Interest du flux. Ce n'est qu'au moment de l'émission de l'Interest, quand l'interface de sortie sera déterminée, que le champ "Pace Objective" du paquet sera mise à

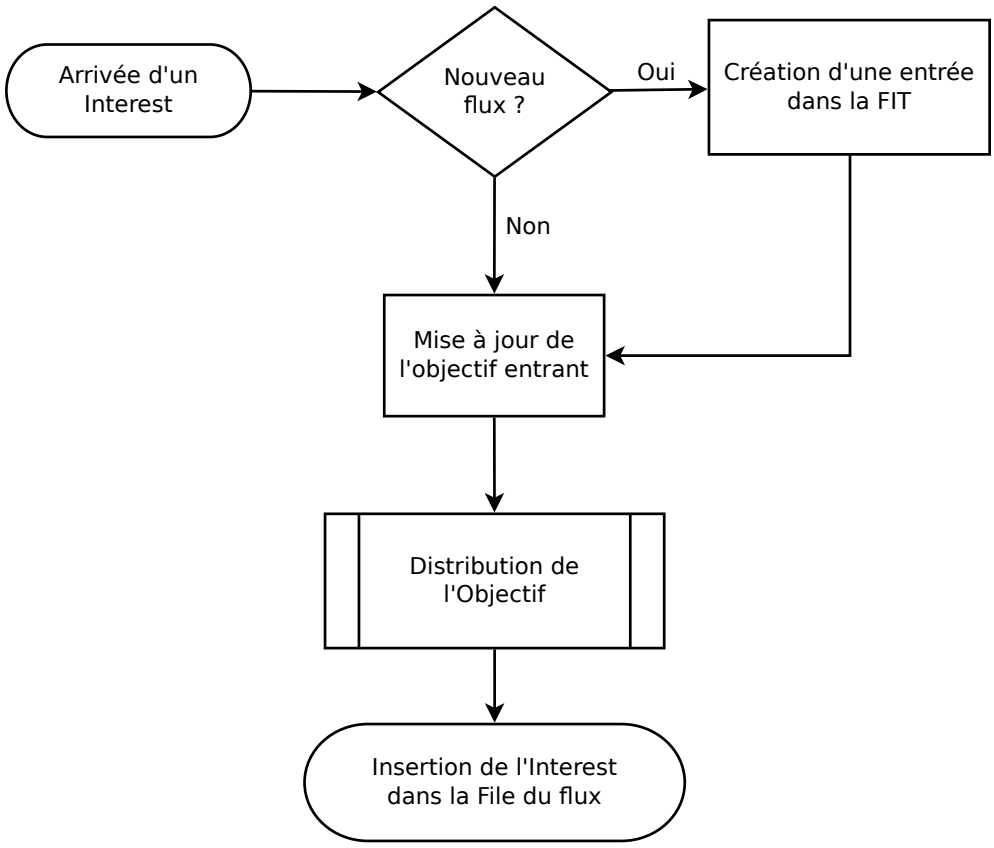


FIGURE 3.5 Arrivée d'un Interest

Algorithme 3.1: Distribution de l'Objectif

Assure: $p_i\_obj = \sum_{k=1}^{N} p_{o,k}\_obj$

jour.

pour toutes les *interfaces de sortie* **faire**

Mettre la contrainte sortante à une part égale de la contrainte entrante

fin

3.3.3 Arrivée d'une Data

De la même manière, la Figure 3.6 présente les mécanismes mis en place lors de l'arrivée d'une Data. Après la mise à jour de la contrainte entrante pour l'entrée de la FIT correspondante, le nœud vérifie que la loi d'Agrégation de Contraintes (cf Équation 3.4) est respectée. Si ce n'est pas le cas, la contrainte sortante actuelle est supérieure à la somme des contraintes entrantes. Elle est donc mise à jour et réduite à cette somme :

$p_o\_max = \sum_{k=1}^{N} p_{i,k}\_max$. Cette nouvelle contrainte est ensuite inscrite dans le champ "Pace Constraint" du paquet qui peut maintenant être émis.

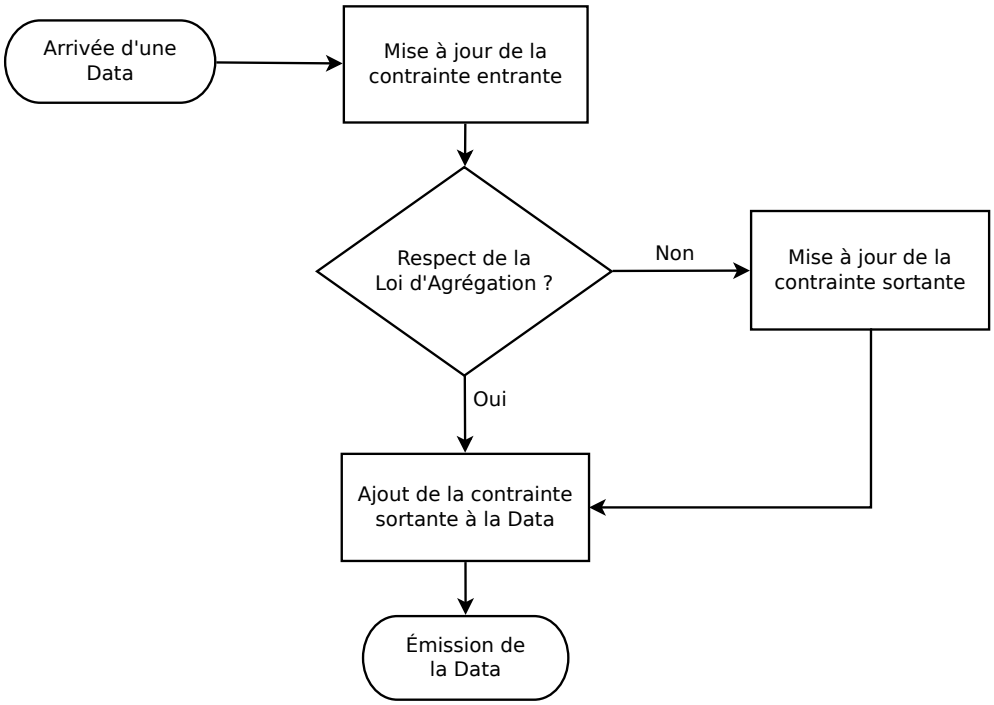


FIGURE 3.6 Arrivée d'une Data

3.3.4 Supervision

Notre principe de Supervision ainsi que les réactions qu'il provoque sont schématisés sur la Figure 3.7. La supervision a lieu toutes les 100ms pour faire écho à la fréquence que peuvent avoir des AQM comme CoDel [58]. Cette fréquence est recommandée par la RFC et montre de bonnes performances dans nos simulations. Nous utilisons justement une AQM pour détecter si le lien est congestionné ou non. L'AQM se veut très simple et est décrite dans l'Algorithme 3.2. Celui-ci détecte s'il y a eu au moins une perte sur le lien durant la dernière période de supervision. Comme pour chaque algorithme, des

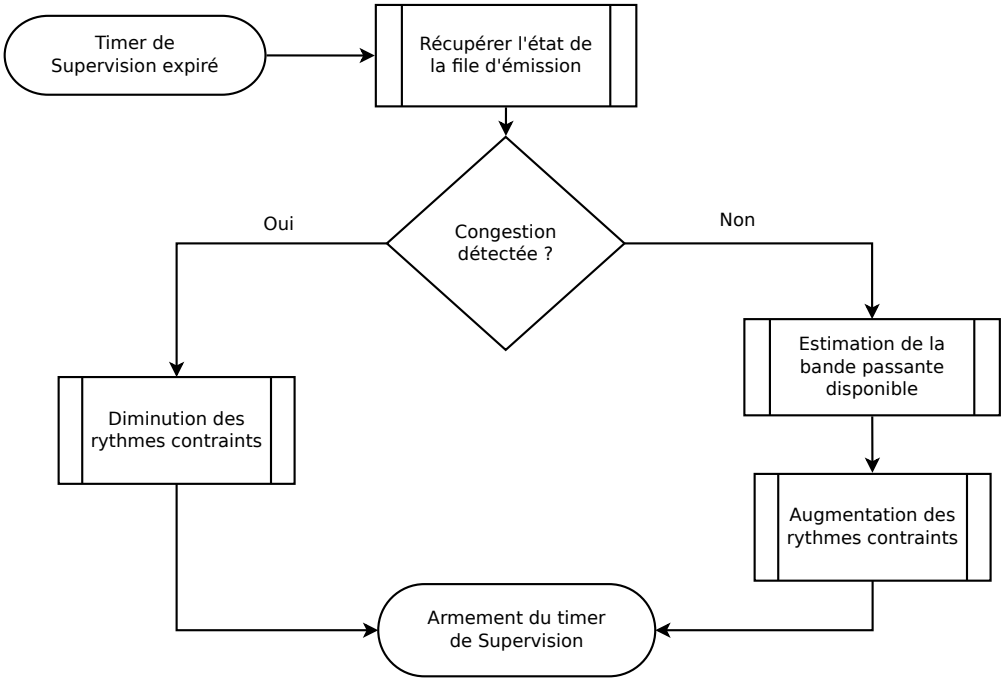


FIGURE 3.7 Supervision

implantations plus complexes afin d'en améliorer les résultats.

Algorithme 3.2: Détection de congestion

Résultat: Lien congestionné
si *une perte a eu lieu durant la dernière période* **alors**
  retourner *Vrai*
sinon
  retourner *Faux*
fin

En cas de détection de congestion, l'Algorithme 3.3 est appelé. Pour la première

Algorithme 3.3: Diminution des rythmes contraints

pour tous les *flux actifs sur l'interface* **faire**
  Réduire le rythme contraint p_o_max de 10%
fin

version de CCC, nous avons choisi de réagir en diminution multiplicative comme le fait par exemple les versions les plus classiques de TCP. Cependant, réduire de moitié nous semblait potentiellement excessif (notamment car la réaction est ici locale). Nous

avons donc préféré ne réduire que de 10% les rythmes contraints des flux actifs sur cette interface.

Si aucune congestion n'est détectée, le nœud estime la bande passante utilisée par l'ensemble des flux et donc la bande passante qui reste disponible. Cette estimation est faite via l'équation suivante :

$$bande\_passante\_disponible = capacité\_lien - \sum_{flux\ k} p_o^k\_max * data\_size^k \tag{3.5}$$

On estime l'utilisation du lien par le flux k en multipliant sa contrainte sortante sur ce lien par la taille du paquet Data. L'utilisation totale du lien est alors déduite en sommant les contributions de chaque flux utilisant le lien. La bande passante disponible est donc la capacité du lien à laquelle on soustrait l'estimation de l'utilisation du lien. Cette valeur est ensuite donnée à l'Algorithme 3.4. Son rôle est de distribuer la bande

Algorithme 3.4: Augmentation des rythmes contraints

Entrée: Bande passante disponible

Assure: $p_o^k\_max \leq \sum_{n=1}^{N} p_{i,n}^k\_max, \forall\ flux\ k$

pour tous les *flux actifs sur l'interface* **faire**

  Augmenter le rythme contraint d'une part égale de la bande disponible

fin

passante inutilisée parmi les flux actifs sur cette interface. Une part égale est donnée à chacun de ces flux et cette partie représente l'implantation de notre stratégie d'équité. Comme cette action provoque une augmentation du rythme contraint sortant, il faut que le nœud s'assure que la loi d'Agrégation des Contraintes est toujours respectée. Pour le flux k, le nouveau rythme contraint sortant vaut alors le minimum entre l'agrégat et la valeur qu'il prendrait avec l'ajout de sa part de la bande passante inutilisée.

Dans tous les cas, la prochaine supervision est prévue.

3.3.5 Définition d'une stratégie d'allocation

Pour cette première implantation de CCC, nous cherchons à ce que la répartition globale des flux soit équitable (comme vu dans le chapitre précédent). Cependant, d'autres stratégies d'allocation peuvent être mises en place. Nous pouvons par exemple imaginer que l'opérateur souhaite économiser ses ressources et fixe une limite de débit pour ses utilisateurs ou encore qu'il dispose de certains liens à haut coût et qu'il préfère les utiliser le moins possible et seulement lorsqu'il n'a pas d'autre choix. Dans notre cas, la stratégie d'allocation prend place dans les algorithmes de répartition du rythme contraint (les

Algorithme 3.3 et 3.4). Nous choisissons en effet de diminuer les rythmes contraints des flux en pourcentage lors d'une congestion et de les augmenter d'une valeur égale lorsqu'il y a du débit disponible. Cela va naturellement favoriser les flux qui ont le moins de débit alloué et contribuer à l'équité dans le réseau. Finalement, lors de l'implantation de chacun de nos algorithmes, il faut réfléchir à quelle est la stratégie d'allocation que nous essayons de mettre en place. Par exemple, dans le cas où un opérateur souhaiterait favoriser l'utilisation d'un lien par rapport à un autre, il faudra qu'il travaille notamment sur l'algorithme de distribution des objectifs (Algorithme 3.1). Dans l'implantation actuelle, nous distribuons l'objectif de manière égale sur l'ensemble des liens disponibles. Dans un tel cas, l'opérateur préférera ne pas distribuer d'objectif, ou peu, sur les liens à hauts coûts pour en limiter l'utilisation. L'architecture modulaire que nous proposons permet ainsi l'adoption de différentes stratégies d'allocation sans avoir forcément le besoin de changer celle-ci.

3.4 Évaluation de CCC

Nous avons implanté CCC sur le simulateur ndnSIM [63]. Les sources de notre implantation sont disponibles librement pour reproduire et utiliser nos résultats [1].

Pour une première évaluation, nous utilisons la topologie illustrée sur la Figure 3.1. Cette topologie assez simple nous permet de bien comprendre les mécanismes mis en place par notre solution. Le consommateur C^1 peut récupérer le contenu cherché depuis deux producteurs, P_1^1 et P_2^1 tandis que le consommateur C^2 n'a lui qu'un producteur à sa disposition, P_1^2. Le flux F^1, en rouge sur la figure, entre le consommateur C^1 et ses deux producteurs dispose de deux chemins, un pour chaque producteur :

— le chemin $c_1^{1,1}$, reliant C^1 à P_1^1 via les nœuds N_1 et N_2,

— et le chemin $c_1^{1,2}$, reliant C^1 à P_2^1 via les nœuds N_1 et N_3.

Le flux F^2, en bleu sur la figure, entre le consommateur C^2 et son producteur dispose d'un unique chemin :

— le chemin $c_1^{2,1}$, reliant C^2 à P_1^2 via les nœuds N_2, N_1 et N_3.

Le point remarquable de cette topologie est que les deux flux sont en concurrence sur le lien entre les nœuds N_1 et N_2, mais pas sur le lien entre les nœuds N_1 et N_3 où les deux flux sont dans des sens contraires. Les deux consommateurs ont des objectifs supérieurs à la capacité du réseau, ils essayent donc d'obtenir le maximum possible. Les deux contenus recherchés ont une taille de 400Mo et sont morcelés en fragment de 8400 octets. Le consommateur C^1 commence à l'instant $t = 0s$ tandis que le consommateur

1. `https://gitlab.tesa.prd.fr/athibaud/ccc-codebase`

C^2 commence lui à l'instant $t = 10s$. Dans ce scénario, nous comparons la première version de notre solution CCC, décrite dans ce chapitre, aux solutions de contrôle de congestion les plus notoires de la littérature, ICP [46] et PCON [57]. Comme expliqué dans le Chap. 1, nous avons décidé de coupler ICP comme mécanisme de bout en bout avec PCON-FS, la partie saut par saut de PCON et PCON-CS, la partie bout en bout de PCON avec la *forwarding strategy* DRF [48]. La distribution optimale nous indique

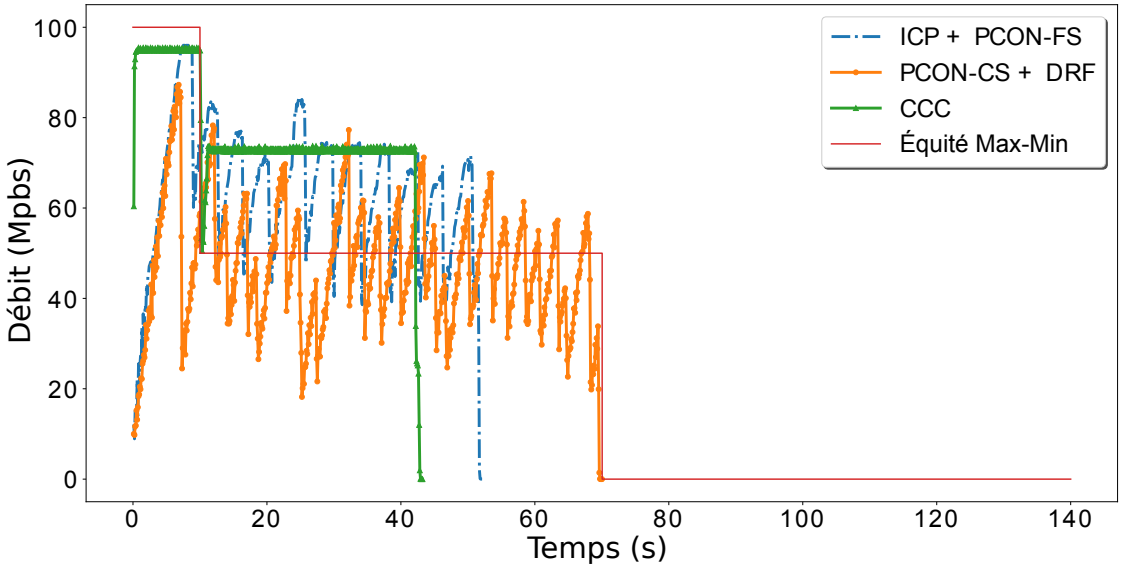


FIGURE 3.8 Débit du consommateur C^1

que les deux consommateurs doivent obtenir un débit d'au total 50Mbps, lorsqu'ils sont tous les deux actifs. Dans cette distribution, le consommateur C^1 utilise complètement son chemin $c_1^{1,2}$ et laisse l'autre chemin inutilisé pour que le consommateur C^2 puisse lui aussi atteindre 50Mbps de débit, via son unique chemin $c_1^{2,1}$. Le lien partagé entre les nœuds N_1 et N_2 n'est alors utilisé que par le consommateur C^2. Les Figure 3.8 et 3.9 montrent, respectivement, les débits obtenus par les consommateurs C^1 et C^2 lors de nos simulations sur ndnSIM.

