Congestion control for Content-Centric Networking

Natalya Rozhnova

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et électronique (Paris)

Présentée par

Natalya ROZHNOVA

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Congestion control for Content-Centric Networking

soutenue le 20 mai 2015

devant le jury composé de :

M. Fabio MARTIGNON	Rapporteur
M. Walid DABBOUS	Rapporteur
M. David ORAN	Examinateur
Mrs. Lixia ZHANG	Examinateur
M. Giovanni PAU	Examinateur
M. Serge FDIDA	Directeur de thèse

"*There is nothing simpler than carrying out the impossible. You just have to imagine what you must do, and turn your mind off completely. When you come to your senses, everything is already behind you...*"

- Max Frei, "The Stranger"

"Нет ничего проще, чем совершить невозможное. Стоит только представить себе, чем сейчас придётся заниматься, и сознание тут же отключается. А когда приходишь в себя — всё уже позади..."

- Макс Фрай, "Чужак"

Biography

Natalya Rozhnova was born in Novosibirsk, Russia. She received her Engineer degree in Telecommunication Network and Switching Systems from Siberian State University of Telecommunications and Information sciences, Novosibirsk, Russia, in 2010. In 2011, she received her M.S. degree in Networking from University Pierre & Marie Curie (UPMC) where she pursued her Ph.D. degree in Computer Science. She had the pleasure of being advised by Prof. Serge Fdida. Her research interests are related to the future Internet architectures and protocols, with an emphasis on traffic engineering and performance analysis for information-centric networking.

Acknowledgements

I am using this opportunity to express my special appreciation and thanks to my advisor Professor Serge Fdida, he has been a tremendous mentor for me. Without his guidance and encouragement I would never know what is research and how the things work. I thank Serge for being such a perfect advisor and providing continous support in every step of my dissertation. I would also like to thank him for being patient with my questions and for giving me the invaluable lessons that I will never forget.

I would also like to thank Lixia Zhang, Giovanni Pau, Dave Oran, Fabio Martignon and Walid Dabbous for having accepted to be part of my PhD defense committee. Thanks for your useful comments and suggestions on my PhD dissertation.

During my doctoral research, I had a great fortune to have worked with different researchers, engineers and great colleagues. I extremely thank Dave Oran, Ashok Narayanan, Won So, Mark Stapp for their willingness to share their knowledge and experience with me, and their support and guidance during my sojourn at "Cisco Systems". Special thanks to Dave Oran who welcomed me in 2012 for a research internship at "Cisco Systems". I thank Dave for encouraging my research and for kindly and enthusiastically participating in my growth as a research scientist. His advices and contribution have been priceless. "Cisco Systems"'s NDN team will always stay in my heart.

I would also like to thank Yaogong Wang and Edmund Yeh for our valuable discussions and meetings during my sojourn in Boston.

Many thanks to Naceur Malouch, Bruno Baynat for being available for many useful discussions and providing valuable comments on my work. Thanks for being patient with my questions on congestion control and queuing theory, and for helping me to find a right way.

I extremely thank my friends that I have met during my years at LIP6 and Université Pierre et Marie Curie (UPMC): Marc-Olivier Buob, Jordan Augé, Sergey Kirgizov from Complex Networks team, Valetin Averchenko from Laboratoire Kastler Brossel. Thanks for your infinite support and for being anytime available for discussions. Your contribution in my research and personality was awesome. Special thanks to Marc-Olivier Buob for continous support, encouragement and for his invaluable help.

I would also like to specially thank Valentin Averchenko and Dominik Maxein from Laboratoire Kastler Brossel with whom we formed a music band at UPMC, enjoyed the music and our life behind research. Thanks to everybody who supported us during our concerts at Cité Internationale Universitaire de Paris.

Many thanks to Andrea Araldo for presenting my paper at GLOBECOM 2014.

And, of course, without my friends in France as well as in Germany and USA my life would be not be so filled many wonderful moments during my thesis, thanks to Marina and Roman Johansen, Ksenia Sryvkova, Dina Bessede, Inga Grin, Ieva Sliesoraityte, Maria

Nazarova. Thanks also to my friends in Russia for their support and for being with me all this time : Regina Yakushina, Natalya Shaidouk, Olga Sabitova, Rouslan Tinaev, Alexey Raspopine, Marina Samsonova. Thanks the horde !

I would also like to thank Marguerite Sos for her very professionnal and perfect work, for her great attention to every member of the team and, in particular, for her willingness to help. Also, thanks to Ciro Scognamiglio for technical support. It has been a pleasure for me to be a member of NPA team as well as LINCS where I met a lot of awesome people. It's difficult to list everybody here, so I'm just saying : thank you all guys!

Words cannot express how grateful I am to my mother, father and my granny for all of the sacrifices that they have made for me. I dedicate this dissertation to them with my love and greatest respect.

Résumé

Les principes d'ingénierie et de l'architecture actuelle de l'Internet ont été créés au cours des années 1960-1970. Ceux-ci cherchaient surtout à offrir un partage des ressources efficace permettant l'utilisation d'equipement rare et coûteux à distance. Ils reposent sur la mise en relation de deux machines : celle qui cherche à utiliser des ressources et celle qui les possède.

Depuis, l'equipement est devenu beaucoup moins coûteux et ce besoin a perdu de son importance : les utilisateurs s'intéressent désormais davantage au contenu en lui-même qu'à sa localisation. Afin de mieux répondre à ce nouveau besoin, il est nécessaire de concevoir de nouvelles architectures de réseaux adaptés aux services proposés aux utilisateurs (vidéo à la demande, web, ...). Celles-ci doivent répondre à la question "quoi" et non plus "où". La thématique des réseaux orientés contenu est devenue centrale dans les domaines actuels de recherche. La communauté de recherche s'intéresse tout particulièrement à l'architecture de CCN (Content-Centric Networking) proposée par Van Jacobson et al. dans "Networking Named Content" en 2009.

Or en général, les ressources d'un réseau informatique sont partagées par un très grand nombre d'utilisateurs, le réseau peut devenir congestionné voire saturé. C'est pourquoi le contrôle de congestion est un élément critique pour garantir son bon fonctionnement. Encore récemment, les problématiques d'ingénierie de trafic et de qualité de service n'étaient pas explorées dans le cadre des réseaux CCN.

L'objectif de cette thèse consiste d'abord à évaluer l'impact de la congestion dans ce type de réseaux, puis à concevoir un mécanisme de contrôle de congestion et à en évaluer l'efficacité.

Au cours de nos travaux, nous avons créé une solution efficace de contôle de congestion dans les réseaux CCN. Le problème de contrôle de congestion a une grande importance et n'était pas abordé jusqu'à présent. Nous avons commencé par concevoir le mécanisme HoBHIS (hop-by-hop Interest shaping) qui s'appuie sur la notion d'équilibrage de flot définie dans CCN. Ce mécanisme est déployé dans chaque routeur CCN. Il consiste à surveiller les conversations actives partageant le même tampon de transmission afin de contrôler dynamiquement le taux d'envoi de demands de contenu correspondants. Ce contrôle cherche à garantir qu'au niveau du goulot d'étranglement, la taille de la file d'attente des paquets de contenu correspondants tende vers un seuil préconfiguré. Nous avons ensuite étendu le design de HoBHIS afin de répondre à d'autres problématiques spécifiques à CCN, notamment un mécanisme permettant de contrôler le débit d'envoi d'un client afin d'éviter une congestion des files d'attente des noeuds CCN impliqués dans la communication.

Nous avons aussi prouvé mathématiquement l'efficacité de nos propositions, puis évalué les performances de ces mécanismes à l'aide du simulateur ns3-based Named-Data Networking (ndnSIM). Notre implémentation est open source et disponible publiquement sur

Internet.

Les résultats de notre travail ont été publiés dans des conférences internationales renommées. Par ailleurs, nous avons collaboré avec *Cisco Systems (Boston, USA)* qui s'est intéressé à nos travaux. Les résultats issus de cette collaboration ont été publiés dans un article international qui a reçu un "Best Paper Award" et sont également presentés dans cette thèse.

Abstract

The current Internet architecture was designed over 40-50 years ago and its primary goal was to share expensive and scarce physical resources. People cared mostly about *where* the resources are and, thus, the final communication model allowed to solve the problem of resource sharing grew to a conversation between two machines: one wishing to use the resource and one providing access to it.

The computers and devices have become cheap and ubiquitous commodities since. Thus, the primary need of resource sharing has lost its importance. Now, people care about *what* content network contains and not *where* it is located. In order to address this new need, it is important to design a new network architecture moving away from packets and getting closer to the service delivered to the user. The novel communication model should answer the question *what* and no longer *where*. Thus, Information-Centric Networking (ICN) is becoming an important stream of research. In particular, the architecture of Content-Centric Networking (CCN) proposed by Van Jacobson and al. in [34] appears as the most popular one. It aims at developing a new architecture better suited to distribute content at scale. It is based on a content routing paradigm and uses extensive caching capabilities.

It is well known that the network resources are shared between a large number of users. It may potentially create a risk for buffer overflow and performance degradation. That is why, congestion control is an important component to guarantee network performance. Congestion control schemes have been widely studied in the past but only recently in the context of CCN.

The objective of the research work presented in this dissertation is to explore the congestion control risk of such an architecture as CCN, identify the bottlenecks and propose strategies to circumvent them. We are interested in developing an effective mechanism for congestion control in order to guarantee good network performance.