Pendant les dix premières secondes de la simulation, les deux solutions utilisant une fenêtre de congestion vont progressivement monter jusqu'à utiliser les deux chemins disponibles. Notre solution est capable de converger beaucoup plus rapidement à ce niveau et ne subit pas les oscillations caractéristiques de l'utilisation de la méthode AIMD (Additive Increase Multiplicative Decrease). Au contraire, notre solution montre des débits stables qui ne seront modifiés que lorsqu'un nouveau flux arrive ou qu'un flux déjà présent se termine. Le débit du consommateur avec CCC n'atteint cependant pas

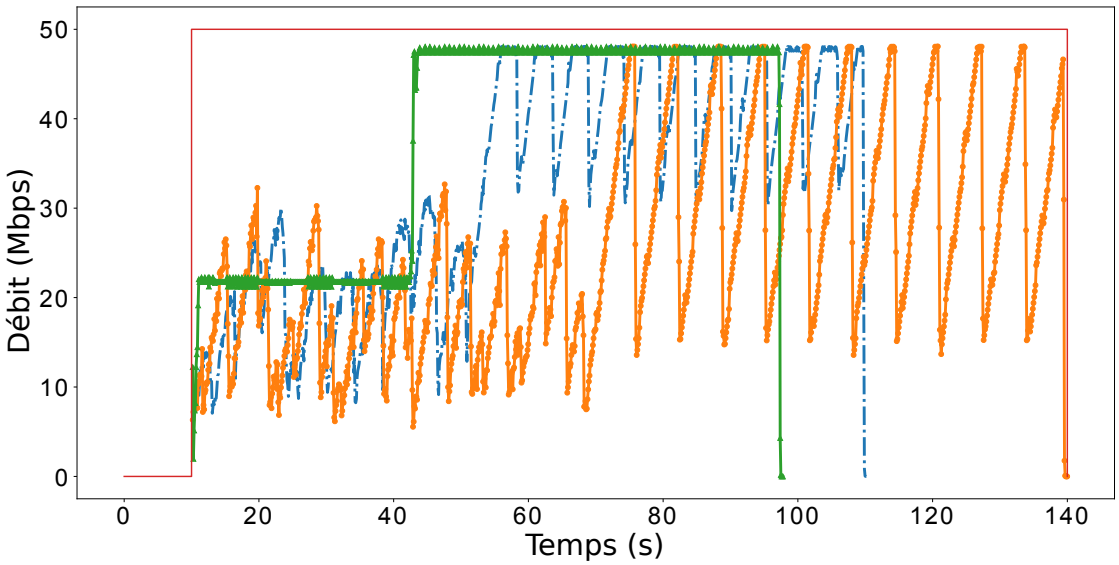


FIGURE 3.9 Débit du consommateur C^2

véritablement 100Mbps. Les 100Mbps de capacité sont en réalité une capacité de niveau 2 et nous traçons le débit de la couche NDN. C'est pourquoi nous atteignons un débit légèrement inférieur à la capacité (à cause de l'overhead).

À $t = 10s$, le consommateur C^2 démarre. La distribution d'équité Max-Min qui indiquait 100Mbps pour le consommateur C^1 et 0 pour C^2, indique maintenant 50Mbps pour chacun. Elle représente la distribution d'équité Max-Min. Cependant, aucune des solutions étudiées ici, pas même CCC, n'atteint cette distribution. En réalité, elles atteignent une équité Max-Min mais seulement sur le lien qui est partagé. C'est ce que nous appelons une équité locale par la suite (à l'inverse d'une équité globale que nous cherchions à atteindre). En effet, après $t = 10s$, le chemin $c_1^{1,2}$ du consommateur C^1 est bien utilisé entièrement comme pour la solution optimale, mais le deuxième chemin $c_1^{1,1}$ avec le lien partagé par les deux flux est utilisé à hauteur d'environ 25Mbps. De même, l'unique chemin $c_1^{2,1}$ du consommateur C^2 est, lui aussi, utilisé à 25Mbps. Au total, le lien limitant entre les nœuds N_1 et N_2 est bien utilisé entièrement mais la capacité est partagée équitablement entre les deux flux. L'objectif d'équité Max-Min n'est donc pas atteint de manière globale pour le moment. Notre solution propose les mêmes résultats que les deux autres solutions en termes d'équité mais dispose de plus d'opportunités pour améliorer cet aspect dans le futur. C'est donc un résultat encourageant : notre solution peut garantir une équité entre les flux localement. De plus, la convergence lorsque le deuxième flux démarre est très rapide (quelques périodes de supervision pour diminuer l'allocation du premier flux et ainsi augmenter celle du second). De la même manière,

lorsque le premier flux se termine, il ne faut qu'une période de supervision pour que la capacité du lien entre N_1 et N_2 ne soit allouée complètement au second flux.

Avec CCC, les débits sont en moyenne plus importants que pour les deux autres solutions. Les temps pour télécharger les contenus sont donc plus courts avec :

— environ 42s pour le premier contenu contre environ 51s et 70s ;

— et environ 88s pour le second contenu contre environ 100s et 130s.

Dans le chapitre 5, nous effectuerons une évaluation de notre solution sur des réseaux satellites.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les principes et choix de conception de notre solution Cooperative Congestion Control. C'est une solution distribuée qui a pour but de répartir équitablement les flux sur le réseau, tout en utilisant efficacement les opportunités de multi-chemins disponibles. Pour cela, les consommateurs, nœuds et producteurs coopèrent à l'aide d'objectifs et contraintes inclus directement dans les paquets NDN. De plus, chaque nœud met en place une supervision locale et estime si un lien est trop ou pas assez utilisé. Grâce à l'ensemble de ces informations, les nœuds décident d'augmenter ou de diminuer les allocations des flux dans le but de leur faire utiliser le réseau de manière efficace et équitable. Les principes et l'architecture de CCC permettent d'avoir une implantation modulable. La première version implantée, que nous avons choisie simple dans le but d'évaluer l'architecture globale, montre des résultats très prometteurs. Bien qu'elle ne soit pas capable d'atteindre une distribution globalement équitable avec le critère Max-Min, elle fait aussi bien que les autres solutions de l'état de l'art et propose une équité Max-Min locale. C'est-à-dire que sur un lien partagé, les différents flux présents auront localement une répartition équitable. Notre solution permet aussi d'utiliser efficacement les multi-chemins quand il y en a et propose une convergence rapide de la distribution quand un nouveau flux arrive ou lorsqu'un flux se termine. Contrairement aux solutions utilisant une fenêtre de congestion, la distribution est stable en dehors de ces évènements. Dans le prochain chapitre, nous allons voir quels sont les problèmes qui se posent pour notre solution quand la topologie dispose de boucle et comment faire évoluer CCC pour surmonter ces problèmes.

Chapitre 4

Cooperative Congestion Control dans des topologies avec boucles

Sommaire

4.1 Introduction

Dans le chapitre précédent, nous avons présenté notre solution Cooperative Congestion Control. Pour simplifier la conception, nous avons tout d'abord travaillé sur des topologies de réseaux en forme d'arbre. C'est-à-dire que l'ensemble des nœuds présents sur les chemins reliant un consommateur aux producteurs du contenu recherché représente un arbre (où la racine est le consommateur et les feuilles les producteurs). Une part de la complexité du problème était ainsi retirée grâce à cette simplification. En effet, pour un nœud donné, les Interests ne pouvaient provenir que d'une seule interface. Il n'y avait donc qu'un seul objectif entrant et qu'une seule contrainte sortante pour le flux considéré. De manière générale, une forme arborescente n'est pas envisageable car peu réaliste. Considérons donc dorénavant un réseau présentant des boucles. Reprenons dans la figure 4.1 la notation décrivant la signalisation d'un flux du point de vue d'un nœud quelconque du réseau. Comme dans le chapitre précédent, il y a N interfaces disponibles pour rejoindre un

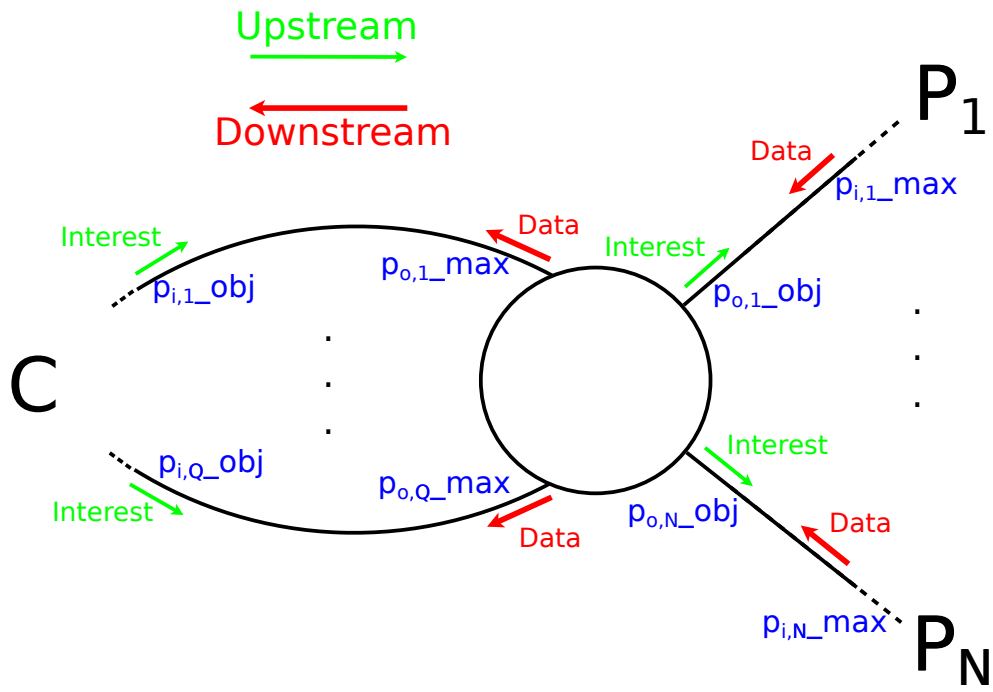


FIGURE 4.1 Signalisation de CCC du point de vue d'un nœud du réseau

fournisseur du contenu. Cependant, au lieu de n'y avoir qu'une seule interface pour atteindre le consommateur, il peut y en avoir plusieurs. Nous notons Q le nombre de ces interfaces. Les Interests de ce flux, qui proviennent bien tous du consommateur C,

sont répartis comme expliqué dans le chapitre précédant sur différents chemins par les nœuds en aval. Certains de ces chemins se croisent à nouveau ici, sur notre nœud. Pour ce flux, l'objectif total qu'il doit répartir en sortie correspond à la somme des Q objectifs qu'il reçoit. Comme pour les contraintes entrantes, le nœud doit mettre en place de l'agrégation d'objectifs. Dans le sens inverse, Q contraintes doivent être émises. Comme pour les objectifs sortants, le nœud doit mettre en place de la distribution des contraintes. L'architecture va donc être plus symétrique qu'auparavant puisqu'il y faut maintenant à la fois agréger les Q objectifs entrants puis les distribuer vers les N interfaces de sortie (pour l'Upstream) et agréger les N contraintes entrantes puis les distribuer vers les Q interfaces de sortie (pour le Downstream). CCC conserve toujours ses trois principes que sont la coopération, la supervision et la répartition, mais nous verrons que les mécanismes et algorithmes qui s'appliquent sur le flux d'Interests et leur objectif vont avoir leur pendant sur le flux de Data et leur contrainte.

4.2 Choix de conception

Dans cette section, nous présentons les différents choix de conception faits pour la nouvelle version de CCC. La Table 4.1 rassemble les anciennes et nouvelles notations de CCC. Nous expliquons les nouvelles notations dans la suite de cette section.

Table 4.1 Liste des Notations de CCC.

Notation	Description
Q	Nombre d'interfaces d'entrée des Interests
N	Nombre d'interfaces de sortie des Interests
$p_{i,q}\_obj$	Rythme Objectif entrant sur l'interface q
$p^k_{i,q}\_obj$	Rythme Objectif entrant sur l'interface q pour le flux k
$p_{o,q}\_max$	Rythme Contraint sortant sur l'interface q
$p^k_{o,q}\_max$	Rythme Contraint sortant sur l'interface q pour le flux k
$p_{o,n}\_obj$	Rythme Objectif sortant sur l'interface n
$p^k_{o,n}\_obj$	Rythme Objectif sortant sur l'interface n pour le flux k
$p_{i,n}\_max$	Rythme Contraint entrant sur l'interface n
$p^k_{i,n}\_max$	Rythme Contraint entrant sur l'interface n pour le flux k
$p_{l,n}\_obj$	Rythme Objectif local sur l'interface n
$p^k_{l,n}\_obj$	Rythme Objectif local sur l'interface n pour le flux k
$p_{l,q}\_max$	Rythme Contraint local sur l'interface q
$p^k_{l,q}\_max$	Rythme Contraint local sur l'interface q pour le flux k

4.2.1 Nouvelles lois

Les lois de *Conservation des Objectifs* et *Agrégation des Contraintes* énoncées dans le chapitre précédent (3.3 et 3.4) doivent s'adapter. Le fait que les informations entrantes soient maintenant multiples n'en est pas l'unique cause. La Figure 4.2 illustre un nouveau problème que nous avons nommé "goulot d'étranglement caché". En effet, si nous gardons

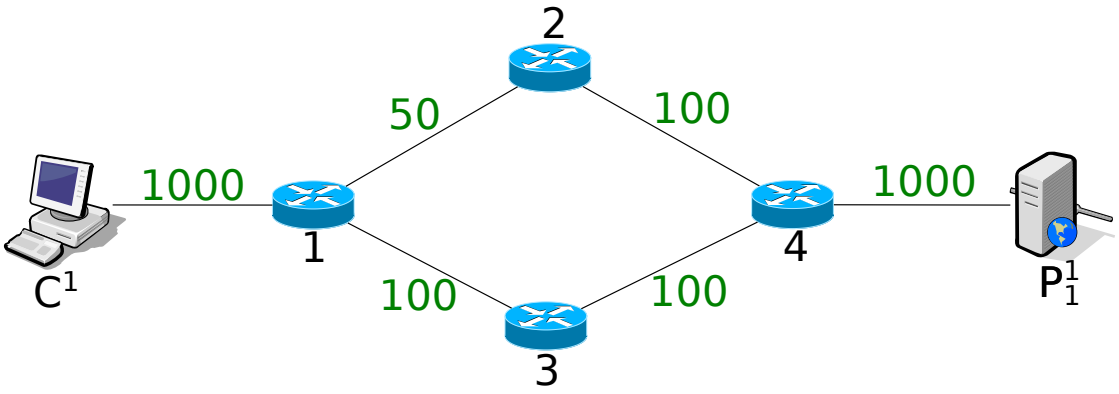


FIGURE 4.2 Exemple du goulot d'étranglement caché

la loi de *Conservation des Objectifs* en état (c'est-à-dire que la somme des objectifs entrants doit être égale à la somme des objectifs sortants), le nœud 1 peut être amené à demander plus que possible sur les deux chemins qui sont disponibles. Cela induirait le nœud 4 en erreur car il n'a pas la vision sur le lien entre les nœuds 1 et 2 (qui est ici limitant). Supposons par exemple que le nœud 1 reçoive un objectif correspondant à 200Mbps (les valeurs vertes correspondent ici aux débits des liens en Mbps). Avec la loi telle qu'énoncée, la somme des objectifs sortants doit donc aussi correspondre à 200Mbps. Une distribution pourrait être de demander 100Mbps pour chacun des deux chemins. Du point de vue du nœud 4, des demandes correspondant à 100Mbps arrivent depuis les nœuds 2 et 3. En supposant que le nœud 4 reçoit des contraintes de seulement 150Mbps, il va devoir choisir comment les répartir entre les deux chemins. Le choix logique de son point de vue (demandes et capacités de 100Mbps sur les deux chemins) est de partager également les contraintes et de donner 75Mbps à chacun. Pour le chemin passant par le nœud 2, la contrainte de 75Mbps est réduite à 50Mbps dû à la capacité du lien entre les nœuds 1 et 2. Finalement, le consommateur C^1 obtient un débit de 125Mbps alors qu'il aurait pu obtenir un débit de 150Mbps. La loi de *Conservation des Objectifs* cache, en effet, la capacité maximale du chemin et induit en erreur le nœud 4 sur la répartition des contraintes. Il est donc nécessaire de remplacer la loi de conservation des objectifs par la suivante :

Loi : Non Création d'Objectifs :

$$\sum_{n=1}^{N} p_{o,n}\_obj \leq \sum_{q=1}^{Q} p_{i,q}\_obj \tag{4.1}$$

Le nœud 1 a maintenant le droit de supprimer de la demande s'il estime que celle-ci ne peut pas être satisfaite (comme dans notre cas où le lien n'avait pas assez de capacité pour). La loi a ici le rôle d'interdire la création d'objectifs par les nœuds du réseau. C'est aussi l'idée qu'il y a derrière la loi d'*Agrégation des Contraintes* que nous avons décidé de renommer pour mieux faire apparaître ce concept :

Loi : Non Suppression de Contraintes :

$$\sum_{q=1}^{Q} p_{o,q}\_max \leq \sum_{n=1}^{N} p_{i,n}\_max \tag{4.2}$$

Avec cette loi, les nœuds peuvent ajouter des contraintes, mais pas en supprimer. De plus, cela met aussi en valeur la symétrie entre les traitements en Upstream et en Downstream, comme indiqué dans l'introduction.

4.2.2 Architecture symétrique

Dans cette nouvelle version de CCC, nous présentons une architecture symétrique entre le flux d'Interests (Upstream) et le flux de Data (Downstream). Ce n'était pas le cas dans la précédente version où il n'y avait une répartition inter-flux seulement sur les contraintes et une répartition inter-chemins seulement sur les objectifs. La répartition inter-flux des contraintes était déclenchée par la supervision et la répartition inter-chemins des objectifs par l'arrivée d'un nouvel Interest (et donc d'un nouvel objectif). L'architecture générale est maintenant identique pour l'Upstream et le Downstream et se décompose en deux processus de répartition :

— la répartition inter-chemins des objectifs, respectivement des contraintes, qui est déclenchée par l'arrivée d'un Interest, respectivement d'une Data ;
— la répartition inter-flux des objectifs, respectivement des contraintes, qui est déclenchée par l'Upstream supervision, respectivement le Downstream supervision.

Nous avons donc deux algorithmes différents qui gèrent la répartition. Pour qu'ils n'interfèrent pas de manière néfaste entre eux (asynchrone et non concertée), nous

introduisons deux variables locales supplémentaires ($p_{l,q}\_max$ pour les contraintes et $p_{l,n}\_obj$ pour les objectifs). L'algorithme de répartition inter-flux a le rôle, pour une interface donnée, de répartir la capacité du lien sur l'ensemble des flux présents et détermine la variable locale de chacun de ces flux sur l'interface. La variable locale représente l'allocation théorique entre les flux et sert ensuite d'intermédiaire pour la seconde répartition où elle représentera une limite à ne pas franchir pour respecter cette allocation. L'algorithme de répartition inter-chemin a le rôle, pour un flux donné, de répartir la donnée entrante totale (objectif ou contrainte) sur l'ensemble des chemins. Pour cela, il détermine le rythme sortant qui devra être inférieur au rythme local déterminé lors du dernier appel à l'algorithme de répartition inter-flux. Les rythmes locaux d'un flux peuvent ne pas respecter les deux lois car ils ne représentent pas l'allocation finale. La seconde répartition devra, elle, bien s'assurer que les lois sont respectées lors qu'elle détermine les rythmes sortants.

La supervision a, elle, le rôle de s'assurer que la répartition finale sur une interface est convenable. Elle utilise donc les rythmes sortants des flux présents sur l'interface.

4.2.3 Un peu de dissymétrie

Bien que l'architecture soit symétrique, les implantations des algorithmes pour l'Upstream et le Downstream vont différer. En effet, la signification des informations montantes, les objectifs, n'est pas la même que celle des informations descendantes, les contraintes. Les contraintes ont pour rôle de représenter la répartition des flux que nous recherchons. Les objectifs ne sont finalement qu'une aide pour atteindre cette répartition. Nous verrons en détails ces implantations dans la section 4.3 et leurs performances dans la section 4.4. Un premier aperçu de cette dissymétrie est présenté dans la prochaine sous-section où certaines notions n'ont de sens que pour le Downstream.

4.2.4 Nouvelles notions

Comme nous allons le voir dans la suite, les nouveaux algorithmes que nous avons défini ici sont plus sophistiqués que les premières versions introduites dans le chapitre précédent. Pour cela, nous allons avoir besoin d'expliciter de nouvelles notions.

4.2.4.1 Flux limité

La première notion est celle de *Flux limité*. Elle a un lien direct avec les deux lois définies précédemment. Comme pour la plupart des notions que nous définissons, elle a

deux versions : une pour l'Upstream et l'autre pour le Downstream. Un flux est Upstream limité lorsque le nœud émet autant d'objectifs qu'il en reçoit :

Définition : Un flux est Upstream limité quand $\sum_{n=1}^{N} p_{o,n}\_obj = \sum_{q=1}^{Q} p_{i,q}\_obj$

Un nœud ne peut donc pas augmenter un objectif sortant si le flux est Upstream limité, sans enfreindre la loi de *Non Création d'Objectifs*. De la même manière, un flux est Downstream limité lorsque le nœud émet autant de contraintes qu'il en reçoit :

Définition : Un flux est Downstream limité quand $\sum_{q=1}^{Q} p_{o,q}\_max = \sum_{n=1}^{N} p_{i,n}\_max$

Ainsi, un nœud ne peut pas augmenter une contrainte sortante si le flux est Downstream limité, sans enfreindre la loi de *Non Suppression de Contraintes*.

4.2.4.2 Flux satisfait

Nous définissons un *flux satisfait* comme un flux dont la demande (objectif) est satisfaite via une allocation (contrainte). Cette notion n'a donc du sens que pour le Downstream. C'est aussi une notion locale, c'est-à-dire qu'elle ne concerne qu'une interface d'un nœud. Un flux peut en effet être satisfait sur une interface du nœud mais pas sur une autre. Pour résumer, un flux est satisfait sur une interface lorsque le nœud émet au moins autant de contraintes qu'il ne reçoit d'objectifs :

Définition :
Un flux est satisfait sur l'interface q quand $p_{o,q}\_max \geq p_{i,q}\_obj$

Cette notion est utilisée par la supervision pour s'assurer que la répartition est convenable mais aussi par la répartition inter-flux lorsqu'elle ne l'est justement pas. La répartition inter-flux détermine les variables locales, comme expliqué précédemment. Il faudra donc utiliser ces variables à la place des variables sortantes dans la notion de *flux satisfait*.

De plus, comme un flux peut tout à fait être plus que satisfait, nous spécifions l'état de flux sur-satisfait comme celui d'un flux satisfait mais où la contrainte sortante est strictement supérieure à l'objectif entrant. C'est le cas où le nœud prévoit une plus grande allocation de ressources sur un lien pour un flux que ce qui a été demandé.

4.2.4.3 Distribution satisfaisante

La notion de *distribution satisfaisante* est utilisé par la supervision et la répartition inter-flux. Elle a bien entendu un lien fort avec la notion de *flux satisfait*. Pour une

interface donnée, il n'est pas forcément possible de satisfaire tous les flux présents. Une distribution satisfaisante n'est donc pas une distribution où tous les flux sont satisfaits, bien qu'une telle distribution est évidemment satisfaisante. Le but est de commencer par satisfaire tous les flux, si cela est possible. C'est seulement une fois que tous les flux sont satisfaits que le nœud peut proposer plus à certains flux. Une distribution est donc satisfaisante si aucun flux est sur-satisfait ou, s'il en existe au moins un sur-satisfait, alors tous les autres sont satisfaits. Une distribution satisfaisante est définie de la manière suivante :

Définition : Une distribution est satisfaisante si et seulement si :
si un flux est sur-satisfait, alors tous les autres flux sont au moins satisfaits.

De manière logique, la répartition inter-flux va chercher à établir une répartition satisfaisante et la supervision va vérifier qu'elle l'est bien.

4.2.4.4 Distribution équitable

La dernière notion que nous définissons ici est la notion de *distribution équitable*. Nous utilisons ici le critère d'équité Max-Min, mais au niveau local du lien considéré. Cette notion nous permet de directement comparer les allocations entre les flux sur une interface. Pour le Downstream, un flux peut se voir assigner une allocation inférieure aux autres flux seulement car il est limité ou satisfait. En effet, si le nœud n'est pas en mesure d'augmenter son allocation, cela ne doit pas impacter les autres flux. Une distribution Downstream équitable est donc définie de la manière suivante :

Définition : Une distribution est Downstream équitable si et seulement si :
$\forall F_1, F_2$ si $p_{o,q}^{F_1}\_max \leq p_{o,q}^{F_2}\_max$ alors F_1 est satisfait ou F_1 est Downstream limité

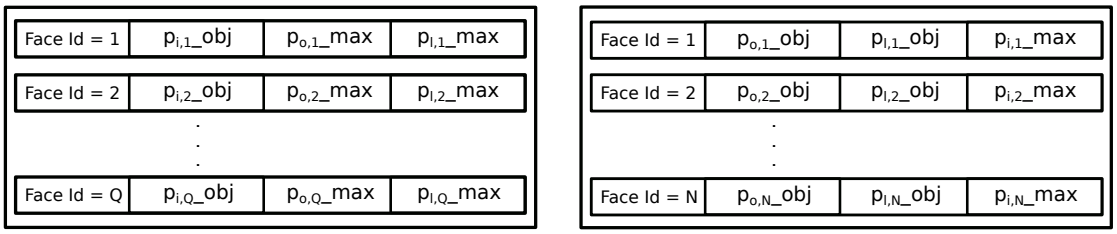
Pour l'Upstream, il n'y a pas de notion de satisfaction donc un flux peut se voir assigner une allocation inférieure aux autres flux que s'il est limité. Une distribution Upstream équitable est donc définie de la manière suivante :

Définition : Une distribution est Upstream équitable si et seulement si :
$\forall F_1, F_2$ si $p_{o,n}^{F_1}\_obj \leq p_{o,n}^{F_2}\_obj$ alors F_1 est Upstream limité

4.3 Structure et Algorithmes

La signalisation reste inchangée par rapport à la version du chapitre précédent. Nous avons cependant introduit les variables locales (qui ne font donc pas parties de

la signalisation) et les structures de données sont amenées à changer pour pouvoir les sauvegarder. La FIT ne change pas (voir Figure 3.4a du chapitre précédent) mais les entrées des IFIL et OFIL contiennent maintenant les variables locales associées. La IFIL est maintenant aussi une liste puisque nous considérons plusieurs interfaces pouvant atteindre le consommateur. La Figure 4.3 présente le nouveau format de ces deux tables.



(a) Input Face Information List (b) Output Face Information List

FIGURE 4.3 Tables de CCC

Pour rappel, notre architecture est symétrique entre le Downstream et l'Upstream où chacun possède trois algorithmes :

- la répartition inter-flux qui détermine les rythmes locaux de tous les flux présents sur une interface donnée,
- la répartition inter-chemin qui détermine les rythmes sortants pour un flux sur toutes les interfaces disponibles,
- et la supervision qui vérifie que l'ensemble des rythmes sortants des flux sortants sur une interface donnée respecte les critères de la stratégie d'allocation.

Ces algorithmes sont eux-mêmes appelés lors de deux mécanismes asynchrones :

- l'expiration du timer de supervision qui appelle l'algorithme de supervision, qui appelle à son tour et au besoin l'algorithme de répartition inter-flux,
- l'arrivée d'un paquet (Data pour le Downstream et Interest pour l'Upstream) qui appelle l'algorithme de répartition inter-chemin.

4.3.1 Downstream

Le premier mécanisme travaille à l'échelle d'une interface et gère plusieurs flux à la fois. Le premier algorithme de ce mécanisme est la supervision sur le Downstream et fonctionne comme indiquée sur le diagramme de la Figure 4.4. Lorsque le timer de supervision expire (1) pour l'interface d'indice q, le nœud va estimer l'utilisation de la bande passante (2) en utilisant la formule suivante :

$$bande\_passante_q = \sum_F (p^F_{o,q}\_max * data\_size^F) + \sum_{F'} (p^{F'}_{i,q}\_max * interest\_size^{F'}) \quad (4.3)$$

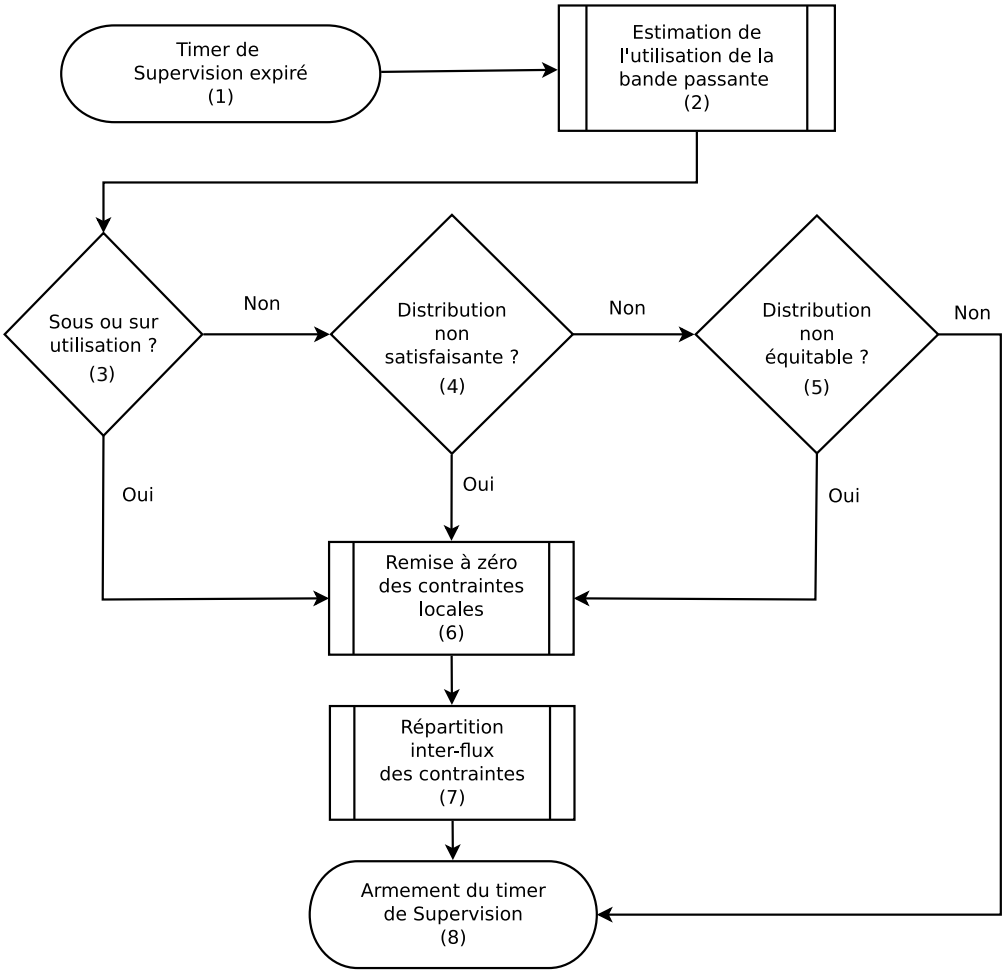


FIGURE 4.4 Downstream Supervision

où F représente les flux dont les Data sont émises par l'interface et F' les flux dont les Interests sont émis par l'interface. Pour la Downstream supervision, nous nous intéressons aux Data qui sont émises par notre interface. Cependant, cette interface émet aussi des Interests (pour des flux qui sont dans le sens inverse) et ces Interests utilisent eux aussi une partie de la bande passante. Bien que la taille des Interests soit faible en comparaison à celle des Data, ils ne sont pas à négliger. C'est pourquoi, la formule 4.3 comprend deux composantes : une pour les Data des flux à superviser et une pour les Interests des flux qui circulent dans le sens inverse.

La supervision va maintenant s'assurer que la répartition actuelle des ressources entre les flux actifs sur l'interface vérifie trois propriétés. La première est que toute la capacité du lien est utilisée (3), mais pas au-delà (pour éviter la congestion). La deuxième est

que la distribution est satisfaisante (4). Finalement, la troisième est que la distribution est équitable (5). Si l'une de ces trois propriétés n'est pas vérifiée, les contraintes locales sont remises à zéro (6) et le deuxième algorithme est appelé : l'algorithme de répartition inter-flux (7) via l'Algorithme 4.1. Dans un premier temps, l'algorithme de répartition

Algorithme 4.1: Répartition inter-flux sur le Downstream

Entrée: Bande passante disponible
Sorties: $p^F_{l,q}\_max$, $\forall\ flux\ F$
Assure: $\sum_F p^F_{l,q}\_max = bande\_passante\_disponible$
pour tous les *flux actifs et non satisfaits sur l'interface* **faire**
  Augmenter le rythme contraint local d'une part égale de la bande disponible jusqu'à satisfaction du flux
fin
Mise à jour de la bande passante disponible
si *Il reste de la bande passante disponible* **alors**
  pour tous les *flux actifs et non limités sur l'interface* **faire**
    Augmenter le rythme contraint local d'une part égale de la bande disponible jusqu'à limitation du flux
  fin
  Mise à jour de la bande passante disponible
fin
si *Il reste de la bande passante disponible* **alors**
  Équilibrage des contraintes locales entre les flux actifs sur l'interface
fin

inter-flux cherche à satisfaire l'ensemble des flux actifs sur l'interface. Pour cela, il donne une part égale de la bande passante disponible à chacun de ces flux. Si cette part égale est supérieure au montant nécessaire pour satisfaire un flux, ce flux est juste satisfait et le surplus est mis de côté pour la deuxième distribution. Une fois la première distribution faite, s'il y a du surplus, cela signifie que tous les flux actifs sont satisfaits. La deuxième distribution sert maintenant à limiter les flux qui ne le sont pas déjà. La première distribution ne se soucie effectivement pas de la limitation des flux. Après la première distribution, un flux peut donc avoir reçu une part qui le limite théoriquement. De la même manière que la première distribution, la deuxième donne une part égale du surplus à tous les flux non limités. Si la part est supérieure au montant nécessaire pour limiter un flux, la contrainte locale est augmentée pour limiter le flux mais pas au-delà. Le surplus est à nouveau mis de côté pour une troisième et dernière distribution. Après la deuxième distribution, s'il y a du surplus, tous les flux sont au moins satisfaits et limités (la première distribution peut aller au-delà de la limite du flux et la deuxième au-delà

de la satisfaction). La troisième distribution a pour but de distribuer tout le surplus. Il n'y a donc aucune condition pour borner l'augmentation d'une contrainte sortante. Nous avons donc choisi ici d'essayer d'équilibrer les contraintes locales des flux dans le but de rendre effective notre stratégie d'allocation (équité Max-Min). C'est aussi pour cette raison que dans les deux premières distributions, nous donnions une part égale à chaque flux. Comme nous autorisons les contraintes locales à aller au-delà de la limite du flux, la loi de Non Suppression de Contraintes pourrait ne pas être respectée. Les variables locales ne sont que des indications pour le troisième algorithme qui aura la mission de déterminer les contraintes sortantes qui respecteront effectivement la loi de Non Suppression de Contraintes. Finalement, le timer de supervision est ré-armé (8).