We have developed a first comprehensive solution for congestion control in CCN networks. This problem is of utmost importance and has not been globally addressed in the past. We have designed our original hop-by-hop Interest shaping mechanism (HoBHIS) that nicely exploits the flow balance enforced in CCN between Interest and Chunk packets introduced in CCN. It mostly consists in monitoring active conversations sharing the transmission buffer of a CCN node face in order to dynamically adjust their Interest sending rate and enforce the Chunk queue length to converge to a defined objective. This mechanism is implemented in each CCN node. Then, we extended the design of HoBHIS in order to address several important concerns that might occur in CCN. We proposed a Tolerance mechanism designed to control the Clients sending rate as well as prevent the loss of Interest packets. A thorough evaluation was conducted using different simulation scenarios. We observed that the results fully satisfy the design objectives and we can conclude that HoBHIS is an efficient and operational solution to the problem of congestion

control in CCN. The performance evaluation of our solution is done using the implementation of our mechanisms in ns_3-based Named-Data Networking Simulator (ndnSIM) that implements Named Data Networking communication model. Our implementation is publicly available and can easily be found on the internet.

Our progress and results have been published in international conferences. The work has generated interest from the networking community. Particularly, "Cisco Systems" invited us to join their project on developing Named-Data Networking (NDN) traffic control mechanism. As one of the important results of this collaboration is a research article that got a SIGCOMM ICN workshop'13 “Best paper award”. This paper is also presented as a part of the dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing importance of content has triggered a tremendous interest from the networking research community, moving away from packets and getting closer to the service delivered to the user. In particular, the architecture of Content-Centric Networking (CCN) proposed by Van Jacobson and al. in [34] appears as the most popular one. At this stage, CCN defines design principles and has developed a CCNx Open Source Platform. However, many functionalities are still at an early stage, or have been considered in very simple contexts.

CCN decouples the sender from the receiver in a mode that is similar to the Publish and Subscribe service model. Content names are used instead of network addresses to convey the information to the interested parties. The content might be located anywhere in the network thanks to extensive caching capabilities. Therefore, the data is not necessarily associated with the content publisher as the content can be delivered by any cache in the network. In CCN, the request from the content consumer is called *Interest* and the part of the associated content is called a *Chunk* or *Data*. We will name a stream of Interest/Chunk pairs a *Conversation*.

Traffic engineering has been lightly addressed in CCN. Congestion might arise in such networks as chunks can saturate the transmission buffer of a network face. It is thus necessary to regulate the stream of chunks, and therefore, the associated stream of interest in order to avoid congestion and performance degradation. This problem has not yet been formalized and studied in CCN.

1.1 Work description

In this research work, we propose a congestion control mechanism for CCN, based on hop-by-hop Interest shaping. It relies on the assumption that any CCN router can control the future rate of data-chunks it will receive by shaping the rate of the Interest it is currently sending towards content providers, as one Interest retrieves at most one Data packet. We monitor the level of Chunks stored in the router transmission buffer in order to dynamically adjust the associated Interests rate that have generated the Chunks in that buffer. We first introduced in [52], a hop-by-hop Interest shaping mechanism based and exploiting the "one Interest - one Chunk" rule enforced in CCN. In this scheme, each CCN router controls the future rate of data-chunks by shaping the rate of corresponding Interests it is pushing upstream. CCN routers dynamically adjust their Interest sending rate by monitoring the level of Chunks stored in their transmission buffer. The rate

resulting from this operation is called the shaping rate. An important design function is the control one used to derive an optimum shaping rate maintaining the queue length around a target value named an objective. Various important parameters are calibrated in order to efficiently operate the algorithm as a function of the network characteristics. We demonstrate the convergence properties of this Hop-by-hop Interest Shaping mechanism (HoBHIS) and provide a performance analysis based on various scenarios.

As a consequence of the CCN design principles, we have to regulate the stream of chunks as well as the stream of interests in order to avoid congestion and to improve network performance. We then complete in [53] the analysis and design of HoBHIS by extending its operation when facing more complex situations. We study its behavior in a multicast scenario and test its effectiveness when Interest aggregation is used. We also demonstrate the fair resource sharing among competing conversations. Another important concern is related to the ability offered by CCN to a user to send interests without any limiting rate factor, creating a risk for buffer overflow and performance degradation. We introduce a new rate-based mechanism aiming to control the interest sending rate of a content receiver (a client or source of interest). The scheme introduces an exchange of control information between the CCN nodes and the client. The CCN nodes periodically send control packets with an explicit rate specifying the maximum sending rate for each Client on the path used by this conversation. In order to derive an optimum explicit rate, CCN routers use a control function based on the shaping rate computed by HoBHIS.

1.2 Dissertation Roadmap

This dissertation presents a step-by-step design of a congestion control mechanism for Content-Centric Networking. The remainder of the dissertation is organized as follows:

Chapter 2 provides a brief description of the most prominent Information-Centric Networking (ICN) architectures in the litterature. We survey different ICN architectures proposed by researchers all around the world.

We then provide a short introduction to CCN underlining the most meaningful properties for this work. We then present an original model of a CCN node and propose some design modifications.

Further in this chapter we present work related to congestion control in CCN. We discuss the different work from the past that has been useful for our contribution. We also analyse different types of congestion control and provide an analysis of various solutions proposed for congestion control in CCN.

Chapter 3 discusses congestion in CCN, introduces traffic control issues and describes the hop-by-hop Interest shaping mechanism (HoBHIS). We then provide a performance evaluation of our hop-by-hop shaping mechanism using various scenarios and our ns_3-based ndnSIM simulation environment.

Chapter 4 extends the analysis and design of HoBHIS and proposes a new Tolerance mechanism to control the interest rate of a client. The performance evaluation of a complete solution concludes this chapter.

Chapter 5 challenges our schemes in more complex scenarios and proposes improvements in order to develop a unified control that will take into account all possible situations.

Chapter 6 describes our joint work with “Cisco Systems” and presents a proactive

NDN congestion control scheme for two-way traffic.

Finaly, **Chapter 7** concludes the dissertation and discusses future work.

Some Appendixes are also available at the end of the dissertation: Appendix A presents implementations of our solutions in ndnSIM and provides some overview of the source code and its structure. Appendix B describes my activity during the years of my PhD work.

Chapter 2

Information-Centric Networking

2.1 ICN

The current Internet achitecture was developped in 1960s and designed in host-centric fashion due to needs of sharing physical resources. This design makes all communications to be based on point-to-point conversation between named hosts. Today, people are more interested in content itself and no more in its location. But to get a desired content it should be mapped to the machine hosting it. Now if any user wants to get a content, it has to visit a specific server. In reality, multiple users that want to obtain the same content may create a huge load on that server. Moreover, the end-to-end connections are created for each content demand and introduce a lot of redundant traffic in the network.

The Information-Centric Networking tries to solve the problems due to host-centric architecture and makes the content on the first plane. To make it possible, the content has to have globally unique and location-independent names. Now, when the clients want to obtain any content they should specify its name and network will retrieves the content for them from anywhere. This approach completely change the today's Internet.

The increasing importance of content has triggered a tremendous interest from the networking research community. Over the last years, many different information-centric networking (ICN) architectures have been introduced to cope with the evolution of the internet usage towards massive content distribution.

The information-centric approach to the future network architecture are actually being explored by a number of research projects in Europe and in the US.

In this section, we provide a brief description of some of such ICN architectures. More information and detailed surveys of ICN may be found in [10] and [23]. Also, more general and broad survey on future internet architectures is provided in [47].

2.1.1 TRIAD

The TRIAD project [29] was the first to explore in 2001 the benefits of diverging from the classic Internet architecture. It proposes a new Internet architecture identifying the contents by names rather than addresses. TRIAD tries to name content with user-friendly and location-independent names. The name-to-address resolution is performed by the content routers instead of DNS servers. To do so, it uses URLs as the names mapping the DNS portion of the content URL to the nearby replica of the content. Each content router maintains a set of name-to-next-hop mappings. The content router forwards the

requests to the next content router until a copy of the data is found. Its location then returned to the client. Routing in TRIAD is still done at the granularity of server names rather than full URL of individual content.

2.1.2 DONA

The Data-Oriented Network architecture presented in [35] proposes replacing DNS names with flat, self-certifying names, and replacing DNS name resolution with a name-based anycast primitive above the IP layer. This architecture is built on name-based routing as is advocated in TRIAD and extends the ICN idea to content granularity instead of server names.

Name resolution in DONA is done by resolution handlers (RH) instead of DNS servers. Each domain or administrative entity has one logical RH. The RHs have a hierarchical structure according to the provider/customer/peer relationships between the domains or entities.

Content in DONA must first be published, or registered, to enable its retrieval. To do so, the content providers should send the $REGISTER$ messages to its local RH. This messages set up the registration table on each RH that provides next-hop information and the distance to the copy. Once a given content is registered, requests, or $FIND$ messages, can be effectively routed to it. Once the content is located, packets are exchanged with the requester using standard IP routing.

2.1.3 PSIRP/PURSUIT

Publish-Subscribe Internet Routing Paradigm (PSIRP) is an EU FP7 funded project [4] later succeeded by the Publish-Subscribe Internet Technology (PURSUIT) that explores and expands PSIRP's vision.

It proposes a new architecture for the future Internet, based on the publish-subscribe primitives. PSIRP uses a rendezvous system to match publications and subscriptions.

Each peace of data is uniquely identified by a pair of identifiers: a rendezvous identifier (RId) and a scope identifier (SId). The RId is an information item has to be unique within a scope, whereas the SId denotes the scope in which an information item belongs. The content should first be published to enable access to it. In order to publish the content, the publisher has to know the SId as well as to create a RId for the publication that is then forwarded to the rendezvous node of the SId rendezvous network.

A consumer learns the RId and the SId of a desired piece of information and issues a subscription message towards the appropriate rendezvous point. Once this message is received by the rendezvous point, a forwarding path is created between the publisher and the subscriber. Each active publication has a forwarding identifier (FId) that denotes the forwarding path to follow.

2.1.4 4WARD NetInf

Network of Information (NetInf [9]) is an FP7 4WARD project [6] that has later evolved further under FP7 SAIL project [5]. It focuses on higher level issues of information and represents the content in the form of so-called $Information\ Objects$ (IOs). An IO may refer to a certain content (e.g. a song) without specifying the concrete representation (e.g.

the encoding). It enables users to find content independently of certain characteristics. In NetInf IOs are registered into the network with a Name Resolution Service. NetInf adopts a hybrid approach that supports both name resolution and name-based routing.