Le deuxième mécanisme travaille à l'échelle du nœud mais ne gère qu'un seul flux à la fois. Il est déclenché à l'arrivée d'un paquet Data (1) et fonctionne comme indiqué sur le diagramme de la Figure 4.5. Le fonctionnement est sensiblement le même que dans la

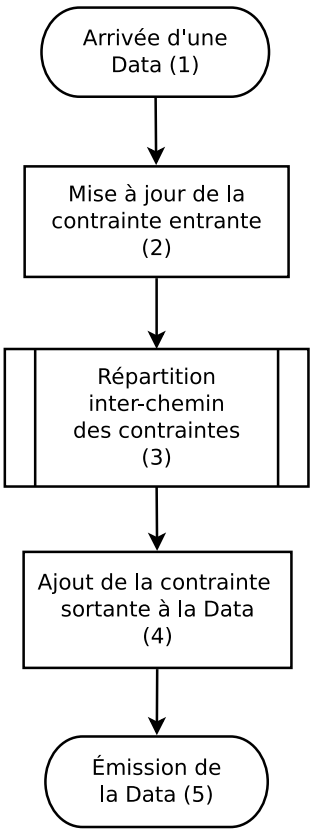


FIGURE 4.5 Arrivée d'une Data

version précédente où à l'arrivée de la Data, la contrainte entrante est mise à jour (2) et où

juste avant l'émission de la Data, la contrainte sortante y est ajoutée (4). La différence a lieu entre ces deux actions avec l'introduction de l'algorithme de répartition inter-chemin des contraintes (3). Dans la version précédente, il n'y avait qu'un unique chemin pour rejoindre le consommateur et il fallait simplement s'assurer que la loi de Non Suppression de Contraintes soit respectée. Dans la nouvelle version, le consommateur peut être joint via plusieurs chemins et une répartition s'impose. L'Algorithme 4.2 est l'implantation de cette répartition inter-chemin des contraintes. Il prend en entrée les contraintes locales

Algorithme 4.2: Répartition inter-chemins sur le Downstream

Entrée: $p_{l,q}\_max, \forall q$
Entrée: $p_{i,q}\_obj, \forall q$
Entrée: $\sum_{n=1}^{N} p_{i,n}\_max$
Sorties: $p_{o,q}\_max, \forall\ q$
Assure: $\sum_{q=1}^{Q} p_{o,q}\_max \leq \sum_{n=1}^{N} p_{i,n}\_max$
Calculer la demande ne pouvant pas être satisfaite (ou ρ_D)
pour tous les *chemins disponibles* **faire**
  Mettre la contrainte sortante à l'objectif entrant auquel est retirée une part égale de ρ_D
  si *la contrainte sortante est supérieure à la contrainte locale* **alors**
    Mettre la contrainte sortante à la contrainte locale
  fin
fin
Calculer le surplus non distribué
pour tous les *chemins non saturés* **faire**
  Augmenter la contrainte sortante d'une part égale du surplus
  si *la contrainte sortante est supérieure à la contrainte locale* **alors**
    Mettre la contrainte sortante à la contrainte locale
  fin
fin

de chacun des chemins (pour ne pas allouer plus que cette valeur), les objectifs entrants (pour s'approcher au mieux de la demande) et le total des contraintes entrantes (la valeur à distribuer entre tous les chemins). Il s'occupe d'assigner les contraintes sortantes pour chacun des chemins et s'assure bien sûr que la loi de Non Suppression de Contraintes est respectée. Dans cette implantation, nous commençons par calculer la demande ne

pouvant pas être satisfaite (que nous appelons ρ_D) via la formule suivante :

$$\rho_D = \max(0, \sum_{q=1}^{Q} p_{i,q}\_obj - \sum_{n=1}^{N} p_{i,n}\_max) \tag{4.4}$$

S'il y a plus de contraintes entrantes que d'objectifs entrants, cela veut dire que l'on peut a priori répondre à la demande, et ρ_D est donc de zéro. Une première distribution cherche à satisfaire tous les chemins disponibles mais en cas de ρ_D non nul, répartir le manque également entre ces différents chemins. La contrainte locale est aussi prise en compte et ne devra pas être dépassée puisqu'elle représente la part allouée au flux sur l'interface considérée. La formule suivante représente la première distribution :

$$\forall q, p_{o,q}\_max = \min(p_{i,q}\_obj - \frac{\rho_D}{Q}, p_{l,q}\_max) \tag{4.5}$$

Nous calculons ensuite le surplus de contraintes entrantes n'ayant pas été distribué :

$$surplus = \sum_{n=1}^{N} p_{i,n}\_max - \sum_{q=1}^{Q} p_{o,q}\_max \tag{4.6}$$

Ce surplus est non nul lorsqu'au moins un chemin a vu sa contrainte sortante restreinte par l'allocation inter-flux (sa contrainte locale). Nous appelons de tels chemins des chemins saturés. Une seconde distribution a ensuite pour but de distribuer ce surplus également parmi les chemins non saturés. La formule suivante représente la seconde distribution :

$$\forall q \mid p_{o,q}\_max < p_{l,q}\_max,\ p_{o,q}\_max = \min(p_{o,q}\_max + \frac{surplus}{Q'}, p_{l,q}\_max) \tag{4.7}$$

Où Q' représente le nombre de chemins non saturés. La première distribution, représenté par l'Équation 4.5, permet d'indiquer à tous les chemins l'incapacité de gérer autant de demandes. Les nœuds en aval peuvent ainsi choisir de basculer de la demande sur d'autres chemins plus disponibles. La seconde distribution, représenté par l'Équation 4.7, permet à l'inverse d'indiquer, lorsque c'est possible, la capacité de gérer plus de demandes. Les nœuds en aval ont ainsi la possibilité d'augmenter la demande sur ces chemins. Finalement, le paquet Data est émis (5).

4.3.2 Upstream

Comme pour le Downstream, l'Upstream est composé de trois algorithmes partagés en deux mécanismes asynchrones. Les algorithmes et mécanismes sont les mêmes qu'en

Downstream mais transposés sur l'Upstream. Ainsi, le premier algorithme est la supervision sur l'Upstream et fonctionne comme indiqué sur le diagramme de la Figure 4.6. Lorsque le timer de supervision expire (1) pour l'interface d'indice n, le nœud estime

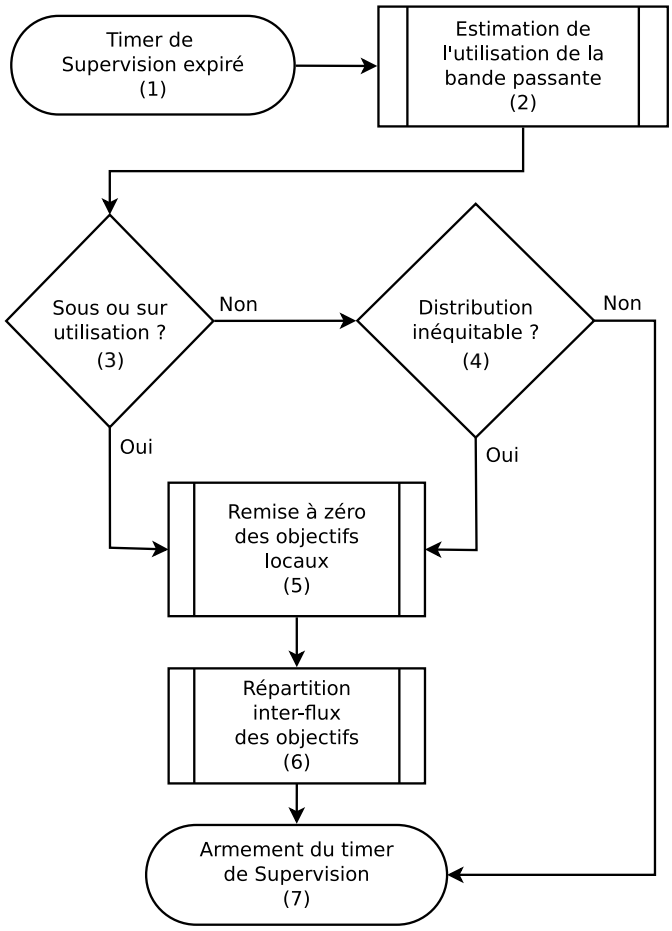


FIGURE 4.6 Upstream Supervision

l'utilisation de la bande passante (2) en utilisant la formule suivante :

$$bande\_passante_n = \sum_F (p^F_{o,n}\_obj * data\_size^F) \tag{4.8}$$

Le nœud estime ainsi le débit qui serait utilisé si tous les flux étaient satisfaits. La supervision s'assure que la répartition actuelle des objectifs sur l'interface vérifie deux propriétés. La première est que la capacité entière du lien est utilisée (3), mais aussi qu'elle ne soit pas plus demandée que possible. La seconde propriété est que la distribution est équitable (4). Si l'une de ces deux propriétés n'est pas vérifiée, les objectifs locaux sont

remis à zéro (5) et le deuxième algorithme est appelé (6). L'algorithme de répartition inter-flux sur l'Upstream est décrit dans l'Algorithme 4.3. Le nœud commence par répartir

Algorithme 4.3: Répartition inter-flux sur le Upstream

Entrée: Bande passante disponible
Sorties: $p^F_{l,n}\_obj$, $\forall$ $flux$ F
Assure: $\sum_F p^F_{l,n}\_obj = bande\_passante\_disponible$
pour tous les *flux actifs et non limités sur l'interface* **faire**
  Augmenter le rythme objectif local d'une part égale de la bande disponible jusqu'à limitation du flux
fin
Mise à jour de la bande passante disponible
si *Il reste de la bande passante disponible* **alors**
  Équilibrage des objectifs locaux entre les flux actifs sur l'interface
fin

la bande passante disponible également entre chacun des flux actifs, mais sans dépasser la limite du flux. Une fois cette première distribution faite, s'il y a du surplus, cela signifie que tous les flux sont limités. La seconde et dernière distribution a pour but de distribuer ce surplus. Comme pour le Downstream, nous avons choisi d'équilibrer les objectifs locaux dans le but de rendre efficace notre stratégie d'allocation (équité Max-Min). Finalement, le timer de supervision est ré-armé (7).

En Upstream, le deuxième mécanisme est déclenché à l'arrivée d'un Interest (1) et fonctionne comme indiqué sur le diagramme de la Figure 4.7. Le fonctionnement est très similaire à celui de la version précédente où le noeud vérifie si l'Interest correspond à un nouveau flux ou non (2). Si oui, il créé une nouvelle entrée dans la FIT (3). Dans tous les cas, il met à jour l'objectif entrant avec celui reçu (4). La différence par rapport à la version précédente est sur l'implantation de l'algorithme de répartition inter-chemin des objectifs (5). L'implantation repose sur la même conception que celle du Downstream et est présentée dans l'Algorithme 4.4. Il prend en entrée les objectifs locaux de chacun des chemins (pour ne pas demander plus que cette valeur), les contraintes entrantes (pour s'approcher au mieux de l'allocation) et le total des objectifs entrants (la valeur à distribuer entre tous les chemins). Il s'occupe d'assigner les objectifs sortants pour chacun des chemins et s'assure du respect de la loi de Non Création d'Objectifs. Comme pour le Downstream, nous commençons par calculer l'allocation ne pouvant pas être

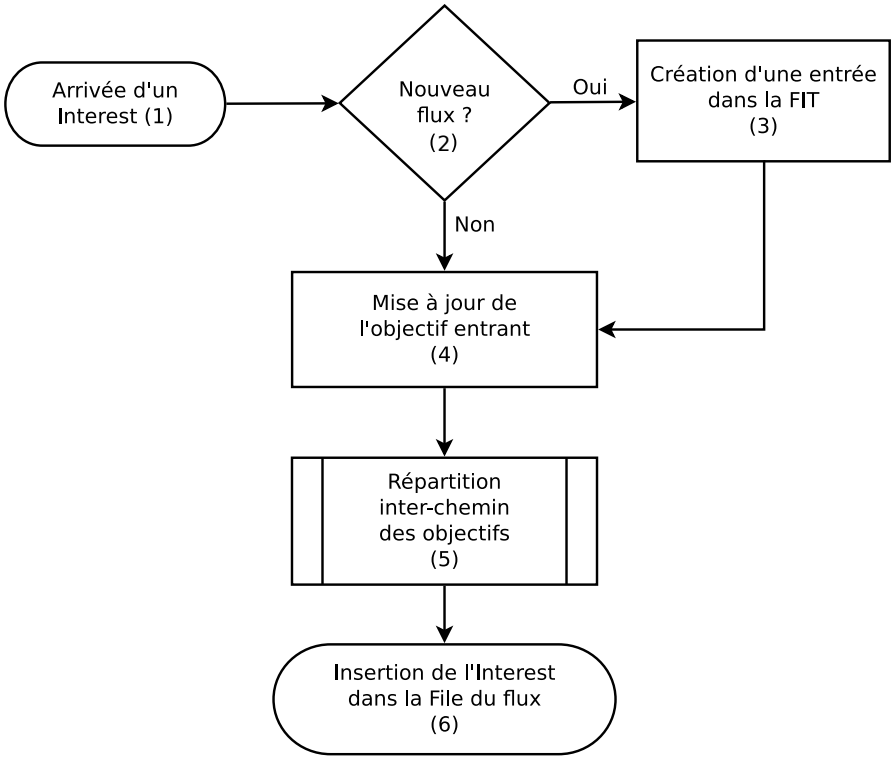


FIGURE 4.7 Arrivée d'un Interest

demandée (que nous appelons aussi ρ_U) via la formule suivante :

$$\rho_U = \max(0, \sum_{n=1}^{N} p_{i,n}\_max - \sum_{q=1}^{Q} p_{i,q}\_obj) \tag{4.9}$$

S'il y a plus d'objectifs entrants que de contraintes entrantes, cela veut dire que l'on peut a priori demander autant que l'allocation, et ρ_U est donc de zéro. Une première distribution cherche à demander autant que l'allocation sur tous les chemins disponibles mais en cas de ρ_U non nul, de répartir le manque également entre ces différents chemins. L'objectif local est aussi pris en compte et ne devra pas être dépassé puisqu'il représente la part allouée au flux sur l'interface considérée. La formule suivante représente la première distribution :

$$\forall n, p_{o,n}\_obj = \min(p_{i,n}\_max - \frac{\rho_U}{N}, p_{l,n}\_obj) \tag{4.10}$$

Algorithme 4.4: Répartition inter-chemins sur le Upstream

Entrée: $p_{l,n}\_obj, \forall n$
Entrée: $p_{i,n}\_max, \forall n$
Entrée: $\sum_{q=1}^{Q} p_{i,q}\_obj$
Sorties: $p_{o,n}\_obj, \forall\ n$
Assure: $\sum_{n=1}^{N} p_{o,n}\_obj \leq \sum_{q=1}^{Q} p_{i,q}\_obj$
Calculer l'allocation ne pouvant pas être demandée (ou ρ_U)
pour tous les *chemins disponibles* **faire**
  Mettre l'objectif sortant à la contrainte entrante auquel est retirée une part égale de ρ_U
  si *l'objectif sortant est supérieure à l'objectif local* **alors**
    Mettre l'objectif sortant à l'objectif local
  fin
fin
Calculer le surplus non distribué
pour tous les *chemins non saturés* **faire**
  Augmenter l'objectif sortant d'une part égale du surplus
  si *l'objectif sortant est supérieure à l'objectif local* **alors**
    Mettre l'objectif sortant à l'objectif local
  fin
fin

Nous calculons ensuite le surplus d'objectifs entrants n'ayant pas été distribué :

$$surplus = \sum_{q=1}^{Q} p_{i,q}\_obj - \sum_{n=1}^{N} p_{o,n}\_obj \tag{4.11}$$

Ce surplus est non nul lorsqu'au moins un chemin a vu son objectif sortant restreint par l'allocation inter-flux (son objectif local). Nous appelons de tels chemins des chemins saturés. Comme pour le Downstream, une seconde distribution a ensuite pour but de distribuer ce surplus également parmi les chemins non saturés. La formule suivante représente la seconde distribution :

$$\forall n \mid p_{o,n}\_obj < p_{l,n}\_obj,\ p_{o,n}\_obj = \min(p_{o,n}\_obj + \frac{surplus}{N'}, p_{l,n}\_obj) \tag{4.12}$$

Où N' représente le nombre de chemins non saturés. La première distribution, représentée par l'Équation 4.10, permet d'indiquer à tous les chemins le manque de besoin. Les nœuds

en amont peuvent ainsi choisir de basculer de l'allocation sur d'autres chemins plus demandant. La seconde distribution, représentée par l'Équation 4.12, permet à l'inverse d'indiquer, lorsque c'est possible, un surplus de demandes. Les nœuds en amont ont ainsi la possibilité d'augmenter l'allocation sur ces chemins si c'est possible.

4.4 Évaluation des performances

Dans cette section, nous évaluons la nouvelle version de CCC. Les scénarios du chapitre précédent ont été ré-évalués et les performances restent inchangées. Dans les prochaines sous-sections, nous présentons le nouveau scénario utilisant une topologie complexe contenant des boucles et les performances sur trois cas distincts.

4.4.1 Scénario

Pour illustrer l'évolution de CCC dans ce chapitre, nous avons conduit notre évaluation sur la classique topologie Abilene. Le routage mis en place est inspiré de [48] et est présenté dans la Figure 4.8. Cette topologie est composée de 11 routeurs représentant chacun

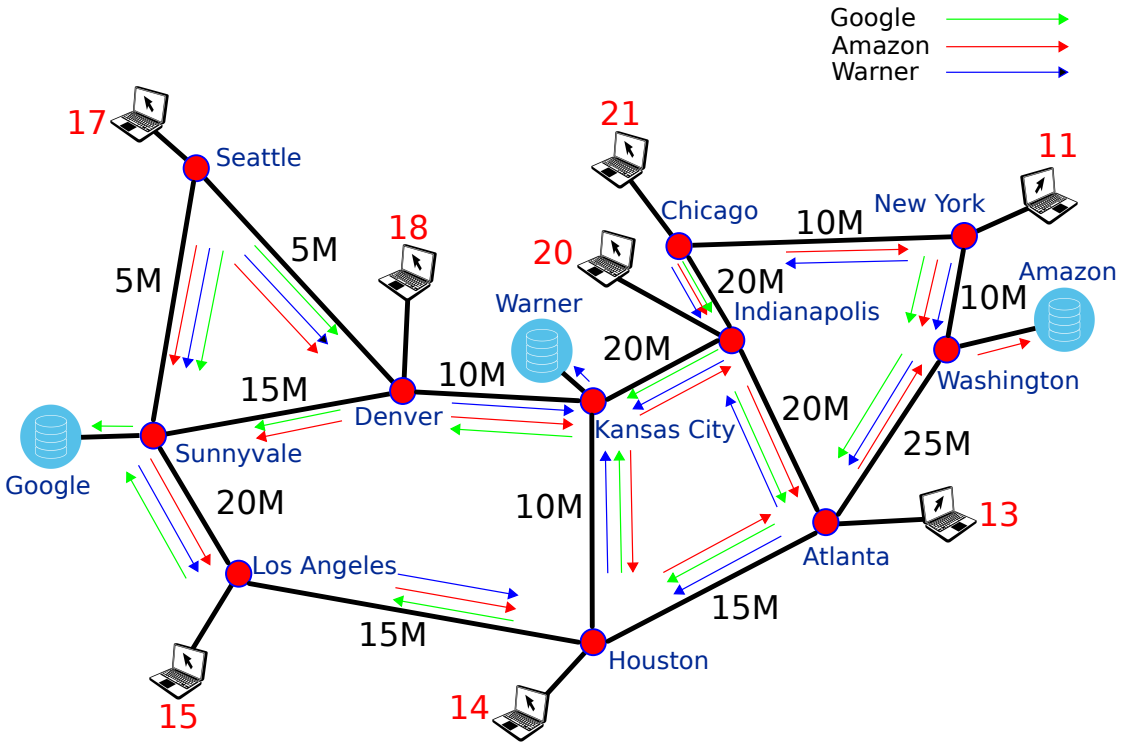


FIGURE 4.8 Topologie Abilene

une ville des États-Unis. Sur chacun de ces routeurs est connecté un consommateur ou un producteur. Au total, il y a 22 nœuds dans le réseau (dont 3 producteurs et 8 consommateurs). Chaque producteur fournit un contenu différent. Au total, 24 flux sont donc possibles (8 consommateurs pour 3 contenus). Nous avons simulé chaque pair de flux possible sur cette topologie et avons observé trois cas distincts :

— Les deux flux sont sur des parties séparées du réseau et ne se croisent jamais.
— Les deux flux se croisent mais ne partagent jamais de lien dans le même sens.
— Les deux flux se croisent et doivent se partager un ou plusieurs liens du réseau.

Pour tous les scénarios, le premier flux commence à $t = 0s$ et le second à $t = 5s$.