2.1.5 Content-based Networking

Content-based Networking [21] introduces a unique naming scheme where each content is identified by a set of attribute/value pairs instead of hierarchical names or flat labels. A user request is a selection predicate that is a logical disjunction of conjunctions of elementary constraints over the values of individual attributes. The matching of publication and subscription is a search process to find contents that match the selection predicates declared by the consumers. As CBN differentiates itself from all above-mentioned ICN architectures by its different semantics, routing and forwarding require totally different approaches.

2.1.6 The Network is a database

The network paradigm presented in [27] aims at offering a language inspired from database background over the network. Such a language would allow users and services to characterize accurately their interests and retrieve the corresponding data. The data is described using multiple attributes and modeled as a table (e.g. a movie could be characterized by a title, a year, etc. leading to the "movie" table). The proposed architecture extends Publish/Subscribe service model. It acts as a mediator between data providers (which both offer the data to the network and describe the corresponding "tables") and the data consumers which express their interest by issuing SQL-like queries over this collection of tables. The proposal is still in its early stage and in the same way as CBN, this architecture requires totally different approaches.

2.1.7 CCN/NDN

Content-Centric Networking (CCN) proposed by Van Jacobson and al. in [34] appears as the most popular one. This proposal became later one of the Future Internet Architecture (FIA) projects under the new name of Named Data Networking (NDN [66]).

This architecture differs from the traditional host-based communication principle in many ways. One important change is that network addresses are replaced by content names to distribute information in CCN networks. Moreover, extensive in-network caching capabilities are introduced to benefit from the observation of the traffic flowing in the network. As a consequence, data packets can be delivered by any CCN router in addition to the source, according to various caching algorithms. That differs CCN from other ICN architectures where content must first be published to enable retrieval and, thus, data cannot be generated dynamically in response to queries.

We will discuss this architecture in details later in this dissertation because our congestion control work is based on the architectural features of this ICN approach. In this dissertation, we use CCN and NDN as the synonyms.

2.1.8 Why CCN?

As we have seen above, the addressing in current internet architecture does not allow to meet the demands of today's applications and mobile environments. The different solutions have been proposed. Some of them implement the functionality above the existing Internet architecture, others propose a completely different way by replacing the whole network architecture by a new one. All of these proposals are trying to move from host-centric to content-oriented networking. But their main challenges mostly consist in the efficient routing of the unstructured "flat" names and mapping the names to the opaque labels. We compare now the different architectures presented earlier in this Chapter with CCN in order to explain why we have finally chosen this proposal.

DONA is implemented above the IP level. As we have seen, unlike CCN, content in DONA can not be generated dynamically in response to requests because it must first be published, or registered. Once the content is located, packets are exchanged using standard IP routing. If the content is moving, new registration should be propagated through the network to enable its retrieval. In CCN content may be located everywhere in the network thanks to the extensive caching capabilities and the nodes forward the requests to the different possible content locations.

Another proposal, PSIRP, suffers from its use of unstructured indentifiers. In CCN, in contrast, the names have an hierarchical structure that facilitates locating and sharing data.

4WARD NetInf focuses on higher-level issues of information and represents the content in the form of Information Objects. The goal of this architecture is similar to CCN and consists in creating a new "network of information" paradigm where the information objects have their own identity. Unlike CCN, content retrieval in NetInf consists in two-step approach where the first step is to resolve content name to its location and then route to this location. CCN routes packets directly according the content name.

CCN has generated a great interest in the research community and numerous research projects has been funded all over the world. The first project that has introduced CCN was CCNx ([1]). Later, NDNx ([2]) has been forked from CCNx and continued to make research in this direction. In Europe, we have been a part of the ANR CONNECT project started in 2010 and focused on CCN architecture, routing, caching and traffic control. It gathered the following partners: Alcatel Lucent Bell Labs, INRIA, Institut Telecom PARISTECH, France Telecom Orange, Université Pierre et Marie Curie (UPMC) LIP6. Also, the big corporations like "Cisco Systems", "Alcatel Lucent", "Huawei" have started research on CCN/NDN and are very motivated to continue in this direction.

In addition to different proposed architectures, CCN has many evaluation tools. The first implementation [1], has been developed by CCNx project. Later, the NDN project has created a fork NDNx, [2], of the (L)GPL'd CCNx codebase to add new features, fixes etc. that may differ from CCNx timeline.

Later, a ccnSim, [51], has been developed to evaluate the caching performance of CCN. Hovewer, some features are implemented in the simplest way that does not allow the evaluation of different forwarding strategies, traffic control schemes, etc.

Another Content-Centric Networking Packet-Level Simulator (CCNPLsim), [43], has been developed by OrangeLabs. The drawback of this solution is that it uses a custom discrete-event simulator that is unfamiliar to most researchers.

A different solution is proposed in [8] and presents the Direct Code Execution (DCE)

for CCNx implementation inside the NS-3 simulator. The main advantage of this proposal is that it is directly based on unmodified CCNx code that provides more realism. Hovewer, the implementation is complex and difficult to modify to explore different design approaches.

Another simulator that implements Named Data Networking (NDN) communication model is ndnSIM, [3] that represents an ns_3 module. This simulator is optimized for simulation purposes and easy to use, modify and extend. Even if this implementation is relatively recent, it becomes very popular in the research community. We use this simulator for implementing our congestion control schemes presented in this dissertation.

2.2 Content-Centric Networking

The existing ICN architectures that we briefly discussed in section 2.1 have different ideas but, hovewer, most of them share some common features such as name-based routing, content caching, content-based security. In this chapter we provide a short introduction to CCN in order to highlight the most important architectural features useful in the context of our contribution. We discuss then the common features of ICN architectures based on the example of CCN.

2.2.1 CCN architecture

Any content requested by a Client (or a source of Interests) can be divided into a number of data packets. In order to request a given piece of content, a Client will have to send an interest packet. This interest packet will be forwarded to a location where the content is stored. It could be the server or a CCN router where the content has been cached. As a consequence, in order to fetch a given content, the Client will have to submit as many Interest packets as the number of chunks that exist for that particular content. We observe that an Interest triggers the reception of a single chunk transmitted on the reverse path used by the one followed by the associated Interest from the Client to the location where the chunk is retrieved. The "One Interest - One Chunk" rule is an interesting property that enforces a flow balance in a CCN network.

Each CCN router supports three different types of storage. The Interests are forwarded to the data source thanks to the Forwarding Information Base (FIB). In addition, the Pending Interest Table (PIT) keeps track of the forwarded Interests so that the chunks can be returned to their requestor. Finally, a huge cache is used to store the chunk packets using any caching strategy in order to reduce the network response time for frequently asked content. As the content can be delivered by any cache in the network, the time elapsed from sending an Interest to retrieving the corresponding chunk is defined as a random variable $A(t)$. We will call this delay the *Response delay*. In the CCN node model we have proposed in [52] and also considered in this study, we introduced a transmission buffer associated with each face, that differs from the cache.

An important optimization in CCN is the ability to aggregate Interest packets looking for the same content when flowing through a given node. As a consequence, a single Interest copy will be sent to the network and will be used to retrieve the corresponding chunk. In this situation, the router updates an existing PIT entry adding the interfaces that have requested the same chunk and where the corresponding Interest was aggregated (namely dropped). Once a corresponding chunk is received, the node copies it to all

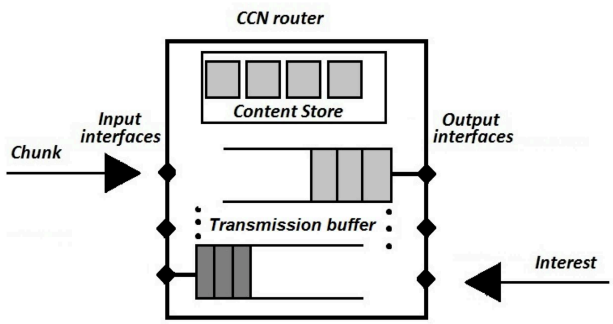


Figure 2.1: The CCN node model

interfaces from this entry. We will call this operation *Multicast*.

2.2.2 CCN node model

The general CCN node model was introduced in [34]. A simple router model is illustrated in Figure 2.1. In this figure, we omit the forwarding information base (FIB) and the pending interest table (PIT). The former is used to forward Interests toward the data source while the later keeps track of the forwarded Interests so that the returned Chunks can be sent to its requestor. We first introduce our model of a CCN router in order to illustrate where congestion can appear.

In [34] the authors advocate that *"CCN does not have FIFO queues between links but rather an LRU memory (the cache)"*. However, in a CCN router, it is important to differentiate the memory allocated for transmiting packets from the one used for caching. If a single Content-Store was used to store both packets waiting for transmission and cached Data chunks, the whole cache will become congested by Data chunks waiting for transmission through the congested interface. The transmission queue is subject to congestion if the Interest aggregate rate arriving in a router single output interface exceeds the link capacity. Finally, it is crucial that at any given time, each output interface schedules which packet is the next one to be transmitted on the physical channel. This can be achieved with the use of per-interface queue rather than a single shared cache.

2.2.3 Discussion

In this section, we explore the particularities of design and the questions that should be answered in future research proposals. Our research is mostly based on traffic control area that is of course depending on others components and architectural features of CCN. So, we are trying to look more into different central features of ICN and, in particular, in CCN such as routing, caching, security and transport layer.

As we mentionned in Chapter 2, in ICN the clients should specify the name of content only and network will fetch it for them from anywhere. Thus, in ICN, routers should forward requests based on the name of the requested content. But that is not easy to provide an effective routing mechanism because, as we mentionned before, the names should be location-independent and the number of content is much larger than the number of hosts. Moreover, due to caching capabilities the content moves in the network much

faster than the hosts in host-centric approach. All of this introduces a scalability problem.

As stated in [63], the unique feature of CCN is the ability of each CCN router to handle network failures locally without relying on global routing convergence. The authors explore the following question does CCN really need any routing algorithm. The work shows that the routing protocols still remain very beneficial for CCN. Hovewer, the routing in ICN and, in particular, in CCN is still an open question and effectiveness of the solutions should mostly answer the question: *"Can it scale?"* discussed in [44]. As stated in [48], even if many papers declare that routing in ICN should be completely different from routing in IP Internet, all existing proposals for routing in ICN use the same old routing algorithms for single-instance destinations. We will briefly describe some routing proposals in Section 2.3.

Earlier in this chapter we have seen that each content in ICN should have a globally unique and persistent name. In CCN, names are human-readable and composed of explicitly identified sequence of components such as: globally-routable name, organizational name and conventional/automatic component for versioning and segmentation. Thanks to hierarchical names the prefix match is equivalent to saying that the Data packet is in the name subtree specified by the Interest packet. Thus, CCN names and signs every Chunk so that the content may be located anywhere and retrieved by anyone. Moreover, the CCN names are location-independent and so the clients could get the content from everywhere and verify its integrity via the digital signature. The caching capabilities are supported by the nature of ICN. In CCN, it is supposed that the caches could be deployed on every CCN router. This idea could potentially reduce the load on the servers, decrease the amount of redundant traffic and decrease the latency. Moreover, almost all proposals about CCN caching are using this idea as the basic one. But there is no answer on the question: is it really necessary to deploy the cache on every router to have an effective caching mechanism? Another point that should be taken into account when designing a caching scheme is how big should be size of the router's cache to maximize the cache performance and keep the reasonable lookup delays. We will briefly explore the work related to CCN caching in Section 2.3.

Another important area for ICN is traffic control that is also a key component of ICN and may depend on concrete ICN architectures. It allows to ensure the efficient and fair network resource utilization. In CCN, the symmetric paths for Interests and Chunks are enforced by the architecture and potentially open a lot of new possibilities for design of traffic control schemes. The receiver driven as well as network assisted approaches are both possible to consider in CCN. We will adress this later in this thesis and we will explore existing solutions in Chapter 2.3. Hovewer, it is important to note that the original proposal of CCN is almost ignoring traffic control and do not specify it. Without this important piece of design, CCN is incomplete. Hence, the congestion control mechanism for CCN is the core addition targeted in this dissertation.

Finally, such key component of ICN as security should also be content-based and so rather different from the one in IP Internet. As mentioned above, in CCN, every client could verify the integrity of the content via its digital signature. It may also be done by every network entity. Moreover, the flow balance between Interest and Chunk together with Interest aggregation scheme prevent from any sort of Data flooding and make the Denial of Service (DoS) attacks more difficult than they are in TCP/IP. A detailed exploration of the CCN security issues is out-of-scope of this dissertation.

2.3 Related work

The CCN framework was first introduced by Van Jacobson and the PARC research group in [34, 24]. Various issues arising in CCN have been considered such as content router issues [11], data transfer modelling [19] or chunk-level caching [20].

In this section we explore the work related to different key components of CCN discussed in Chapter 2.2. We will first briefly describe the existing works on routing and caching, and provide a detailed state of the art on congestion control that is the target key point in this dissertation.

2.3.1 CCN routing

The current proposals for CCN routing are mostly based on extending existing link state routing protocols such as Open Shortest Path First (OSPF) [42] and Intermediate system to intermediate system (IS-IS) [45]. For example, [60] extends OSPF to distribute name prefixes and calculate routes to name prefixes and proposes the OSPF for Named-data (OSPFN). Even if OSPFN is able to compute the shortest path to content, the routers should keep track of a large number of content names. Others disadvantages of OSPFN is that it still uses IP addresses as router IDs, relies on Generic Routing Encapsulation (GRE) tunnels to cross legacy networks, and computes only a single best next-hop for each name prefix. Trying to resolve the problems of OSPFN, the authors propose a Named-Data Link State Protocol (NLSR) [31] which runs on top of NDN. It is still a link-state protocol as OSPF. Link state advertisements are propagated throughout the entire network, each node knows the complete topology and the location of each instance. Even if NLSR has several advantages over OSPFN, it still suffers from scalability problems.

In [57] the authors experiment with blind routing algorithms such as flooding, expanding ring, and random walk implemented over CCN. The authors try to find out optimum fall back scheme for CCN if no match is found in FIB. But the applicability of this mechanisms needs to be verified. Recent work [28] proposes a routing algorithm based solely on distances to named data objects or name prefixes. Authors claims that this scheme does not require any routing information about the physical topology or information about all the replicas of the same content. But this work is still on its initial stage and requires detailed performance analysis.

2.3.2 CCN caching

Standalone and cooperative Web caching have been widely studied in the past([49, 59, 38, 26]). Appearance of ICN has offered new opportunities for caching and it becomes a hot topic in the research community. Different works on in-network caching have been proposed in the litterature. In this section we briefly describe some of them.

In [50] the authors have presented the modelling and evaluation of caching policies based on Markov chains. [56] investigates the potential of caching data at the router-level in CCN. Another work [51] evaluate the cache performance in CCN considering impact of various elements such as network topology, multi-path routing, content popularity, caching decisions and replacement policies.

There are many proposals about content replacement policies. For example, [39] focuses on the mechanisms that spread popular contents to the network edge. In this

proposal, each replica of content has an age. The scheme is based on the following rules: the closer a replica is to the network edge the longer age it has; the more popular a replica is, the longer age it has. A replica is replaced if age is expired or if the cache memory of the node is full. [22] proposes a scheme where the number of chunks to be cached is adjusted based on the popularity of the content (access count). When the access count increases, algorithm exponentially increases the number of chunks to be cached.

Another work [61] try to estimate the benefit from in-network caching for ICN. The authors first formulate the in-network caching problem into Mixed-Integer Linear Programming problem. They then also propose a novel LB (Least Benefit) cache policy and a new forwarding scheme with shallow flooding (FSF for short). They finally show that with in-networking caching, the average hops of the ICN network can be reduced significantly. A lot of other works on caching have been proposed, such as ([37, 55]), but the comparison of the performance of caching algorithms is out-of-scope of this dissertation.

2.3.3 Congestion control in CCN

The work related to traffic engineering in CCN has developed recently. Congestion can appear in CCN and therefore, we promote the utilization of an algorithm to control the level of congestion of CCN routers filled by Data packets. We follow similar principles as in [41], where the authors present ERAQLES, a rate control mechanism for Available Bit Rate (ABR) ATM communications, and define an analytical method to compute the advertised explicit rate to which the sources have to adapt. A similar work is presented in [25]. In [40], the authors describe a rate-based hop-by-hop congestion control mechanism in which a desired service rate is computed at each switch as a function of the target queue occupancy. Feedback information is then exchanged between neighbor switches so that they can dynamically adjust the service rate for each connection.

Hop-by-hop congestion control schemes have been widely studied in the past but only recently in the context of CCN. Traffic control suited to CCN has been considered lately, mostly to study congestion and packet loss in CCN routers. The existing schemes can be classified into two categories: receiver-based and hop-by-hop mechanisms.

2.3.3.1 Receiver-driven congestion control

Some preliminary work on a transport protocol for CCN is presented by S. Arianfar and al. in [12]. This protocol is based on the TCP congestion window principles but uses Data packets as acknowledgments to enforce decisions to increase or decrease the congestion window. The scheme introduced in [16] is one of the first window-based algorithms designed to control the Interest sending rate of a receiver in TCP-like environments. The proposed mechanism uses Chunks packets and associated timers as relevant signals to regulate the number of Interests sent by the receiver. One serious issue of this approach is to properly set the timeout value. The solution used in [16] sets the timer to the mean value of RTT, that, unfortunately, does not take into account the variability of content sources. The mechanism was later improved in [18] by adding route labels to the Chunks so that the client can identify multiple paths. CCTCP introduced in [54] also represents a receiver-driven control solution. The authors propose to list in each interest packet the subsequent Chunks the client intends to request. The network nodes indicate whether they have cached this Chunk. Thus, the receiver maintains separate congestion states for different content locations where future interest will be satisfied. A similar predictive approach has been proposed in [15] where the authors have proposed some improvements

to reduce the complexity of the protocol. Another recent work adressed to resource management in ICN, [13], proposes a deadline-based approach where data packets are assigned with a lifetime. Lifetimes are then used for scheduling and resource management in the network and retransmission logic in the end hosts. Hovewer, this work is still in its initial stage and does not propose a comprehensive solution yet.

As shown in the past, the receiver-driven schemes face a fairness problem in networks with heterogeneous RTTs. Some analysis and improvements can be found in [58, 30]. As caching is a key feature of CCN, the RTT related to the content sources appears as a crucial parameter for receiver-driven mechanisms. A recent work, [14], compares the performance of these schemes and concludes that the receiver-driven solutions based on timeouts are not suitable for CCN because of the unpredictability of the content locations.

2.3.3.2 Hop-by-hop congestion control

A second category of congestion control mechanism uses hop-by-hop interest shaping rather than letting the receiver infer network congestion using timeouts. HoBHIS [52] was the first interest-based shaping mechanism developped for CCN. In [46] the authors present a fair sharing mechanism where Interests exceeding a Data fair rate are discarded. ICP introduced in [16] was then extended in [17] where the authors describe a joint hop-by-hop congestion control mechanism that is found similar to HoBHIS. The basic idea developped in this general approach is to shape the Interest of every conversation flowing through a CCN node to control congestion of Chunks packets and improve network performance. Other proposals based on interest shaping have been presented in [65, 64]. The proposed scheme shapes the interests so that the associated data rate is equal to some predefined ratio of the reverse link capacity. The authors introduces a NACK scheme where an NDN node sends a NACK to the downstream node in case if it can not neither satisfy or forward any given Interest. They also propose a simple coloring scheme for marking the working status of each interface with regard to data retrieval.