4.4.2 Cas où les flux ne se croisent pas

Un exemple de ce cas est lorsque le consommateur 21 demande le contenu Amazon et le consommateur 14 demande le contenu Google. La Figure 4.9 illustre les chemins utilisés par nos deux flux. Le flux rouge entre le consommateur 21 et le producteur du

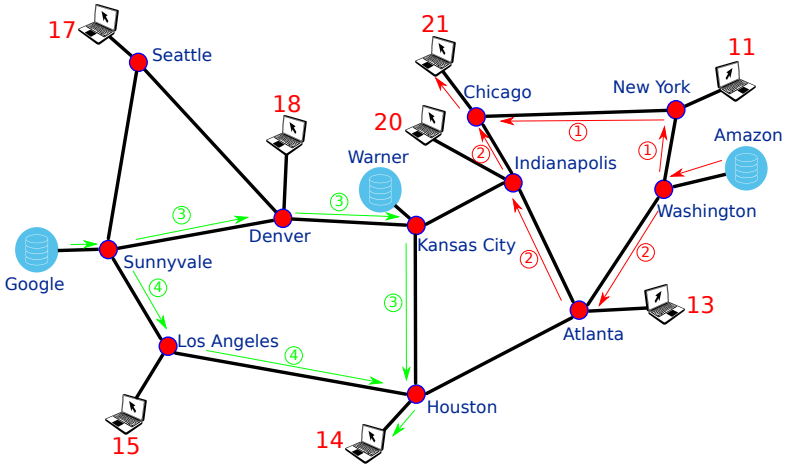


FIGURE 4.9 Consommateur 21 - Amazon et Consommateur 14 - Google

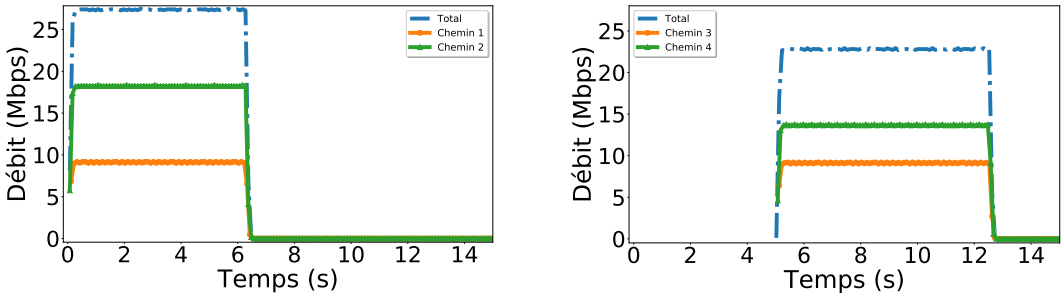
contenu Amazon dispose de deux chemins :

— le chemin passant par Washington, New-York et Chicago (chemin 1),
— et le chemin passant par Washington, Atlanta, Indianapolis et Chicago (chemin 2).

Le flux vert entre le consommateur 14 et le producteur du contenu Google dispose lui aussi de deux chemins :

— le chemin passant par Sunnyvale, Denver, Kansas City et Houston (chemin 3),
— et le chemin passant par Sunnyvale, Los Angeles et Houston (chemin 4).

La Figure 4.10 présente les résultats de ce scénario. Comme les flux ne se croisent jamais,



(a) Consommateur 21, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 4.10 Cas où les flux ne se croisent pas

le démarrage du second flux n'a pas d'impact sur les performances du premier. Les deux flux utilisent leur deux chemins respectifs de manière optimale. Le flux du consommateur 21 récupère environ 10Mbps sur le chemin 1 et 20Mbps sur le chemin 2, pour un total d'environ 30Mbps. Le flux du consommateur 14 récupère lui environ 10Mbps sur le chemin 3 et 15Mbps sur le chemin 4, pour un total d'environ 25Mbps. Tous les chemins sont utilisés dans leur totalité et le temps de téléchargement des contenus est minimal.

4.4.3 Cas où les flux se croisent dans des sens opposés

Dans le cas suivant, nous avons deux flux qui se croisent mais qui ne partagent pas un lien dans le même sens. Nous illustrons ce cas avec la paire de flux suivante : Consommateur 15 demandant le contenu Amazon et Consommateur 14 demandant le contenu Google. La Figure 4.11 illustre les chemins utilisés par nos deux flux. Le flux rouge entre le consommateur 15 et le producteur du contenu Amazon dispose d'un unique chemin, passant par Washington, Atlanta, Houston et Los-Angeles (chemin 1). Le flux vert entre le consommateur 14 et le producteur du contenu Google est le même que dans le cas précédent et dispose de deux chemins :

— le chemin passant par Sunnyvale, Denver, Kansas City et Houston (chemin 2),
— et le chemin passant par Sunnyvale, Los Angeles et Houston (chemin 3).

Lorsque le consommateur 15 demande le contenu Amazon, il utilise le lien reliant Los-Angeles et Houston en direction de Los-Angeles tandis que lorsque le consommateur 14 demande le contenu Google, il utilise aussi ce lien mais dans le sens inverse. La Figure 4.12 présente les résultats de ce scénario. Lors des cinq premières secondes, le premier flux est seul et utilise son unique chemin sans soucis. Le consommateur 15 arrive à obtenir un débit d'environ 15Mbps. À partir de $t = 5s$, le consommateur 14 démarre et commence

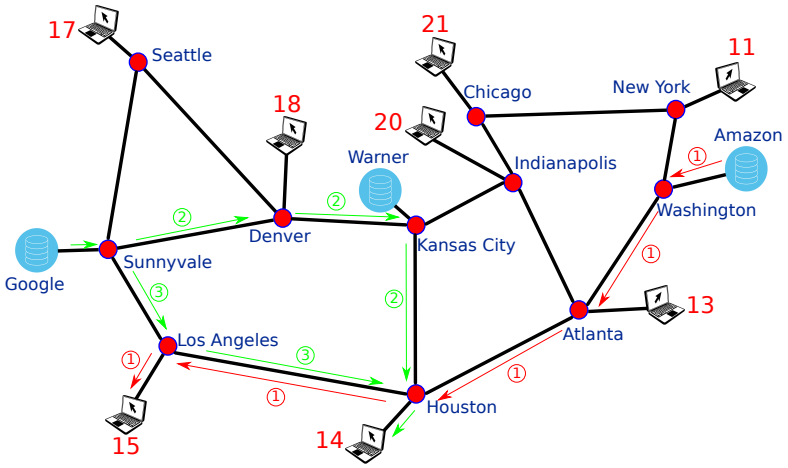
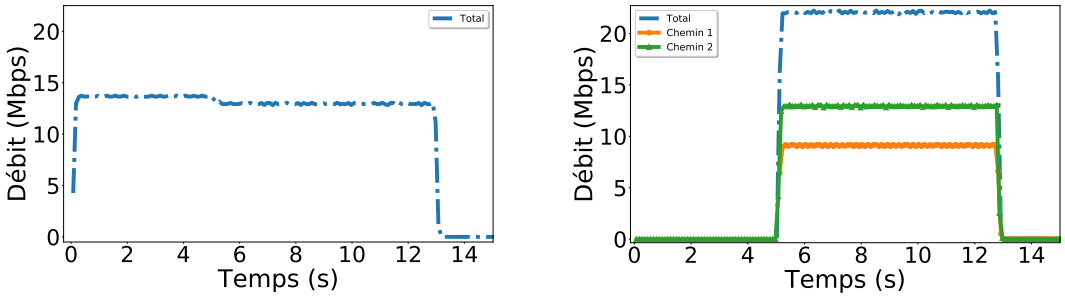


FIGURE 4.11 Consommateur 15 - Amazon et Consommateur 14 - Google



(a) Consommateur 15, Contenu Amazon

(b) Consommateur 14, Contenu Google

FIGURE 4.12 Cas où les flux se croisent dans des sens opposés

lui aussi à utiliser le lien reliant Los-Angeles et Houston. Même s'ils ne partagent pas le lien dans le même sens, les Data d'un flux transitent dans la même direction que les Interests de l'autre flux. Comme expliqué dans la section 4.3.1, les Interests prennent une part de la capacité du lien (estimée via l'Équation 4.3). Nous remarquons donc que le débit total obtenu par le premier flux diminue légèrement lorsque le second flux démarre. De même, le second flux obtient moins de débit sur le chemin 2 que s'il avait été seul, car les Interests du premier flux utilisent une petite part de la capacité du lien entre Los-Angeles et Houston (en direction de Houston). L'autre chemin (le chemin 1) du second flux n'est pas impacté et obtient bien environ 10Mbps. Au total, le second flux obtient environ 24.5Mbps et utilise au mieux les deux chemins qui lui sont disponibles. La prise en compte des Interests dans le calcul de l'utilisation de la capacité d'un lien nous permet d'être précis et de ne pas provoquer de sur-utilisation de cette capacité. De ce fait, les files d'attente des nœuds ne se remplissent pas et aucune perte ne se produit.

4.4.4 Cas où les flux partagent des portions de réseau

Le dernier cas est le cas le plus intéressant. En effet, nos deux flux vont être en compétition pour l'utilisation de certains liens du réseau. La Figure 4.13 illustre les chemins utilisés par nos deux flux. Le flux rouge entre le consommateur 18 et le producteur

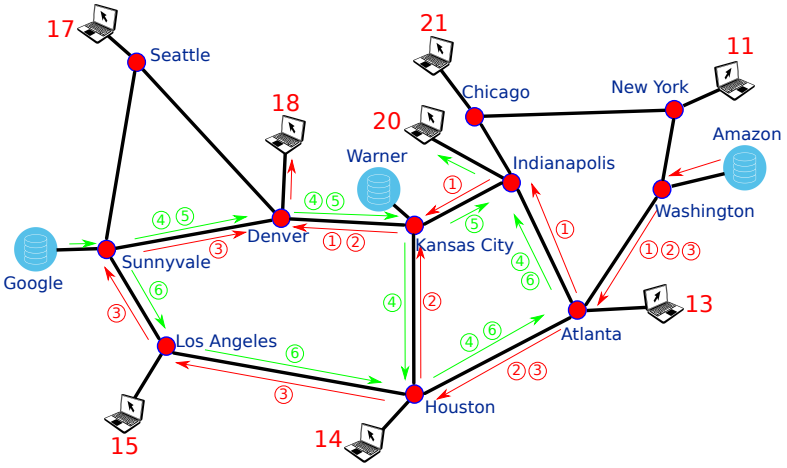


FIGURE 4.13 Consommateur 18 - Amazon et Consommateur 20 - Google

du contenu Amazon dispose de trois chemins :

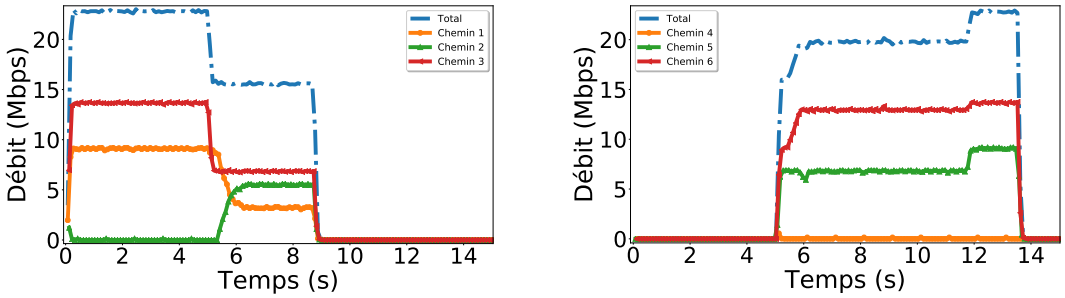
- le chemin passant par Washington, Atlanta, Indianapolis, Kansas City et Denver (chemin 1),
- le chemin passant par Washington, Atlanta, Houston, Kansas City et Denver (chemin 2),
- et le chemin passant par Washington, Atlanta, Houston, Los Angeles, Sunnyvale et Denver (chemin 3).

Le flux vert entre le consommateur 20 et le producteur du contenu Google dispose lui aussi de trois chemins :

- le chemin passant par Sunnyvale, Denver, Kansas City, Houston, Atlanta et Indianapolis (chemin 4),
- le chemin passant par Sunnyvale, Denver, Kansas City et Indianapolis (chemin 5),
- et le chemin passant par Sunnyvale, Los Angeles, Houston, Atlanta et Indianapolis (chemin 6).

Les deux flux sont donc en concurrence sur les liens reliant Sunnyvale à Denver et Indianapolis à Atlanta. En réalité, le sous-flux 3 du premier flux est en concurrence avec les sous-flux 4 et 5 du second flux sur le lien Sunnyvale-Denver et le sous-flux 1 est en concurrence avec les sous-flux 4 et 6. Le sous-flux 2 est le seul sous-flux à ne pas être en

concurrence avec un sous-flux de l'autre flux. La Figure 4.14 présente les résultats de ce scénario. Lors des cinq premières secondes, le premier flux est seul sur le réseau et



(a) Consommateur 18, Contenu Amazon (b) Consommateur 20, Contenu Google

FIGURE 4.14 Cas où les flux partagent des portions de réseau

atteint un débit d'environ 25mbps, via l'utilisation du chemin 1 à hauteur de 10Mbps et du chemin 3 à hauteur de 15Mbps. Le chemin 2 n'est pas utilisé car il partage des portions de chemin avec les autres chemins. Son utilisation réduirait les deux autres chemins à la fois et donc le débit total obtenu. À $t = 5s$, le second flux commence. Dès qu'il est détecté, les capacités des liens utilisés par les deux flux (Sunnyvale-Denver et Indianapolis-Atlanta) sont partagées également. Le premier flux n'utilisait déjà que la moitié de la capacité du lien reliant Indianapolis à Atlanta (10Mbps sur les 20Mbps disponibles) et reste donc à une utilisation de 10Mbps dans un premier temps (chemin 1). Le second flux obtient lui aussi les 10Mbps restants (chemin 6). Pour le lien reliant Sunnyvale à Denver, la capacité de 15Mbps est partagée en deux parts de 7,5Mbps pour chaque flux (pour les chemins 3 et 5). Les deux flux ont ainsi un débit total d'environ 17.5Mbps. La distribution aurait pu en rester là sauf que la réduction de l'utilisation du chemin 3 permet un basculement de débit du chemin 1 vers le chemin 2 actuellement inutilisé. Ce basculement, même s'il n'apporte rien au débit du premier flux (qui reste à 17.5Mbps), permet de libérer de la capacité sur le lien reliant Indianapolis à Atlanta. Le second flux peut alors augmenter son débit sur le chemin 6 qui l'emprunte et obtenir un débit total de 22.5Mbps. Son débit a été augmenté sans réduire le débit du premier flux et aucun chemin ne peut être utilisé plus sans réduire l'utilisation d'au moins un autre chemin. Lorsque le premier flux se termine, le second flux peut utiliser le chemin 5 totalement et obtenir un débit global d'environ 25Mbps. Comme pour le premier flux et son chemin 2, le chemin 4 n'est pas utilisé car son utilisation engendrerait une baisse de l'utilisation des deux autres chemins et donc du débit global. Dans l'implantation, un flux est en réalité considéré actif pendant 3 secondes après avoir reçu la dernière Data. C'est pourquoi l'augmentation des débits du second flux est autant décalée par rapport à

la terminaison du premier flux.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une évolution simple et cohérente de notre solution Cooperative Congestion Control. Nous prenons maintenant en compte que, pour un nœud, plusieurs interfaces sont disponibles pour atteindre le consommateur. Contrairement aux topologies arborescentes, un nœud peut recevoir plusieurs objectifs pour un même flux et doit donc envoyer plusieurs contraintes en retour. Le nœud doit maintenant être capable de distribuer l'ensemble des contraintes entrantes sur les chemins disponibles et s'assurer que la répartition entre les objectifs de chaque flux sur une interface soit correcte. C'est déjà ce que le nœud faisait dans le sens inverse pour les objectifs et les contraintes respectivement. Nous proposons donc une architecture symétrique sur l'Upstream et le Downstream composé de six algorithmes (trois chacun). Deux algorithmes servent à s'assurer que la répartition inter-flux sur une interface est correcte et à la corriger en cas de problème. Le troisième algorithme permet quant à lui de faire la répartition inter-chemin pour un flux et s'assure du respect de nos lois de Non Création d'Objectifs et de Non Suppression de Contraintes, qui ont été adaptées et renommées pour l'occasion. Nous introduisons cependant de la dissymétrie dans l'implantation des algorithmes gérant la répartition inter-flux via la notion de satisfaction, qui n'a de sens que pour le Downstream. Cela nous permet de sonder le réseau avec les objectifs sans restriction supplémentaire et de chercher à satisfaire les flux avec les contraintes en priorité. Les algorithmes gérant la répartition inter-chemins utilisent ensuite un éventuel ρ pour basculer le débit d'un chemin vers un autre et laisser la place à un autre flux. Bien que l'architecture ait évolué, elle reste modulable et d'autres stratégies d'allocations restent possibles à implanter. Comme pour la version et le chapitre précédent, notre stratégie d'allocation est de proposer une équité Max-Min locale sur chaque interface. Elle prend place dans les algorithmes gérant la répartition inter-flux. Comme vu au chapitre précédent, une équité Max-Min globale n'est pour l'instant pas envisageable par manque d'information des nœuds mais reste une motivation pour la suite des travaux.

Chapitre 5

Contrôle de Congestion sur réseaux Satellite

Sommaire

5.1 Introduction

Dans ce chapitre, nous nous intéressons à l'utilisation des ICNs sur des réseaux satellite. Les réseaux satellite ont des caractéristiques qui posent autant de questions qu'elles offrent d'opportunités. En effet, les liens satellite permettent d'atteindre à moindre coût un grand nombre de stations de base (utilisateurs ou point d'accès à des réseaux de plus grande envergure) mais subissent un délai non négligeable. Par exemple, pour un lien satellite géostationnaire, la zone de couverture représente environ un tiers de la surface de la planète mais le délai pour aller de la passerelle à la station de base est d'environ 250ms (et donc 500ms pour un aller-retour). Dans une contribution passée [7], nous avons étudié l'utilisation de CDNs dans des réseaux satellite et montré que les caches permettent d'améliorer la QoE des utilisateurs pour des applications comme le streaming vidéo, le téléchargement de fichier (via P2P) et même la navigation web. En effet, le fait de rapprocher le contenu de l'utilisateur (même simplement du côté de la gateway satellite) permet de réduire le délai de bout en bout et d'améliorer les performances des mécanismes de contrôle de congestion utilisant une fenêtre. Nous adoptons ici une approche différente et étudions le contrôle de congestion dans NDN sur des réseaux présentant un lien satellite. Nous commençons par discuter des différentes approches potentielles puis comparons les performances des solutions de l'état de l'art et de notre solution, Cooperative Congestion Control, en fonction du placement du lien satellite dans le réseau.