Recently, a different solution was proposed in [62]. The mechanism identifies the interdependence between the interests and its associated chunks and analyzes the fair resource allocation in presence of bidirectional traffic. The paper presents the results of our collaboration with “Cisco Systems” and will be presented in details in this dissertation.

Chapter 3

Hop-by-hop Interest shaping

3.1 Controlling the Data rate

Content-Centric Networking (CCN) proposed by Van Jacobson et al. in [34] has been introduced to cope with the evolution of the internet usage towards massive content distribution. This architecture differs from the traditional host-based communication principle in many ways. One important change is that network addresses are replaced by content names to distribute information in CCN networks. Moreover, extensive in-network caching capabilities are introduced to benefit from the observation of the traffic flowing in the network. As a consequence, data packets can be delivered by any CCN router in addition to the source, according to various caching algorithms. In CCN, a request issued by a user is called Interest and the part of the associated content is called Chunk. For convenience, we will name a stream of Interest/Chunk pairs a Conversation.

The key property of CCN is that one Interest retrieves at most one Data packet. It enforces a flow balance in the CCN network that enables multiple Interests to be issued at once. Interest aggregation is an optimization that provides the ability to reduce the network load when multiple interests from different sources request the same content. This is achieved by aggregating them in the CCN router, forwarding a single interest upstream and carrying downstream a single copy of the associated content.

As a consequence of the CCN design principles, we have to regulate the stream of chunks as well as the stream of interests in order to avoid congestion and to improve network performance. We believe that, because of the specific design of CCN, hop-by-hop congestion control provides a suitable solution to this problem. Congestion in a CCN router is defined as the overflow of the buffer associated to an output interface and manifests itself by the loss of data chunks.

The remainder of the Chapter is organized as follows: Our congestion control algorithm is presented in Section 3.1.1. A simple mathematical model is derived in Section 3.1.2 and 3.1.3 in order to highlight the convergence properties of our algorithm. The performance of our solution is evaluated by ns2 simulation and discussed in section 3.1.5. Section 3.1.6 concludes the chapter.

3.1.1 Hop-by-hop Interest Shaping mechanism (HoBHIS)

In this section, we present the hop-by-hop Interest Shaping mechanism (HoBHIS). Every CCN router will control the rate of individual Data chunks conversations by appropriately

shaping the rate of the associated sending Interests.

The congestion control scheme based on hop-by-hop interest shaping was preferred to an end-to-end mechanism such as TCP (Transmission Control Protocol, [32]). Our mechanism is proactive and the share of the network capacity allocated to different conversations is controlled by the algorithm implemented in each CCN node. Interest shaping allows us to anticipate the drop of data packets due to buffer overflow. This is another advantage over TCP congestion control scheme that starts to react only after the drop of one segment, unless a mechanism such as RED is used. In addition, using a hop-by-hop control provides a feedback information more quickly thanks to the smaller distances between hops.

The system under study consists of a set of CCN routers forwarding Interest packets issued by a consumer. As a response, Data chunks are forwarded back to the consumer (namely, the source of the Interests). One Interest corresponds to one chunk of data. Once an Interest has been issued by a given CCN router, the corresponding Data chunk will be piggybacked to that router after a variable delay A(t) named "*Response Delay*", assuming no loss. This parameter differs from the usual RTT from TCP as the Data might be stored in any cache on the path or at the publisher site (the source of the Data). We infer that $A(t)$ should not change drastically on short time scales as it is likely that when the data is stored in a given cache, it should stay there for some time, at least until the network and demand conditions have largely evolved. The objective of HoBHIS is to avoid congestion in CCN Transmission queues by enforcing the queue size to converge to a given objective r, defined as a percentage of the capacity of that buffer. We achieve this objective by *shaping the Interest rate*.

Upon arrival of a Data chunk in the transmission queue, the router computes the Interest rate based on the queue occupancy and the available resources for each conversation. It adjusts the Interest rate according to this information. If the actual number of queued packets is less than some threshold value r, then the router can temporarily increase the shaping rate. On the other hand, if the number of queued packets is higher than r, then the router will decrease its shaping rate.

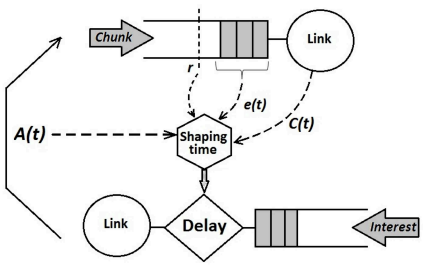


Figure 3.1: Representation of the system

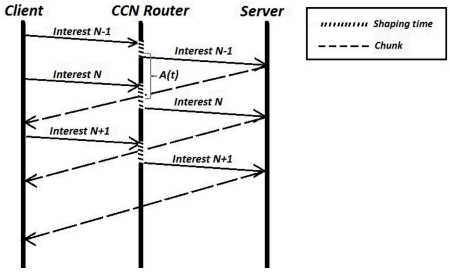


Figure 3.2: Communication process when shaping is enforced

The system under study is presented in figure 3.1. The *Shaping time* component is responsible for the computation of the shaping delay that the Interests packets will have to satisfy. Every Interest packet will be shaped according to the occupancy of the transmission queue, as well as the parameter $A(t)$.

Figure 3.2 provides an illustration of the communication process when shaping is enforced. As can be seen, when an Interest is received, it will be delayed in the CCN

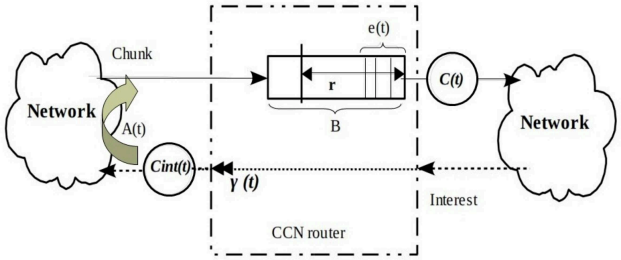


Figure 3.3: Representation of the model

Table 3.1: Notations

$C(t)$	available bandwidth to send the chunks at time t
$C_{int}(t)$	available bandwidth to send the Interests at time t
$\gamma(t)$	shaping rate at time t
$A(t)$	delay from Interest to the related content
$e(t)$	number of queued packets at time t
B	buffer size
r	queue threshold

router if proved necessary in order to avoid congestion of the transmission buffer of that same node.

3.1.2 Single router model

For the purpose of the analysis, we consider the simplified model presented in figure 3.3. It is characterized by a single conversation. In addition, all packets have the same size. The control is applied to each chunk entering the transmission queue. We do not consider any caching policy or routing mechanism. The parameters and notations are given in Table 3.1.

3.1.2.1 Computation of the shaping rate

Let $e(t)$ define the occupancy of a CCN router transmission queue expressed in number of chunks.

We consider the following function that represents the maximum shaping rate at time t, while still being able to control the transmission queue under a feedback delay equal to $A(t)$:

$$\gamma(t) = C(t) + \frac{B - e(t)}{A(t)} \, , or \tag{3.1}$$

$$\gamma(t) = C(t) + h \, \frac{r - e(t)}{A(t)}, \tag{3.2}$$

where h is a design parameter that aims to control the dynamicity of our scheme towards the objective r. In the general case when there are F conversations flowing through a CCN node, we need to divide the available buffer capacity between all active conversations at time t.

It is possible from equation (3.2) that the shaping rate becomes negative. In this situation, the Interests are blocked until either the transmission queue size $e(t)$ becomes less or equal to r, or a Chunk arrives at the queue and the Interest shaping rate is re-evaluated. We use an exponential weighted moving average mechanism to estimate the value of $A(t)$.

3.1.2.2 Resource sharing

In this section we provide a solution for an efficient buffer sharing of the network resources, for the case when multiple conversations are active. Let suppose that there are F active conversations going through the router. In the simplest case the buffer capacity can be shared between the conversations as $\frac{r}{F}$. However, this solution does not take into account the situation where one or more conversations do not consume their share. In this case, the queue length will converge to

$$r' = \sum_{i=1}^{F} [r_i] \leq r, \ i = 1..F \tag{3.3}$$

Each conversation converges to its own threshold r_i. Therefore, as can be seen from Formula 3.3, the total available capacity may not be completely used. In order to improve its utilization, the rest of the resources should be shared between the active conversations. Our solution to solve this issue is described below. The arrival rate for a given conversation can be estimated by the number of chunks in the queue belonging to this conversation. Thus, the rate of a conversation can be expressed as follows:

$$\rho_i(t) = \frac{e_i(t)}{\sum_{j=1}^{F} [e_j(t)]} \tag{3.4}$$

Where $e_i(t)$ is the number of chunks of conversation i at time t and $\sum_{j=1}^{F} [e_j(t)]$ is the total queue length. $\rho_i(t)$ represents the ratio of the total queue length occupied by conversation i at time t. Taking into account equation 3.4, we modify the shaping rate formula in the case of multiple conversations as follows:

$$\gamma_i(t) = C(t) + h.\frac{r.\rho_i(t) - e_i(t)}{A_i^*(t)} \tag{3.5}$$

Monitoring all conversations is far too expensive but we are concentrating on the active conversations only, those for which packets are queued in the buffer, to reduce the number of states stored in the router. It has been shown in the past (cf. [36]) that such per flow control is scalable because of the relatively small number of active flows. A conversation is considered to be active if a transmission queue of the router owns at least one packet of this conversation. Note also that for the sake of simplicity, we consider Chunk packets of constant size. If it was not the case, the length of a chunk packet will be used to adjust the above equation.