5.2 Discussion sur les différentes approches

Dans cette section, nous discutons des différentes approches permettant de faire du contrôle de congestion dans un réseau satellite, en utilisant la pile NDN. Comme nous l'avons vu dans la section 1.4.5, de nombreuses solutions utilisées dans les réseaux terrestres sont des adaptations de TCP. Cependant, les approches fondées sur les fenêtres de congestion ont du mal à utiliser efficacement les satellites géostationnaires. En effet, un long délai implique une augmentation lente de la taille de la fenêtre. De plus, la bande passante élevée de la liaison satellite couplée à la diminution multiple de la fenêtre induit une diminution importante du débit de l'utilisateur lors de la détection d'une perte. Finalement, ces deux facteurs combinés conduisent à des fluctuations longues et inefficaces de la fenêtre de congestion. Pour atténuer ce phénomène, les opérateurs satellite utilisent généralement des *Performance-Enhancing Proxies* [67] (PEP). Le principe consiste à diviser la communication TCP en plusieurs connexions pour isoler le lien satellite. La

partie satellite peut alors utiliser des paramètres spécifiques définis par l'opérateur. Le point positif est que cette méthode est transparente du point de vue de l'utilisateur final et qu'il n'est pas nécessaire d'utiliser une version spécifique de TCP. Une adaptation de cette approche doit être envisagée, car les mécanismes fondés sur les fenêtres de congestion sont également utilisés dans NDN. En raison de la propriété *receiver-driven* de NDN, cette approche semble cependant difficile à mettre en œuvre. En effet, dans les PEPs, les proxies font souvent du "spoofing", à savoir qu'ils cassent la connexion en envoyant un accusé de réception à l'avance. Le but est de falsifier le RTT calculé par l'expéditeur et ainsi d'augmenter plus rapidement la fenêtre de congestion. Avec NDN, il n'y a pas d'accusé de réception. Dans les mécanismes utilisant une fenêtre de congestion, ce sont les paquets Data qui jouent ce rôle. Ainsi, pour un PEP dans NDN, il faudrait qu'il soit capable d'envoyer les Data avant de les recevoir. Une solution pourrait être d'envoyer de faux paquets Data afin de déclencher une augmentation plus rapide de la taille de la fenêtre de congestion et, lorsque la vraie Data arrive, de l'envoyer au consommateur. Mais cela remettrait en question une propriété essentielle de NDN : le *flow balance*. Comme nous l'avons vu dans la Section 1.3, NDN définit en effet le *flow balance* comme le fait que la réception d'un Interest induit l'émission d'au plus une Data. De plus, les nœuds sur le chemin sont supposés effacer les informations du chemin inverse d'un Interest dès que la Data correspondante a emprunté le chemin inverse. Nos fausses et vraies Data ne respecteraient pas ces deux propriétés de NDN. C'est pourquoi nous n'avons pas été plus loin dans cette approche.

Une autre approche consiste à augmenter la taille initiale de la fenêtre de congestion et à rythmer ces premières émissions [68]. Les auteurs expliquent que cette variante de TCP marche aussi bien sur réseaux terrestre que satellite et permet de s'affranchir des PEPs. Notre solution, Cooperative Congestion Control, présentée précédemment, adopte une approche similaire avec la gestion via le rythme des Interests, mais à la fois nouvelle puisqu'elle n'utilise pas de fenêtre de congestion. Cette nouvelle approche devrait nous permettre de ne plus souffrir des défauts des solutions utilisant justement une fenêtre de congestion. En effet, cette métrique est directement transposable en débit et son augmentation n'est pas liée au RTT entre le consommateur et le producteur.

5.3 Comparaison des performances

Dans cette section, nous comparons les performances de notre solution et du couple de solution de l'état de l'art "ICP + PCON-FS" (nous avons montré précédemment que c'est le couple de solutions qui obtient les meilleurs résultats dans l'état de l'art, cf

1.4.5). Pour cela, nous reprenons la topologie Abilene du chapitre précédant auquel nous ajoutons des liens satellite ou remplaçons des liens existants par des liens satellite (voir Figure 5.1, les liens en pointillés étant les liens qui sont terrestres dans tous les scénario). Nous reprenons ainsi nos trois cas d'études (cas où les flux ne se croisent pas, cas où ils se croisent en sens inverse et dans le même sens) mais dans quatre scénarios différents :

— lorsque nous remplaçons le lien d'accès au réseau des consommateurs par un lien satellite (liens en rouge sur la figure) ;
— lorsque nous remplaçons le lien d'accès au réseau des producteurs par un lien satellite (liens en vert sur la figure) ;
— lorsque nous remplaçons un lien du réseau par un lien satellite (liens en bleu sur la figure) ;
— et lorsque nous ajoutons un lien satellite entre les consommateurs et producteurs (liens en mauve sur la figure).

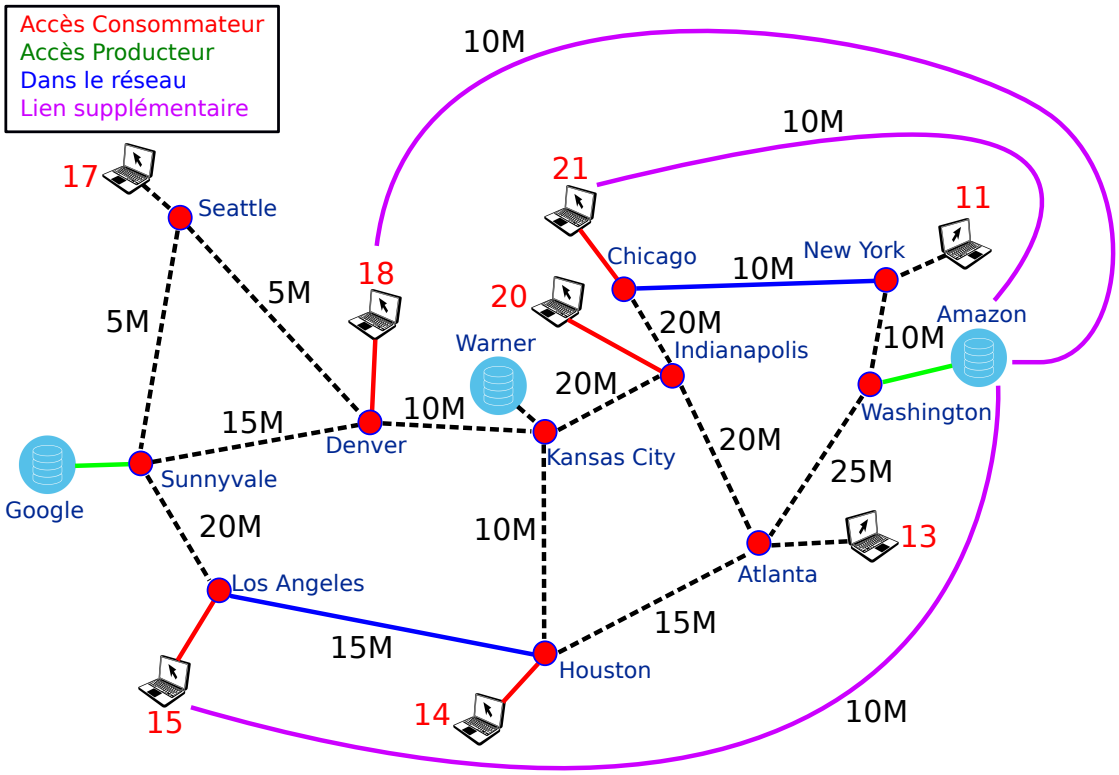


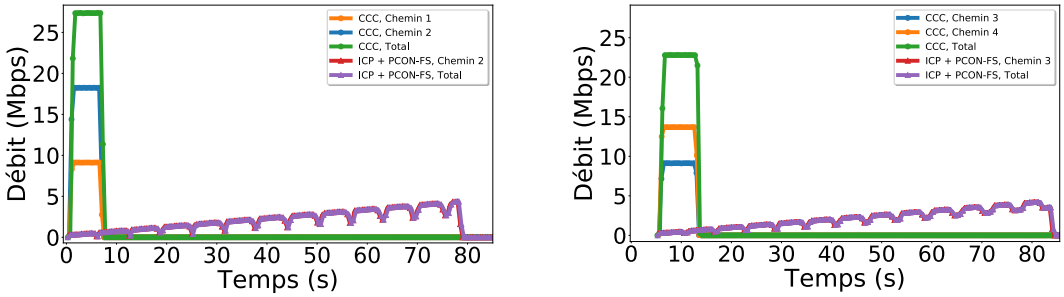
FIGURE 5.1 Topologie Abilene avec liens Satellite

Ces scénarios sont à nouveau conduits sur ndnSIM et le lien satellite est représenté par un lien subissant un délai de 250ms (lien satellite géo-stationnaire). Les débits sont identiques aux liens qu'ils remplacent ou de 10Mbps pour les liens supplémentaires.

5.3.1 Lien Satellite en accès sur les consommateurs

Ce premier scénario est un scénario très fréquent et permet de voir les performances d'utilisateurs ne disposant que d'une connexion satellitaire. Les liens satellite sont les liens en rouge sur la Figure 5.1 et permettent donc à nos utilisateurs d'accéder au réseau.

La Figure 5.2 présente les résultats de ce scénario dans le premier cas d'étude (cf Figure 4.9). Pour rappel, le consommateur 21 demande le contenu du producteur Amazon à partir de $t = 0$ et le consommateur 14 demande le contenu du producteur Google à partir de $t = 5$. Les deux consommateurs récupèrent leur contenu via deux chemins et sont sur des parts différentes du réseau. Comme les liens satellite sont les liens d'accès des consommateurs, l'ensemble de leur chemin est impacté par l'augmentation du délai. Les



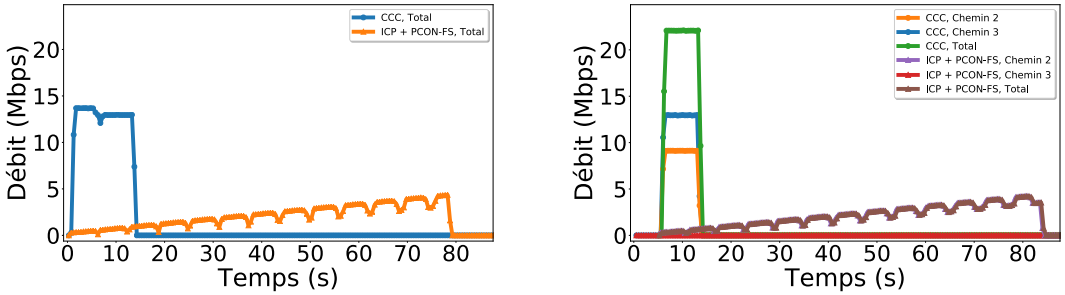
(a) Consommateur 21, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.2 Cas où les flux ne se croisent pas

performances de CCC sont quasiment inchangées. Les deux consommateurs réussissent à utiliser leurs deux chemins respectifs dans la totalité et le temps de téléchargement du contenu est minimisé. Le démarrage est cependant légèrement plus lent que lorsqu'il n'y a pas de lien satellite. Ceci est du aux échanges d'objectifs et de contraintes qui prennent plus de temps à parcourir les chemins. Les performances du couple "ICP + PCON-FS" sont très faibles pour nos deux consommateurs (les temps de téléchargement sont d'environ 80s contre une dizaine pour CCC). Les courbes sont similaires entre les deux consommateurs. La fenêtre de congestion n'augmente pas assez vite à cause du lien satellite. Seulement un des deux chemins disponibles est utilisé car avec PCON-FS, les autres chemins commencent à être utilisés lorsqu'une marque de congestion est reçue. Le téléchargement du contenu se termine avant d'avoir saturé le premier lien et aucune marque n'est émise par les nœuds du réseau. Au final, le réseau n'est pas utilisé à son plein potentiel et le temps de téléchargement des contenus est très long.

La Figure 5.3 présente les résultats du premier scénario dans le deuxième cas d'étude (cf Figure 4.11). Pour rappel, le consommateur 15 demande le contenu du producteur

Amazon à partir de $t = 0$ et le consommateur 14 demande le contenu du producteur Google à partir de $t = 5$. Le consommateur 15 récupère son contenu via un chemin tandis que le consommateur 14 récupère le sien via deux chemins. Le chemin 1 du consommateur 15 croise le chemin 3 du consommateur 14 sur le lien reliant Los Angeles à Houston, mais les flux sont dans des sens inverses. Comme les liens satellite sont les liens d'accès des consommateurs, l'ensemble de leur chemin est impacté par l'augmentation du délai. Les conclusions du premier cas d'étude sont valables ici aussi : CCC fonctionne bien

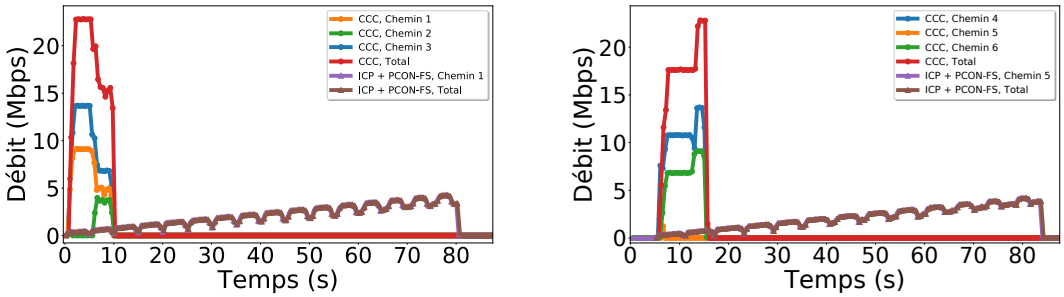


(a) Consommateur 15, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.3 Cas où les flux partagent des portions de réseau

malgré un temps de démarrage légèrement plus lent que sans satellite et le couple "ICP + PCON-FS" n'arrive pas à utiliser le réseau de manière efficace. CCC s'adapte bien au fait que les flux se croisent dans des sens opposés. En effet, quand il le remarque, il réserve une place pour les Interests du flux de sens opposé. Au vu des faibles performances du couple "ICP + PCON-FS", le problème ne se pose pas.

Finalement, la Figure 5.4 présente les résultats du premier scénario dans le troisième cas d'étude (cf Figure 4.13). Pour rappel, le consommateur 18 demande le contenu du producteur Amazon à partir de $t = 0$ et le consommateur 20 demande le contenu du producteur Google à partir de $t = 5$. Les deux consommateurs récupèrent leur contenu via trois chemins qui se croisent plusieurs fois dans le réseau. Comme les liens satellite sont les liens d'accès des consommateurs, l'ensemble de leur chemin est impacté par l'augmentation du délai. Les performances du couple "ICP + PCON-FS" sont similaires aux deux premiers cas et cela pour les mêmes raisons. De plus, le problème de basculement de flux ne se pose pas ici puisque chacun des flux peine déjà à utiliser un unique chemin. Les performances de CCC sont à nouveau très satisfaisantes mais l'équilibre du basculement ne se passe pas idéalement. Lorsque le premier flux est seul, il utilise l'ensemble des chemins disponibles comme attendu. Lorsque le deuxième flux commence, ils se partagent bien également les liens reliant Sunnyvale à Denver et Indianapolis à Atlanta, atteignant ainsi un débit d'environ 17.5Mbps chacun. Toutefois, le premier flux ne bascule pas vers



(a) Consommateur 18, Contenu Amazon

(b) Consommateur 20, Contenu Google

FIGURE 5.4 Cas où les flux se croisent dans des sens opposés

son second chemin de manière optimale afin de libérer de la bande passante au second flux. Dans cette configuration, plusieurs équilibres de basculement sont en effet possibles. L'état final dépend de quand sont prises les décisions et de quand sont reçus les nouvelles informations (sur les objectifs ou contraintes). La réception des nouvelles informations dépend du délai sur les liens et ce problème n'est pas spécifique aux réseaux satellite. Le temps de téléchargement pour le premier flux n'est pas impacté car il obtient bien un débit d'environ 17.5Mbps. Le deuxième flux n'obtient pas autant de débit que lors d'un basculement optimal mais obtient malgré tout bien les 17.5Mbps grâce à la mise en place de l'équité dans CCC. Une vision globale semble ici nécessaire pour que les nœuds se rendent compte que l'équilibre n'est pas optimal. Une des ambitions de CCC était de rester décentralisée et nous n'envisageons pas de remettre celle-ci en jeu. Une solution envisageable serait d'introduire un sondage aléatoire des opportunités disponibles pour explorer de nouveaux équilibres. Cette recherche se ferait une fois l'équité locale atteinte et ne mettrait donc pas en péril les débits minimums atteints ici. Nous discuterons de cette piste dans les perspectives de ce mémoire.

5.3.2 Lien Satellite en accès sur les producteurs

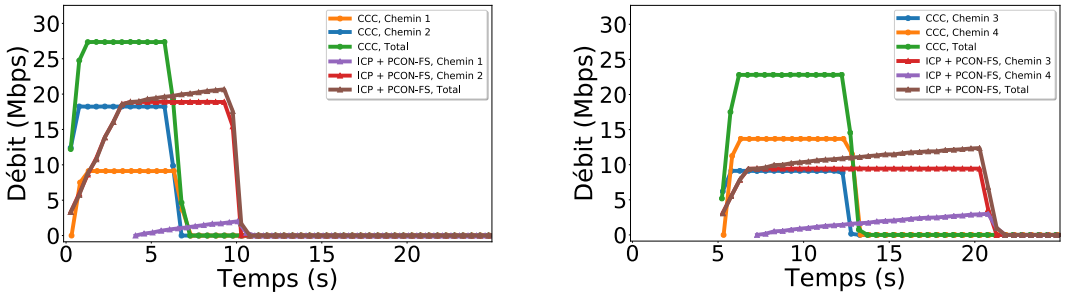
Dans ce deuxième scénario, nous avons remplacé le lien d'accès des producteurs par un lien satellite de débit équivalent. Les liens satellite sont les liens en vert sur la Figure 5.1 et permettent donc aux producteurs d'accéder au réseau. Les trois différents cas montrent des résultats similaires aux scénarios avec un lien satellite en accès des utilisateurs. En effet, l'augmentation des délais est la même sur l'ensemble des chemins pour tous les flux. Le positionnement du lien satellite n'est donc pas important, à la condition que tous les chemins soient affectés globalement de la même manière. À nouveau, CCC réussit avec succès à utiliser l'ensemble des chemins tandis que le couple de solution "ICP +

PCON-FS" n'arrive jamais à utiliser pleinement son premier chemin.

5.3.3 Lien Satellite dans le réseau

Dans le troisième scénario, nous avons remplacé certains liens du réseau de cœur par des liens satellite. Les liens satellite sont les liens en bleu sur la Figure 5.1. Le choix a été fait pour que chacun des flux de nos différents cas subisse le délai satellite sur un seul de leurs chemins. Ainsi, nous pouvons voir l'impact d'un lien satellite sur les communications lorsqu'il y a aussi des chemins non concernés.