3.1.2.3 Queue convergence

In this section we demonstrate that the transmission queue length converges to the objective r. We assume a single conversation that can use the entire link capacity to send

its Data. We define the *Response Delay* for packet number j as A_j^* and for packet $j+1$ is A_{j+1}^*. The queue evolution for a single stream is written as follows:

$$e(t+1) = e(t) + \gamma(t).A_{j+1}^* - C.A_{j+1}^* \tag{3.6}$$

and

$$\gamma(t) = C + h\ \frac{r - e(t-1)}{A_j^*}, \tag{3.7}$$

Based on above we can write :

$$e(t+1) = e(t) - h.\frac{A_{j+1}^*}{A_j^*}.e(t-1) + [h.r.\frac{A_{j+1}^*}{A_j^*}] \tag{3.8}$$

This formula can be simplified as

$$e_{i+1} = e_i + \alpha.e_{i-1} + \beta \tag{3.9}$$

with

$$\alpha = -h.\frac{A_{j+1}^*}{A_j^*},\ \beta = h.r.\frac{A_{j+1}^*}{A_j^*}\ and\ e_{i-1} = e(t-1) \tag{3.10}$$

We write e_{i+1} as

$$e_{i+1} = f_{i+1} - \frac{\beta}{\alpha} \tag{3.11}$$

Then we obtain

$$f_{i+1} = f_i + \alpha.f_{i-1}, \tag{3.12}$$

which is the series of Fibonacci.

So, due to [7] we have

$$f_{i+1} = P.z^{i+1} \tag{3.13}$$

where z is the root of equation

$$z = z^0 + \alpha = 1 + \alpha \tag{3.14}$$

Thus, we need that

$$|z| < 1 \tag{3.15}$$

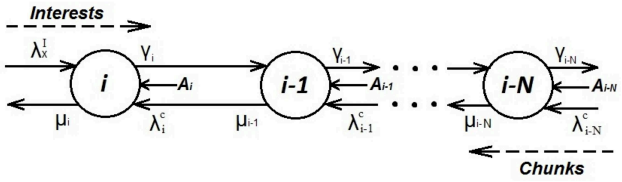


Figure 3.4: Representation of the model

Table 3.2: Notations

λ_i^c	arrival Chunks rate to node i
λ_x^I	arrival Interests rate
μ_i	service rate of node i
γ_i	shaping rate of node i
A_i	delay from Interest to the related content for node i
$e_i(t)$	number of queued packets at time t of node i
B	buffer size
r	queue threshold

and if i tends to infinity then f_{i+1} converges to 0. But the condition $|z| < 1$ will be true only if :

$$0 < h < 2.\frac{A_j^*}{A_{j+1}^*} \tag{3.16}$$

And finally,

$$\lim_{i\to\infty} e_i = \lim_{i\to\infty}[-\frac{\beta}{\alpha}] = r \tag{3.17}$$

We found that the transmission queue is converging to r as expected. We will observe that we can face a burst of Data Chunks during the initialization period. This is because our algorithm starts only after the reception of the first data packet. Until then, the system sends the packets with the maximum available rate. This problem is solved by limiting the initial sending rate according to the shaping rate formula.

We now consider ***F conversations***. Our shaping rate formula is defined by:

$$\gamma_i(t) = C(t) + h.\frac{r.\rho_i(t) - e_i(t)}{A_i^*(t)} \tag{3.18}$$

It appears that the queue size for each flow converges to $\rho_i(t)$. So, the total queue size will converge to r as expected.

3.1.3 Network of nodes

We now extend our simple initial scenario to the case of a network of nodes. This model, described in Figure 3.4, consists of N nodes using HoBHIS. We have now a hop by hop congestion control mechanisms using the shaping rate computed at each router. The

notations are presented in Table 3.2. We start with a single conversation flowing through the system. The *Response Delay* is defined by parameter A. The Chunks arrive into the nodes with the rate λ^c chunks/second.

3.1.3.1 Dependency of delay from Interest to Chunk

The Response Delay A of one node depends on downstream nodes up to the router caching the Data or the source itself. Let's define A_i and A_{i-1} the Response Delays for nodes i and $i-1$ respectively. A_i can be expressed as follows:

$$A_i = A_{i-1} + \sum_{j=1}^{X} \frac{1}{\gamma_{i-1}^j} + \frac{e_{i-1}}{\mu_{i-1}} \tag{3.19}$$

The Response Delay for a given packet p in node i depends on the number of packets X queued for transmission ahead of p towards the downstream node and also depends on the number of packets e_{i-1} queued before the response will be sent-back to node i. The quantity $\sum_{j=1}^{X} \frac{1}{\gamma_{i-1}^j}$ means that we recompute the shaping rate γ_{i-1} for each Chunk entering the queue and hence the Interests could be sent with differents rates.

3.1.3.2 Convergence property

We have demonstrated for a single router model that, if the minimal rate among all nodes is γ_i, associated to node i, then the queue of node i converges to r. In addition, we know that

$$\lim_{t\to\infty} e_i^{\gamma_i}(t+1) = r \; and \; \lim_{t\to\infty} e_i^{\gamma_k}(t+1) = r' \tag{3.20}$$

So, according to the limit inequality theorem we have: $r' \leq r$ for all other routers sharing the same conversation's paths as router i.

3.1.4 Dynamic adjustment of the design parameter h

In this section we analyse in details the behaviour of HoBHIS and study the dynamic adjustment of the design parameter h. This parameter is very important because it has a significant effect on the dynamic convergence properties. Let us study it with a simple analytical model.

In this model, we consider a single-conversation going through the CCN node and constant response delay $A(t) = A$. HoBHIS starts to react only after the reception of a data chunk. During an initialisation period, the Interest sending rate of any CCN node equals its maximum possible rate to send the Interests, namely:

$$\gamma_{max,t_0} = \frac{C}{S_I} \tag{3.21}$$

where S_I is the size of an Interest. The oscillations in the data queue might be large during the initialisation period due to a high Interest sending rate. Two situations are possible: case 1) an incoming Interest rate, γ_{in}, is higher or equal to γ_{max,t_0}, and case 2) γ_{in} is less than γ_{max,t_0}.

In the first situation, the router will send a huge number of Interests into the network that may lead to large oscillations in the data queue. Oscillations due to the initialisation

period can not be avoided but the parameter h will be controled in order to stabilize the system as fast as possible.

For the first case where $\gamma_{in} \geq \gamma_{max,t_0}$, the solution is to quickly serve the Chunks in excess. Let's call $e_0(t)$ the number of Interests sent during the initialisation period. We know that for the case where $e_0(t) > r$:

$$h\frac{r - e_0(t)}{A} < 0 \tag{3.22}$$

Thus, to have $\gamma > 0$, we should maintain:

$$|h\frac{r - e_0(t)}{A}| < C \tag{3.23}$$

Thus, we have:

$$|h| < |\frac{CA}{r - e_0(t)}| \tag{3.24}$$

When a first Chunk arrives to the node, we can easily compute h using Formula 3.24. That allows us to make the queue converge to its objective faster and without future oscillations. The only problem we can face here is the shaping rate that immediately jumps from a large value to a small one. In order to reduce this jump of the shaping rate, we can try to maintain it between $\gamma_{stable} \pm \delta$, where δ is defined from:

$$|\gamma(t) - \gamma_{stable}| \leq \delta \tag{3.25}$$

Where $\delta = \epsilon * stable\_value$, $\epsilon \in (0; 1)$ that corresponds to $\delta = \epsilon\gamma_{stable} = \epsilon C$ in our case. Thus, to maintain γ in these bounds we can use $h = \epsilon h_{opt}$. Frequently, for control systems $\epsilon = 0.05$.

Let us study the second case where $\gamma_{in} < \gamma_{max,t_0}$. We know that:

$$h\frac{r - e(t)}{A} > 0 \tag{3.26}$$

It means that in practice the Interest sending rate $\gamma(t) = min\{\gamma_{max,t_0}, \gamma_{in}\} = \gamma_{in}$. When the first Chunk arrives, the shaping rate $\gamma(t)$ will be computed and expressed in number of Chunks per second and then should be applied to the Interests. Hovewer, the queue size is still less than the objective value r. Therefore, the design parameter should adjust the shaping rate $\gamma(t)$ to γ_{in} until the queue converges. We should also take into account the number of Interests that have been sent during the initialisation period and will be quickly returned as Chunks, N_I. Therefore,

$$\gamma_{in} = C + h\frac{r - N_I}{A} \tag{3.27}$$

Thus, the value for h can be derived from this equation and will be:

$$h = \frac{(\gamma_{in} - C)A}{r - N_I} \tag{3.28}$$

3.1.5 Performance evaluation

The aim of this section is to analyze, through simulations, the performance of HoBHIS. We have started our simulations by developing the CCN module in *Network Simulator* 2 (ns_2). According to our knowledge, there was no version of CCN in ns_2 at the time of writing.

Later, we ported our implementation to ns_3-based Named Data Networking Simulator (ndnSIM) [3] that implements Named Data Networking (NDN) communication model. The source code is publicly available on *github.com* and can be easily found on: ***https://github.com/Be1thaz0r/HoBHIS.git***.

In this section, we evaluate the performance of the CCN Interest shaping mechanism using our implementation of the mechanism in ns_2 and we then use our implementation in ndnSIM to evaluate the behavior of HoBHIS with dynamic adjustment of the design parameter h.

3.1.5.1 Simulation scenario for HoBHIS

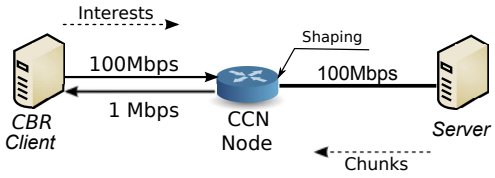
We start with the single router model. We consider both the single conversation and the multi-conversation scenarios presented in Figure 3.5(a) and 3.5(c) respectively.

As depicted in figure 3.5(a) and 3.5(c), our network consists of the client, router and server components. The client sends the Interests to the router that takes the next decision according to the shaping mechanism. The server role is very simple, as for each received Interest it responds with the corresponding Chunk and sends it back to the router. The shaping rate is computed according to the algorithm presented in previous sections. The buffer parameters are B=500 Chunks, r = 100 Chunks, h = 0.1, 0.4 and 0.7. The clients generate Interests with rate 833.3 Chunks/s. The links rates are given in figure 3.5(a) and 3.5(c). The bottleneck is the 1Mb link between the client and the router. Parameter $A(t)$ is a random value uniformly distributed in [0,1] and it is generated in the server for every packet to take into account the variability of the path between the router and the server.

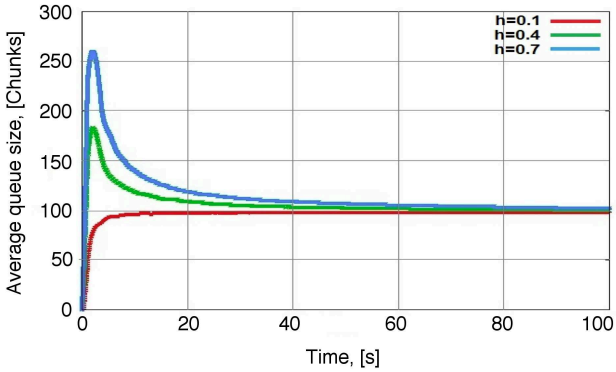
The Network topology used for the evaluation is presented in figure 3.6(a). We consider two conversations from clients 1 and 2 respectively. Data for conversation 1 is in the cache of node 3. As Chunks coming from cache of node 3 can be sent directly and very fast (Interests are not shaped and Response delay $\simeq$ 0), we add some small fixed delay to each Chunk coming from cache in order to avoid a possible overflow of the Chunk output queue. We provide a detailed explanation in Chapter 5. Conversation 2 flows from Client 2 to the Server. The interest rate is 2500 Chunks/s for each client. The parameters for the buffers are B=500 Chunks, r=250 Chunks and h=0.7. The capacity of each link is 2500 Chunks. The conversation 2 starts before conversation 1 and after some time it stops. We are interested in the buffer state for node 3, as well as the rate for each conversation over time.

3.1.5.2 Simulation results for HoBHIS

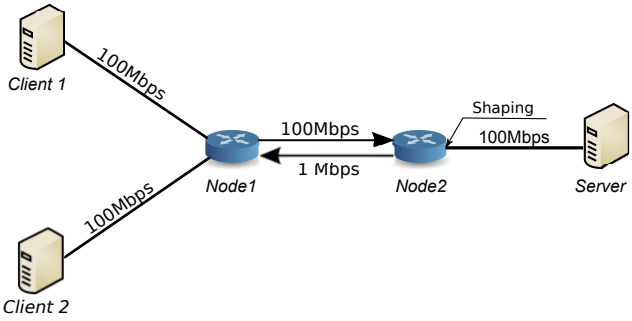
The simulation results for the single conversation scenario are presented in Figure 3.5(b). We have tested our algorithm for different values of h. In Figure 3.5(b) we can see that the queue converges to the objective r as expected (cf. in Section 3.1.3). The bursts in these curves are due to the initialization period when the algorithm is not yet in operation. The different values of h illustrates its influence for the control mode and convergence rate.



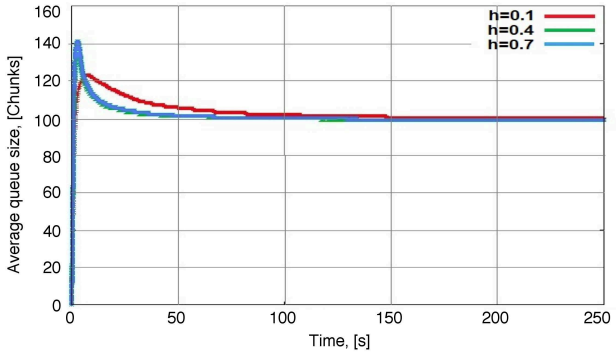
(a) Topology for single conversation scenario



(b) Single conversation scenario

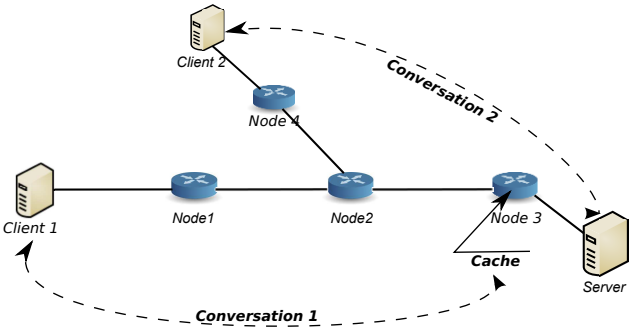


(c) Topology for multi conversation scenario

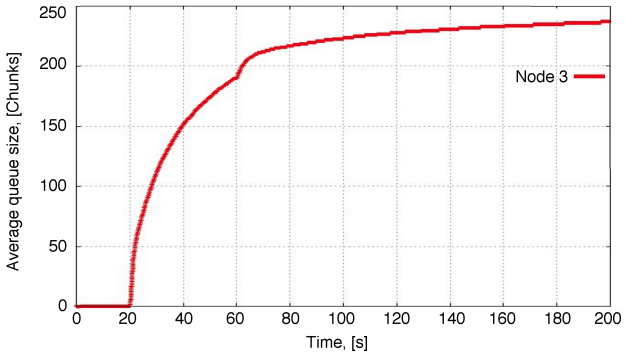


(d) Multi conversation scenario

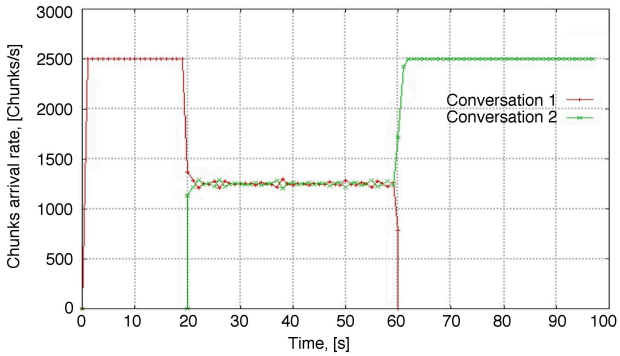
Figure 3.5: Queue convergence for mono and multi conversation scenarios



(a) Network topology



(b) Chunk queue state of node 3



(c) Chunks rate for each conversation

Figure 3.6: Simulation results for network scenario

The higher h the slower the convergence rate towards the objective and the harder the control. The situation repeats for the multi conversation scenario in Figure 3.5(d). The average queue size converges to r as expected (cf. Section 3.1.3).

Figures 3.6(b) and 3.6(c) illustrate the network scenario. As we use the same capacities for each link, the queue of every node is empty until conversation 1 starts. Then, the queue for node 3 starts to fill up and converges to the objective. Figure 3.6(c) illustrates the Chunks arrival rates for each conversation in Chunks/s. At time 20, conversation 1 becomes active. At time 60, conversation 2 becomes inactive. Between time 20 and 60, when the two conversations are active, each of them receives a faire share of the router resources. The Chunks arrival rate, shared between the two conversations are controlled by the minimal shaping rate. In figures 3.6(b) and 3.6(c) we see that our congestion control mechanism is able to adapt the rate and to maintain the queue length at the expected level as it was designed.

3.1.5.3 Resource sharing

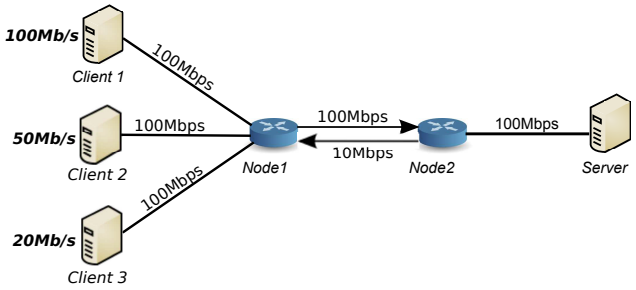
We consider now the multi-conversation scenario and observe the share of buffer capacity allocated to a given conversation. The network topology for this scenario is presented in Figure 3.7(a) and consists of three Clients asking for different contents with different rates. We are interested in observing the buffer state of node 2 containing the Chunks of all active conversations. As the Tolerance mechanism is not used in this scenario, the assymetric link is configured between two nodes in order to have only one bottleneck for Data packets and avoid congestion due to possible Interest buffer overflow. The Interest and Chunk packet's sizes are set to 40 and 1500 bytes respectively. The buffer parameters are as follows: total buffer size =500 Chunks, r=300 Chunks, h=0.7. The Clients 1, 2 and 3 are continuously sending Interests with 100Mb/s, 50Mb/s and 20Mb/s respectively.

We compare two possibilities for buffer sharing. In the first one we share the buffer capacity according to $\frac{r}{F}$, while the second one allows each conversation to share a part calculated thanks to formula (3.4). In the first case, it can be seen from Figure 3.7(b), that the total queue length is less than the expected threshold r. It means that the available resources are not completely used.

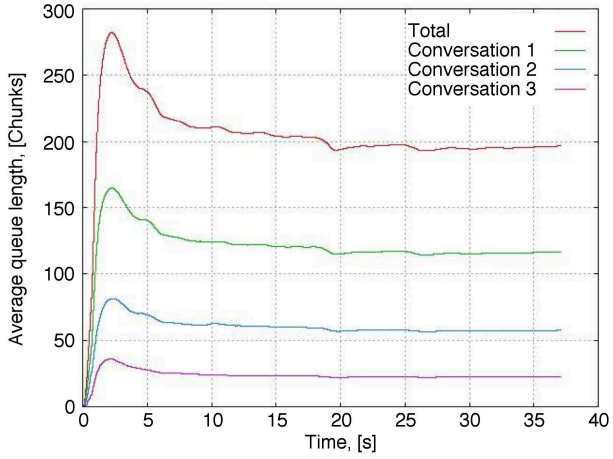
The second possibility of buffer sharing is demonstrated in Figure 3.7(c). Unlike in the previous case, we observe that the buffering resources are fully utilized. The buffer capacity left by any conversation is shared among other ones and the total queue length converges to r as expected by HoBHIS.