La Figure 5.5 présente les résultats de ce scénario dans le premier cas d'étude (voir la Figure 4.9). Le flux du consommateur 21 est impacté par le lien satellite reliant Chicago à New York. Seul le chemin 1 subit le délai de ce lien satellite, le délai du chemin 2 reste inchangé. Le flux du consommateur 14 est lui aussi impacté par un lien satellite mais par celui reliant Los Angeles à Houston. Comme pour le premier flux, un seul de ses chemins subit le délai satellite (son chemin 4). Comme pour les scénarios



(a) Consommateur 21, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.5 Cas où les flux ne se croisent pas

précédents, les performances de CCC sont quasiment inchangées. La différence ici est que seul le chemin avec le lien satellite voit son démarrage légèrement retardé. Pour le couple "ICP + PCON-FS", la taille de la fenêtre de congestion de nos deux consommateurs augmente suffisamment rapidement pour utiliser leur chemin non satellite mais peine ensuite à utiliser leur second chemin. Le chemin non satellite aide malgré tout à accélérer l'utilisation du lien satellite car les paquets Data reçus via ce chemin contribuent à l'augmentation de la fenêtre de congestion. L'augmentation de débit sur le chemin avec le lien satellite est en effet plus rapide que lors des précédents scénarios (où il n'y avait pas de chemin non satellite pour aider). Nous pouvons également noter que PCON-FS utilise le chemin le mieux classé par l'algorithme de routage. Si en termes de délai, il est logique de mieux classer les chemins non satellite, une autre logique pourrait en décider

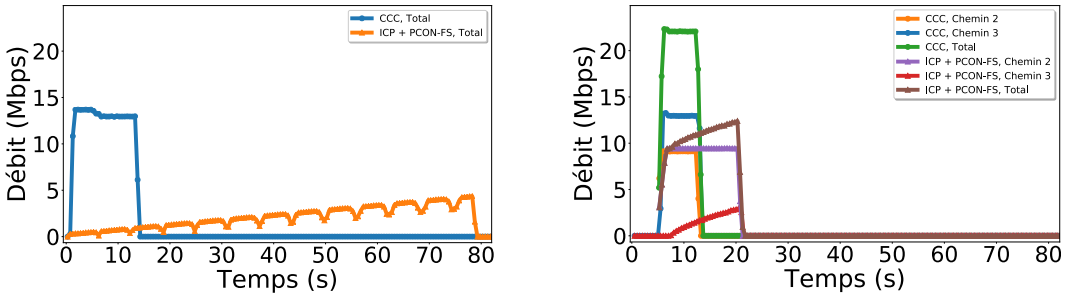
autrement. Les performances seraient équivalentes à celles des scénarios précédents si le chemin satellite était utilisé en premier.

Le deuxième cas d'étude (préalablement présenté dans le chapitre précédent, cf Figure 4.11) confirme les conclusions précédentes. Si pour CCC, les résultats s'approchent de l'optimum présenté dans le chapitre 2, "ICP +PCN-FS" en est très loin. En effet, pour cette dernière :

— le consommateur 15 récupère son contenu via un unique chemin affecté par un lien satellite et n'arrive pas à utiliser efficacement le débit qui est disponible ;

— le consommateur 14 récupère son contenu via deux chemins dont un affecté par un lien satellite, arrive à utiliser son premier chemin convenablement mais peine avec le second.

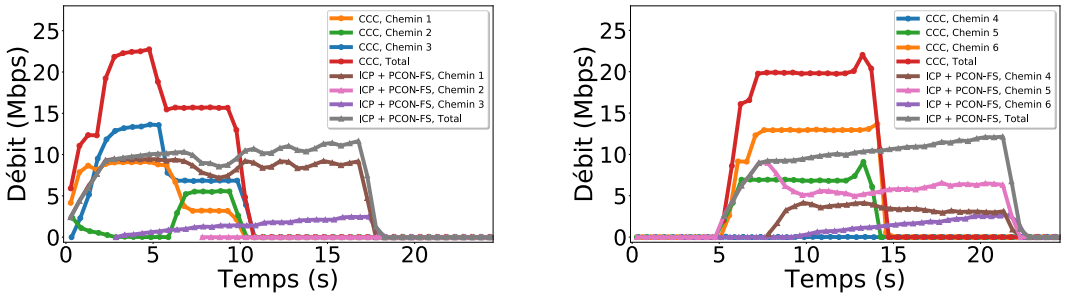
Les résultats sont présentés sur la Figure 5.6.



(a) Consommateur 15, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.6 Cas où les flux se croisent dans des sens opposés

Finalement, la Figure 5.7 présente les résultats du troisième scénario dans le troisième cas d'étude (décrit dans la Figure 4.13). Les deux consommateurs ont trois chemins chacun à leur disposition, dont un seul utilise le lien satellite reliant Los Angeles à Houston. Avec CCC, les consommateurs utilisent convenablement les chemins disponibles. Pour le consommateur 18, l'utilisation du chemin 3 est visiblement différée par rapport à celle du chemin 1 (le chemin 2 n'étant pas utilisé quand le flux est seul). C'est en effet sur ce chemin 3 qu'est le lien satellite (entre Los Angeles et Houston). Nous constatons le même phénomène avec le débit sur le chemin 6 du consommateur 20 qui passe aussi par ce lien. À nouveau, les performances du couple "ICP + PCON-FS" sont similaires aux deux premiers cas : le chemin avec le lien satellite a une augmentation de débit trop lente pour être utilisé convenablement. La *forwarding strategy* PCON-FS a des difficultés à départager les chemins qui se recoupent entre eux. Pour le consommateur 20, le chemin 5 est initialement utilisé jusqu'à ce que Denver émette des marques de congestion. Lorsqu'il les reçoit, Indianapolis fait du partage de charge entre ses liens le reliant à Kansas City et



(a) Consommateur 18, Contenu Amazon (b) Consommateur 20, Contenu Google

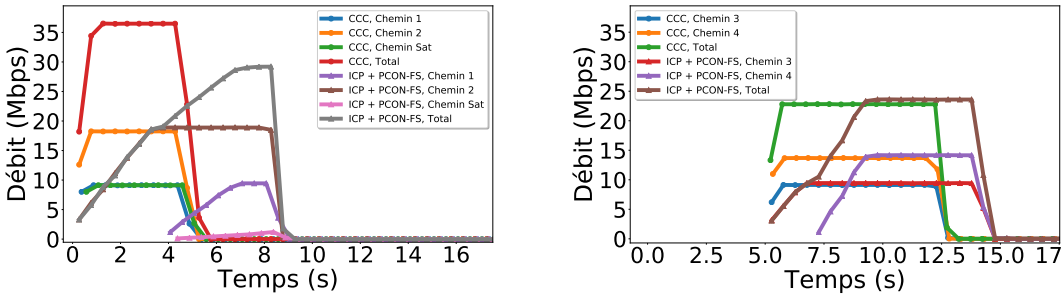
FIGURE 5.7 Cas où les flux partagent des portions de réseau

Atlanta. Le deuxième chemin utilisé est ensuite le chemin 4 (le délai étant plus court que sur le chemin 6) mais passe lui aussi par Denver et ses émissions de marque de congestion. Après réception de ces marques, Houston commence alors à utiliser le chemin 6 aussi. PCON-FS n'est pas capable de démêler l'utilisation de chemins dans un tel cas.

5.3.4 Lien Satellite reliant directement consommateur et producteur

Dans ce quatrième et dernier scénario, nous utilisons le réseau Abilene tel qu'il est au chapitre précédent et nous supposons que les consommateurs ont en supplément une connectivité satellite directe avec les producteurs. Les liens satellite sont les liens en mauve sur la Figure 5.1 et ont un débit de 10Mbps. Ce n'est pas forcément représentatif d'un lien satellite à haut débit mais nous voulions que le débit corresponde au débit des autres liens du réseau et ce afin de pouvoir comparer les résultats sans être influencé par les débits.

La Figure 5.8 présente les résultats de ce scénario dans le premier cas d'étude (cf Figure 4.9). Seul le consommateur 21 possède une connexion satellite supplémentaire, le consommateur 14 utilise le réseau Abilene de façon classique. Le flux du consommateur 21 peut donc utiliser trois chemins distincts ici : les chemins 1 et 2 qui utilisent le réseau Abilene et le chemin satellite qui utilise le lien satellite supplémentaire. Les deux flux sont sur des parts du réseau qui ne se croisent pas et le lien satellite supplémentaire entre le consommateur 21 et le producteur Amazon ne remet pas cela en cause. Par conséquent, le démarrage du deuxième flux n'impacte pas le premier flux et inversement la fin du premier flux n'impacte pas le deuxième. Avec CCC, le premier flux utilise les chemins 1 et 2 entièrement mais aussi le nouveau chemin satellite. Le consommateur obtient ainsi un débit d'environ 40Mbps (10 via le chemin 1, 20 via le chemin 2 et 10 via le chemin satellite). Avec le couple "ICP + PCON-FS", le premier flux prend un peu

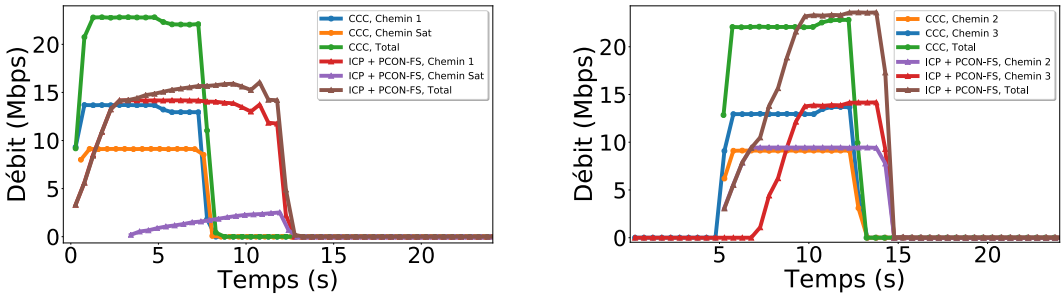


(a) Consommateur 21, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.8 Cas où les flux ne se croisent pas

de temps pour utiliser pleinement les 20Mbps du chemin 2 puis, grâce à l'émission des marques de congestion, commence à utiliser le chemin 1 et le chemin satellite. Cependant, l'augmentation de débit est plus lente à cause des délais et le chemin satellite n'est finalement que très peu utilisé. Le temps de téléchargement est donc plus long et le débit moyen plus bas que celui obtenu par CCC. Le deuxième flux n'empruntant pas de lien satellite, CCC est une nouvelle fois plus rapide mais la différence est due au démarrage du couple "ICP + PCON-FS".

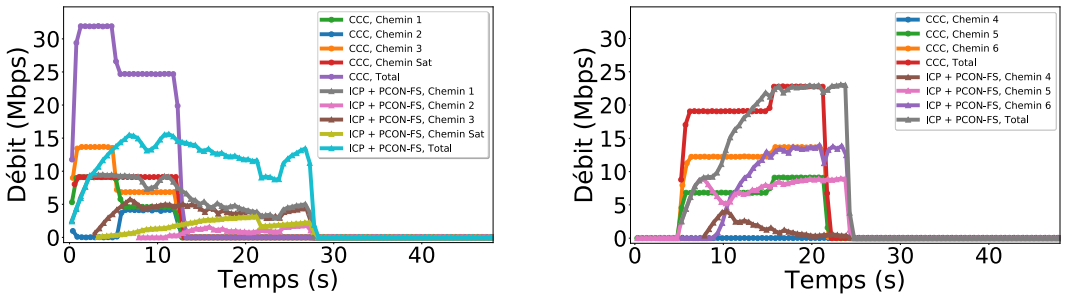
À nouveau, le deuxième cas d'étude (cf Figure 4.11) confirme les conclusions du premier et le fait que les deux flux se croisent ici ne pose pas de problème particulier. Seul le consommateur 15 possède une connexion satellite supplémentaire, alors que le consommateur 14 utilise le réseau Abilene de façon classique. Avec CCC, les performances sont très satisfaisantes et le premier flux peux bénéficier des 10Mbps apportés par le chemin satellite. Avec "ICP + PCON-FS", le chemin satellite n'arrive pas à être utilisé convenablement et seul le chemin classique est utilisé complètement. Les résultats sont présentés sur la Figure 5.9.



(a) Consommateur 15, Contenu Amazon (b) Consommateur 14, Contenu Google

FIGURE 5.9 Cas où les flux se croisent dans des sens opposés

Finalement, la Figure 5.10 présente les résultats de ce dernier scénario dans le troisième cas d'étude (cf Figure 4.13). Les deux consommateurs (18 et 20) peuvent utiliser trois chemins du réseau terrestre, mais le consommateur 18 dispose en plus d'une connexion satellite directe avec le producteur. Pour éviter que le téléchargement du contenu ne se termine sans que nous ayons suffisamment d'informations sur le comportement du couple "ICP + PCON-FS", nous avons doublé la taille du contenu pour ces dernières simulations ("ICP + PCON-FS" et CCC pour pouvoir les comparer). Ainsi, la taille du contenu passe de 20Mo à 40Mo rallongeant ainsi le temps de téléchargement pour nous permettre de mieux analyser l'utilisation des chemins. Avec CCC, le premier flux



(a) Consommateur 18, Contenu Amazon (b) Consommateur 20, Contenu Google

FIGURE 5.10 Cas où les flux partagent des portions de réseau

arrive à atteindre environ 35Mbps lorsqu'il est seul et 27.5Mbps après le démarrage du second flux et utilise les capacités du chemin additionnel comme attendu. Pour le couple "ICP + PCON-FS", l'utilisation du chemin satellite démarre lors de la réception des premières marques de congestion, une fois que le chemin 1 est complètement utilisé. Nous constatons une baisse dans le débit total du flux à partir d'environ $t = 11s$. Aucune perte ne se produit et le consommateur 18 ne réduit jamais la taille de sa fenêtre de congestion durant la simulation. À cause du deuxième flux, Sunnyvale émet des marques de congestion en continu. Lorsque le consommateur 18 reçoit ces marques, il bascule petit à petit l'émission des Interests sur le chemin satellite. Les deux autres chemins sont ainsi moins utilisés mais la perte en débit n'est pas compensée par l'utilisation du chemin satellite. Au final, le lien satellite supplémentaire ralentit le temps de téléchargement du contenu.

5.4 Conclusion

Dans ce chapitre, nous avons étudié les performances de notre solution Cooperative Congestion Control et des solutions de l'état de l'art sur des réseaux satellite. Nous avons

évalué ces solutions sur ndnSIM et sur différents scénarios où le placement du lien satellite diffère. Comme attendu, ICP, une solution utilisant une fenêtre de congestion s'accommode très mal des liens satellite. Lorsque le lien satellite est en accès du consommateur ou du producteur, les performances du couple "ICP + PCON-FS" sont médiocres. L'augmentation de la taille de la fenêtre de congestion est extrêmement longue et ne permet pas d'utiliser convenablement l'ensemble des chemins. Quand le lien satellite est dans le réseau, les chemins non satellite empruntés avant le chemin satellite ne sont pas impactés et sont utilisés efficacement. Les autres ont cependant du mal à être bien utilisés à cause du lien satellite. Finalement, dans le cas où un consommateur dispose d'une double connectivité, satellite en plus de terrestre, nous constatons un débit global plus faible car la *forwarding strategy* de PCON bascule l'émission des Interests sur le chemin satellite. L'utilisation du couple "ICP + PCON-FS" est donc à éviter sur des réseaux satellite et lors de multi-connectivité (terrestre + satellite). De son côté et grâce à son approche basée sur le rythme d'Interests, CCC a de très bons résultats sur chacun des scénarios. Les liens satellite ne perturbent pas le comportement de CCC et voient simplement leur démarrage retardé. L'ajout d'une connectivité satellite en supplément d'une terrestre permet ainsi d'augmenter le débit global auquel le consommateur récupère son contenu. Lorsqu'un basculement de chemins d'un flux est possible pour améliorer le débit d'un autre flux, nous constatons cependant que des équilibres non optimaux peuvent se produire. L'équilibre atteint dépend à la fois du moment où les décisions sont prises et de celui d'arrivée de la signalisation. L'équité sur les liens est toujours conservée mais la maximisation de l'utilisation du réseau à partir de ce point-là n'est pas garantie. Ce problème semble complexe et de futurs travaux pourraient permettre de trouver une solution. Pour conserver une solution distribuée, il est par exemple envisageable d'ajouter plus de signalisation (dans la limite du raisonnable, une vision globale est en effet idéale pour décider mais a du mal à passer à l'échelle) ou introduire un déséquilibre de manière régulière pour explorer de nouvelles voies.

Conclusion et Perspectives

Conclusion

Depuis sa création, Internet a vu son nombre d'utilisateurs augmenter très fortement. Cette popularité est aussi intimement liée aux évolutions de son utilisation. Originellement, Internet était principalement utilisé pour de l'échange de messages (mail) ou de fichiers. De nos jours, il est utilisé pour de très nombreuses applications. Si les échanges de messages sont toujours présents, leur volume n'est pas significatif en comparaison aux autres applications telles que le "streaming" de vidéos sur Youtube ou Netflix par exemple. Ce changement d'utilisation a poussé la communauté scientifique à imaginer et construire des réseaux orientés sur le contenu, d'abord en overlay sur IP avec le pair à pair (P2P) ou les Content Delivery Networks (CDN) puis maintenant directement au niveau réseau avec les Information Centric Networks (ICN). Ce changement de paradigme au niveau de la couche réseau n'est pas sans conséquence sur les couches supérieures. Le travail de cette thèse a donc porté sur l'amélioration de la Qualité d'Expérience des utilisateurs sur un réseau ICN. Dans le premier chapitre, nous avons étudié et comparé les différentes architectures ICN qui ont été proposées. Notre attention s'est portée sur le Named Data Networking (NDN) qui semble être le plus mature et abouti. Dans la suite de ce chapitre, nous avons donc étudié différentes solutions pour la gestion des ressources dans NDN.

Dans le deuxième chapitre, après avoir posé clairement le contexte dans NDN, nous avons défini la notion de flux. En effet, plusieurs utilisateurs peuvent demander le même contenu simultanément, un utilisateur peut récupérer ce contenu depuis plusieurs producteurs et depuis un même producteur mais via plusieurs chemins. Nous avons donc choisi de définir un flux dans NDN comme l'ensemble des messages échangés pour qu'un unique utilisateur puisse récupérer le contenu recherché. Ces messages peuvent donc être émis par plusieurs producteurs et emprunter plusieurs chemins. Dans la suite du chapitre, nous avons formulé la problématique de la thèse comme étant un problème de répartition équitable des flux dans un réseau de contenu utilisant les multi-chemins.

Nous avons décidé d'utiliser le débit comme critère de Qualité d'Expérience de nos utilisateurs et le critère d'équité Max-Min dans notre formulation, car nous estimons qu'il est important, sans a priori, de ne pas favoriser un flux plus qu'un autre. Finalement, nous avons formalisé cela, en nous inspirant du *Multi-Commodity Flow Problem*, avec N problèmes d'optimisation successifs (avec N le nombre de flux) où chaque problème permet de trouver le plus petit débit parmi les flux restants (dont le débit n'a pas encore été déterminé).

Dans les troisième et quatrième chapitres, nous proposons et raffinons notre solution Cooperative Congestion Control (CCC). CCC est une solution distribuée sur chaque nœud et qui a donc pour but de répartir équitablement les flux dans le réseau tout en cherchant à maximiser le débit de ces flux. Dans le troisième chapitre, nous définissons une première version de l'architecture de notre solution et ses trois principes : coopération, supervision et répartition. Les nœuds s'échangent contraintes et objectifs, supervisent leur file d'émission et décident grâce à ces informations les répartitions à faire pour satisfaire la stratégie d'allocation choisie (une équité Max-Min entre les flux dans notre cas). Notre cas d'étude dans ce chapitre se limite aux topologies arborescentes. Cette simplification limite les interactions entre les échanges montants et descendants (des consommateurs vers les producteurs et inversement) et nous a permis de définir la première version de notre architecture. Notre solution est volontairement modulable via une liberté du choix de stratégie d'allocation et d'implantation des algorithmes constituant notre architecture. Les différents algorithmes ont en effet certains critères généraux à respecter, mais nous ne définissons dans ce chapitre qu'une implantation délibérément simple. Le but est, dans un premier temps, d'évaluer l'architecture de CCC et de montrer son potentiel.

Dans le quatrième chapitre, nous levons la limitation des topologies arborescentes et nous intéressons à l'évolution de notre solution. Les trois principes restent les mêmes et la volonté d'avoir une modularité dans l'implantation aussi. L'architecture devient plus symétrique avec la supervision et la double répartition inter-chemin et inter-flux qui se fait aussi bien sur le Downstream que sur l'Upstream. Pour nous aider lors de l'implantation des différents algorithmes, nous définissons les notions de flux satisfaits et limités et de distributions satisfaisantes et équitables. Les résultats, obtenus en simulation, sont très satisfaisants cependant seule une équité locale sur chaque lien est atteinte (à l'instar des solutions de l'état de l'art).

Le cinquième et dernier chapitre s'intéresse au cas pratique des réseaux satellitaires. Au vu de la conception *receiver-driven* de NDN, l'approche des PEPs n'est pas envisageable et les solutions utilisant une fenêtre de congestion n'ont donc pas de bonnes performances lorsque le délai de bout en bout augmente trop. Les performances de CCC restent très

bonnes notamment grâce à l'utilisation du rythme des Interests comme métrique. Nous avons évalué CCC et les solutions de l'état de l'art en fonction du placement du lien satellite dans le réseau (en accès des consommateurs, en accès des producteurs, dans le réseau ou une connexion satellite en supplément d'une connexion terrestre) et notre solution s'est toujours très bien accommodée de cette nouvelle contrainte. CCC apporte donc une solution unique fonctionnant sur la plupart des réseaux qui composent Internet.

Les ICNs, et NDN en particulier, ont pour ambition de remplacer la couche IP. Si une phase de cohabitation semble envisageable, le plus probable est que NDN commence par être déployé en overlay sur IP. En effet, malgré sa maturité, le déploiement à grande échelle d'équipements gérant la couche NDN est très coûteux et peut ne pas convaincre les opérateurs réseaux à franchir le pas. NDN possède aussi ses défauts et la concurrence d'une technologie déjà mondialement déployée comme IP est rude. Même si le but était de se passer d'IP et de solutions en overlay comme les P2P ou les CDNs, NDN peut apporter une plus-value pour des services applicatifs tels que la vidéo à la demande ou le téléchargement de fichiers. Un premier banc d'essai est déployé par la communauté scientifique et permet notamment d'effectuer des expérimentations sur une trentaine de nœuds répartis un peu partout sur la planète [1]. Les points présentés dans la prochaine section pourraient offrir de nouveaux arguments à un déploiement à grande échelle une fois mis en œuvre.

Perspectives

Nos travaux présentent une solution prometteuse de répartition des flux dans un réseau de contenu. Comme nous l'avons vu, l'équité Max-Min globale n'est cependant pas garantie par celle-ci. Nous pensons qu'elle reste envisageable et fait donc partie des perspectives de cette thèse. Le problème actuel est un manque d'information des nœuds pour leur permettre de décider localement d'une augmentation ou réduction des débits d'un flux propice à cette équité globale. L'enjeu est donc de trouver une solution qui résolve ce problème tout en gardant l'aspect décentralisé de CCC.

De plus, lors de certains cas, nous nous sommes rendu compte que les mécanismes de basculement de flux pouvaient amener à des équilibres sous-optimaux. En effet, bien que l'équité Max-Min soit garantie sur chaque lien (localement), il peut être possible qu'un flux bascule une partie de son débit d'un de ses chemins vers un autre dans le but de libérer de la place pour un autre flux. Ce deuxième flux peut ainsi utiliser la place libérée, augmenter son débit total et améliorer la satisfaction de son utilisateur. Dans certaines

1. `https://named-data.net/ndn-testbed/`

configurations, notre solution n'est pas capable d'effectuer complètement ce basculement. À nouveau la vision locale du nœud est en cause. L'enjeu est à nouveau de trouver une solution à ce problème conservant la propriété décentralisée de CCC. Une piste pourrait être d'introduire une part d'aléatoire lorsque plusieurs opportunités a priori équivalentes sont disponibles. Cela permettrait de sortir de l'équilibre atteint pour en atteindre un autre potentiellement meilleur pour les autres flux.

Notre solution exploite deux caractéristiques majeures de NDN : l'utilisation des multi-chemins et la gestion au saut par saut. Nous avons mis de côté deux autres propriétés fortes des NDNs : le *multicast* et la mise en cache (ou *caching*). Pour le *multicast*, et avec notre définition de flux, nous considérons les différents consommateurs participants à une session *multicast* comme autant de flux distincts. Pour le *caching*, quand il est *off-path* (voir Section 1.2.4), le nœud est simplement considéré comme un producteur distinct et notre solution gère cela sans problème. Pour le *on-path caching* (ou opportuniste), les nœuds sur le chemin peuvent répondre aux Interests si la Data est disponible dans leur CS. Avec CCC, les contraintes exprimées représentent des réservations de ressource et le fait qu'un nœud sur le chemin puisse émettre la Data peut potentiellement perturber l'équilibre de ces réservations. Ces interactions sont donc un point intéressant à étudier.

Finalement, nous nous sommes beaucoup intéressés au point de vue des utilisateurs via leur QoE mais comment donner plus de possibilité aux opérateurs du réseau. D'un point de vue financier, ils pourraient souhaiter mettre en place des classes de service, limiter l'utilisation de certains liens du réseau (qui seraient particulièrement coûteux à utiliser) ou au contraire favoriser l'utilisation d'autres liens.

Bibliographie

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 66–73, July 2014.

[2] N. Rozhnova and S. Fdida, "An effective hop-by-hop Interest shaping mechanism for CCN communications," in *2012 Proceedings IEEE INFOCOM Workshops*, pp. 322–327, March 2012.

[3] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive Forwarding in Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 42, pp. 62–67, June 2012.

[4] N. Uniyal, D. Kutscher, J. Seedorf, J. Blendin, and D. Hausheer, "Adaptive ICN multipath forwarding for hybrid access," in *2017 International Conference on Networked Systems (NetSys)*, pp. 1–8, March 2017.

[5] A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "An Analysis of NDN Congestion Control Challenges," in *2019 2nd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 18–24, 2019.

[6] A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Cooperative Congestion Control in NDN," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[7] A. Thibaud, J. Fasson, F. Arnal, D. Pradas, E. Dubois, and E. Chaput, "QoE enhancements on satellite networks through the use of caches," *International Journal of Satellite Communications and Networking*, vol. 36, no. 6, pp. 553–565, 2018.

[8] A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Reactivity Enhancement of Cooperative Congestion Control for Satellite Networks," in *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pp. 135–141, 2020.

[9] A. Thibaud, J. Fasson, F. Arnal, D. Pradas, E. Dubois, and E. Chaput, "Évaluation de l'impact de caches pour de la vidéo adaptative par satellite," in *Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*, (Roscoff, France), May 2018.

[10] A. Thibaud, J. Fasson, F. Arnal, R. Sallantin, E. Dubois, and E. Chaput, "Cooperative Congestion Control dans NDN," in *Rencontres Francophones sur la Conception*

de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication, (Lyon, France), Sept. 2020.

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[12] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, *et al.*, "Named Data Networking (NDN) Project," *Technical Reports NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[13] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-oriented (and Beyond) Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, Aug. 2007.

[14] M. Ain, D. Trossen, P. Nikander, S. Tarkoma, K. Visala, K. Rimey, T. Burbridge, J. Rajahalme, J. Tuononen, P. Jokela, J. Kjällman, J. Ylitalo, J. Riihijärvi, B. Gajic, G. Xylomenos, P. Savolainen, and D. Lagutin, *Architecture Definition, Components Descriptions and Requirements*. FP7, feb 2009. Deliverable D2.3.

[15] D. Trossen, G. Parisis, B. Gajic, J. Riihijarvi, P. Flegkas, P. Sarolahti, P. Jokela, X. Vasilakos, C. Tsilopoulos, S. Arianfar, and M. Reed, *Architecture Definition, Components Descriptions and Requirements*. FP7, oct 2011. Deliverable D2.3.

[16] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of Information (NetInf) - An Information-Centric Networking Architecture," *Comput. Commun.*, vol. 36, p. 721–735, Apr. 2013.

[17] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri, "MobilityFirst Future Internet Architecture Project," in *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, (New York, NY, USA), pp. 1–3, ACM, 2011.

[18] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst : A Robust and Trustworthy Mobility-centric Architecture for the Future Internet," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 16, pp. 2–13, Dec. 2012.

[19] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste, "XIA : An Architecture for an Evolvable and Trustworthy Internet," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, (New York, NY, USA), pp. 2 :1–2 :6, ACM, 2011.

[20] H. Du, D. Trossen, C. Theodorou, and T. Karkkainen, *Final Component and Interface Specifications*. Horizon 2020, Jul 2017. Deliverable D2.3.

[21] D. Trossen, M. J. Reed, J. Riihijärvi, M. Georgiades, N. Fotiou, and G. Xylomenos, "IP over ICN - The better IP ?," in *2015 European Conference on Networks and Communications (EuCNC)*, pp. 413–417, June 2015.

[22] M. Guta, C. Ververidis, A. Drougas, I. Andrikopoulos, V. Siris, G. Polyzos, F. Arnal, and C. Baudoin, *Satellite-Terrestrial Integration Scenarios for Future Information-Centric Networks*.

[23] S. Farrell, C. Dannewitz, B. Ohlman, D. Kutscher, P. Hallam-Baker, and A. Keränen, "Naming Things with Hashes." RFC 6920, Apr. 2013.

[24] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN : Line Speed Publish/Subscribe Inter-networking," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 195–206, Aug. 2009.

[25] H. Liu, X. De Foy, and D. Zhang, "A Multi-level DHT Routing Framework with Aggregation," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, (New York, NY, USA), pp. 43–48, ACM, 2012.

[26] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "MDHT : A Hierarchical Name Resolution Service for Information-centric Networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, (New York, NY, USA), pp. 7–12, ACM, 2011.

[27] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR : Named-data Link State Routing Protocol," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, (New York, NY, USA), pp. 15–20, ACM, 2013.

[28] L. Wang, V. Lehman, A. K. M. Mahmudul Hoque, B. Zhang, Y. Yu, and L. Zhang, "A Secure Link State Routing Protocol for NDN," *IEEE Access*, vol. 6, pp. 10470–10482, 2018.

[29] M. Zhang, H. Luo, and H. Zhang, "A Survey of Caching Mechanisms in Information-Centric Networking," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 1473–1499, thirdquarter 2015.

[30] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in information centric networking," *Journal of Network and Computer Applications*, vol. 56, no. Supplement C, pp. 48 – 59, 2015.

[31] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking : A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013. Information Centric Networking.

[32] "ICN Research Group, IRTF." `https://irtf.org/icnrg`.

[33] F. Baker, "Requirements for IP Version 4 Routers." RFC 1812, June 1995.

[34] M. Mosko, I. Solis, and C. A. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format." RFC 8609, July 2019.

[35] P. Gusev and J. Burke, "NDN-RTC : Real-Time Videoconferencing over Named Data Networking," in *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, (New York, NY, USA), pp. 117–126, ACM, 2015.

[36] Z. Zhu and A. Afanasyev, "Let's ChronoSync : Decentralized dataset state synchronization in Named Data Networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, Oct 2013.

[37] I. Moiseenko, M. Stapp, and D. Oran, "Communication Patterns for Web Interaction in Named Data Networking," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, (New York, NY, USA), pp. 87–96, ACM, 2014.

[38] H. M. A. Islam, A. Lukyanenko, S. Tarkoma, and A. Ylä-Jääski, "Towards Disruption Tolerant ICN," *CoRR*, vol. abs/1510.04436, 2015.

[39] G. Tyson, J. Bigham, and E. Bodanese, "Towards an information-centric delay-tolerant network," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 387–392, April 2013.

[40] C.-A. Sarros, A. Lertsinsrubtavee, C. Molina-Jimenez, K. Prasopoulos, S. Diamantopoulos, D. Vardalis, and A. Sathiaseelan, "ICN-based Edge Service Deployment in Challenged Networks," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, (New York, NY, USA), pp. 210–211, ACM, 2017.

[41] "DASH Industry Forum." `http://dashif.org/`.

[42] Y. Liu, J. Geurts, J. C. Point, S. Lederer, B. Rainer, C. Müller, C. Timmerer, and H. Hellwagner, "Dynamic adaptive streaming over CCN : A caching and overhead analysis," in *2013 IEEE International Conference on Communications (ICC)*, pp. 3629–3633, June 2013.

[43] B. Niven-Jenkins, F. L. Faucheur, and D. N. N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement." RFC 6707, Sept. 2012.

[44] G. Ma, Z. Chen, J. Cao, Z. Guo, Y. Jiang, and X. Guo, "A tentative comparison on CDN and NDN," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2893–2898, Oct 2014.

[45] S. Mastorakis, A. Afanasyev, Y. Yu, and L. Zhang, "nTorrent : Peer-to-Peer File Sharing in Named Data Networking," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–10, July 2017.

[46] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP : Design and evaluation of an Interest control protocol for content-centric networking," in *2012 Proceedings IEEE INFOCOM Workshops*, pp. 304–309, March 2012.

[47] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer Issues in Information Centric Networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, (New York, NY, USA), pp. 19–24, ACM, 2012.

[48] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in Information-Centric Networks," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, Oct 2013.

[49] Y. Ye, B. Lee, R. Flynn, N. Murray, G. Fang, J. Cao, and Y. Qiao, "PTP : Path-specified Transport Protocol for Concurrent Multipath Transmission in Named Data Networks," 2018.

[50] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint Hop-by-hop and Receiver-driven Interest Control Protocol for Content-centric Networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, (New York, NY, USA), pp. 37–42, ACM, 2012.

[51] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An Improved Hop-by-hop Interest Shaper for Congestion Control in Named Data Networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, (New York, NY, USA), pp. 55–60, ACM, 2013.

[52] N. Rozhnova and S. Fdida, "An extended Hop-by-hop interest shaping mechanism for content-centric networking," in *2014 IEEE Global Communications Conference*, pp. 1–7, Dec 2014.

[53] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-delay Product Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 89–102, Aug. 2002.

[54] A. Detti, C. Pisa, and N. B. Melazzi, "Modeling multipath forwarding strategies in Information Centric Networks," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 324–329, April 2015.

[55] Y. Ren, Z. Li, J. Li, Y. Qin, H. Wu, and X. Zhou, "DMF : A Dynamic Multi-Path Forwarding Strategy for Information Centric Networks," in *2019 IEEE 21st International Conference on High Performance Computing and Communications ; IEEE 17th International Conference on Smart City ; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 2495–2501, 2019.

[56] A. Kerrouche, M. R. Senouci, and A. Mellouk, "QoS-FS : A new forwarding strategy with QoS for routing in Named Data Networking," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2016.

[57] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A Practical Congestion Control Scheme for Named Data Networking," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, (New York, NY, USA), pp. 21–30, ACM, 2016.

[58] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management." RFC 8289, Jan. 2018.

[59] H. B. Abraham and P. Crowley, "Controlling Strategy Retransmissions in Named Data Networking," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 70–81, May 2017.

[60] A. Z. Khan, S. Baqai, and F. R. Dogar, "QoS aware path selection in content centric networks," in *2012 IEEE International Conference on Communications (ICC)*, pp. 2645–2649, June 2012.

[61] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. V. Krioukov, and L. Wang, "An Experimental Investigation of Hyperbolic Routing with a Smart Forwarding Plane in NDN," *CoRR*, vol. abs/1611.00403, 2016.

[62] K. Lei, J. Yuan, and J. Wang, "MDPF : An NDN Probabilistic Forwarding Strategy Based on Maximizing Deviation Method," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Dec 2015.

[63] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2 : An updated NDN simulator for NS-3," Technical Report NDN-0028, Revision 2, NDN, November 2016.

[64] M. Sargent, J. Chu, D. V. Paxson, and M. Allman, "Computing TCP's Retransmission Timer." RFC 6298, June 2011.

[65] J.-Y. Boudec, "Rate adaptation, Congestion Control and Fairness : A Tutorial," 2000.

[66] T. C. Hu, "Multi-Commodity Network Flows," *Operations Research*, vol. 11, no. 3, pp. 344–360, 1963.

[67] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations." RFC 3135, June 2001.

[68] R. Sallantin, "Optimisation de bout-en-bout du démarrage des connexions TCP." Septembre 2014.

Résumé

Avec l'apparition de service de vidéo à la demande, l'utilisation du streaming a explosé. Le volume de données engendré oblige les opérateurs réseaux à définir de nouvelles solutions. Dans cette thèse, nous nous intéressons à une nouvelle approche, l'Information Centric Networking, qui cherche à déconstruire le modèle d'IP en se focalisant sur le contenu recherché. Pour que les ICNs soit une réelle opportunité pour Internet, il lui faut offrir une meilleure Qualité d'Expérience aux utilisateurs tout en utilisant efficacement les capacités des réseaux. Nous avons conçu Cooperative Congestion Control, une solution distribuée ayant pour but de résoudre ce problème, en répartition les flux équitablement dans les réseaux de contenu. Nous avons éprouvé notre solution sur des topologies avec des liens satellites. Grâce à l'émission des requêtes régulée par notre solution, nous montrons que ces hauts délais n'ont que très peu d'impacts sur les performances de CCC.

Mots clés : ICN, NDN, QoE, Contrôle de Congestion, Équité, Satellite

Abstract

With the emergence of video-on-demand services, the use of streaming has exploded. The volume of data generated forces network operators to define new solutions. In this thesis, we focus on a new approach, Information Centric Networking, which seeks to deconstruct the IP model by focusing on the requested content. In order for ICNs to be a real opportunity for the Internet, it must offer a better Quality of Experience to users while efficiently using network capacities. We designed Cooperative Congestion Control, a distributed solution to solve this problem, by distributing flows fairly across content networks. We tested our solution on topologies with satellite links. Thanks to the emission of requests regulated by our solution, we show that these high delays have very little impact on the performance of CCC.

Keywords : ICN, NDN, QoE, Congestion Control, Fairness, Satellite