3.1.5.4 Dynamic adjustment of h

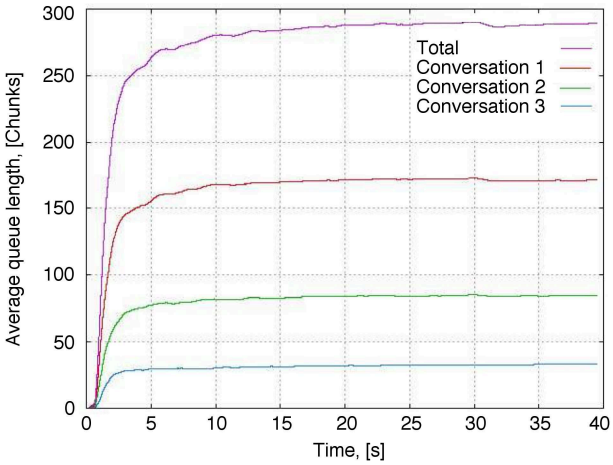
In this section we analyze the convergence rate of our mechanism. We use a simple mono-conversation scenario and one-node topology presented in Figure 3.8. The $A(t)$ is considered to be constant for these experiments. We are interested in the convergence rate of the data queue of the bottleneck link and the shaping rate towards their objectives that are $r = 60$ and $\gamma = C$, under different values of h. We study two cases presented earlier in Section 3.1.4, where the first one causes the big oscillations in the data queue after an initialization period while in the second case the data queue is not sufficiently filled. To generate the oscillations in the data queue during the initialisation period we will use the Client Interest sending rate = 5.000 Interests/s that is higher than γ_{max} in order to use the maximum possible Interest sending rate of a CCN node. For the second



(a) Network topology



(b) Average queue length using $\frac{r}{F}$ to compute a conversation rate



(c) Average queue length using dynamic buffer sharing to compute a conversation rates

Figure 3.7: Resource sharing

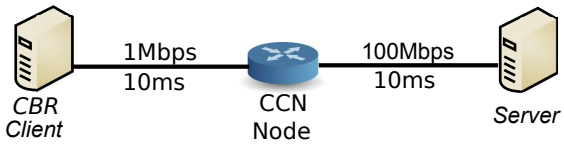


Figure 3.8: Simple topology to study the influence of the design parameter

case we reduce the Client sending rate to 1.000 Interests/s in order to have $\gamma_{in} < \gamma_{max}$.

The curves 3.9(a) and 3.9(c) show the results for the case where the data queue is overloaded due to an excess number of Interests sent during the initialization period. Tuning the value of h we observe the different behavior of the Chunk queue. It is easy to see that the optimum value of h dynamically adjusted thanks to the algorithm presented in Section 3.1.4 provides the best convergence rate to the objective value without oscillations whilst other values of the design parameter do not provide a suitable control. For simplicity, we provide a theoretical shaping rate that may become negative.

The results for the second case are presented in Figures 3.9(b) and 3.9(d). Again, h_{opt} provide a suitable convergence speed towards the objective. The small and large h values cause slow convergence and oscillations respectively.

3.1.5.5 Influence of the Response Delay variations

This section provides an analysis of the influence of the Response delay in the performance of HoBHIS. Indeed, the Response delay is an important parameter as it defines the control loop and has a strong influence on the dynamic of the system. In addition, as the content can be cached anywhere along the path followed by the conversation, this value can change over time. However, we believe that once a content is cached, the likelihood that it will be moved with short time scale is scarce and that the control loop will be mostly influenced by the variation of the delays experienced on that path. The simulation topology is provided in Figure 3.10(a). The buffer objective r for this scenario is fixed to 300 Chunks, the maximum buffer size $B = 50000$ Chunks. We consider that $A(t)$ is a random value uniformly distributed in [0,1] seconds unless otherwise stated. We consider two possible scenarios.

3.1.5.5.a Erratic behavior of the Response Delay

This scenario represents the analysis of the behavior of HoBHIS in the case of an erratic behavior of the Response Delay. If it happens that $A(t)$ suddenly becomes very large with respect to its current estimate (the content has been moved far away), it will take time for our mechanism to adjust the shaping rate to the new value of the control loop. During the time of the new Response delay (equal to 3 seconds in our simulation), the Chunk queue will decrease but the shaping rate is not yet re-evaluated. After 3 seconds a burst of Chunks will arrive because the shaping rate was over-estimated leading to a sharp increase of the transmission buffer. After a little while, the queue will converge again to its objective thanks to the accurate estimate for $A(t)$, demonstrating the efficiency of our solution.

Figure 3.10(b) illustrates the simulation results for this scenario. We can observe that the queue oscillates around r because of the random character of the Response delay. The two bursts on this curve are due to the sudden growth of $A(t)$ as described above.

3.1.5.5.b Large variations of the Response Delay between consecutive packets

In this scenario we consider a different situation where $A(t)$ presents large variations between consecutive packets. $A(t)$ is set to 0.001 seconds for odd packets and 3 seconds otherwise. A prediction mechanism based on the historic of the last 20 values of $A(t)$ is used.

Figure 3.10(c) shows the simulation results for this case. At the origin of the curve, we observe oscillations of the transmission buffer because of the insufficient historic of $A(t)$ to stabilize the queue. Then, $A(t)$ is smoothed and the queue size converge as expected by HoBHIS.

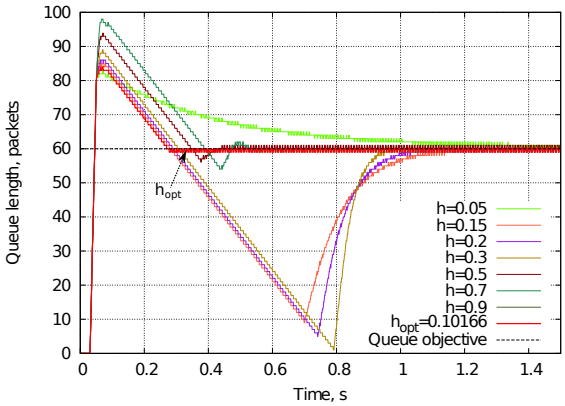
3.1.5.5.c Distributions of the Response Delay

We evaluate now he influence of different distributions of the Response Delay on the behavior of HoBHIS. The topology for this experiment is presented in Figure 3.11(a). In this Figure, the Server node delays the Chunks according to different random distibutions. We compare three distributions: uniform, exponential and pareto. The delays are randomly distributed from 0 to 0.1s. We observe the Chunk queue evolution in the CCN node. The buffer sizes are set to 100 packets for Interest and Chunk queues, the queue objective is set to 60 Chunks. HoBHIS and Tolerance mechanism are in operation in the CCN node. Figures 3.11(b), 3.11(c), 3.11(d) presents the results for uniform, exponential and pareto distributions respectively. We observe almost the same behavior in the queue of the bottleneck node for every distribution. The oscillations in the Chunk queue are due to the random delays generated by Server for each Chunk. The oscillations in the Interest queue are due to the Chunk queue and feedback delay between the Client and the CCN node.

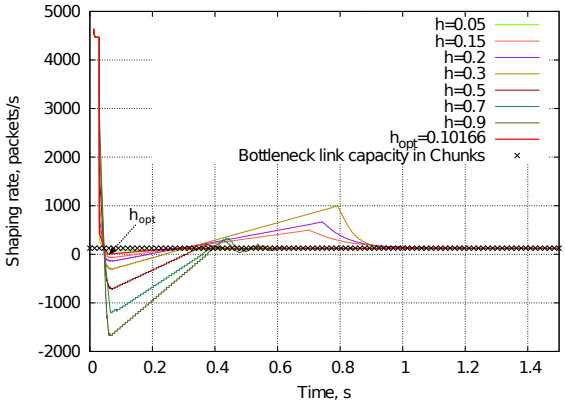
3.1.6 Conclusion

We presented (HoBHIS), the first hop-by-hop Interest shaping congestion control mechanism designed to avoid the congestion that can occur in the output interface of a CCN node. HoBHIS monitors the transmission buffers of a CCN router to compute the Interests rate that have produced the associated Chunks filling these interfaces.

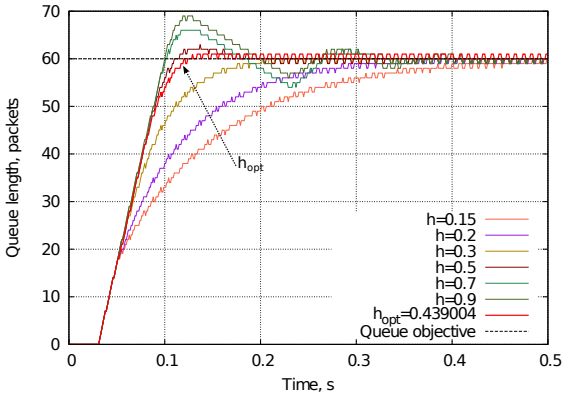
We demonstrated analytically the convergence property of our algorithm. We performed various experiments with different settings and progressive complexity. We analyzed the single and multiple conversation scenarios in a single router model. Finally, a network case was studied to demonstrate the behaviour of our algorithm in more complex conditions. We have seen that the shaping mechanism performs as designed.



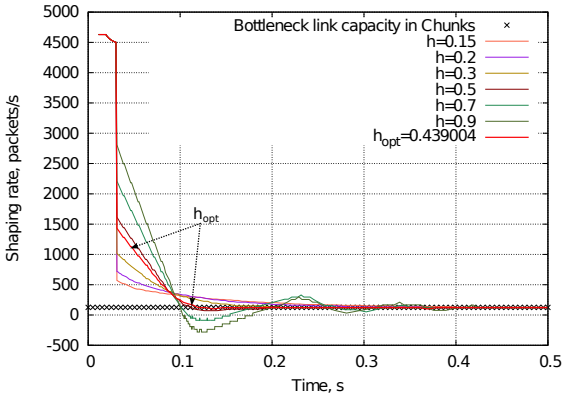
(a) Chunk queue behavior under different values of h, case 1



(b) Shaping rate behavior under different values of h, case 1

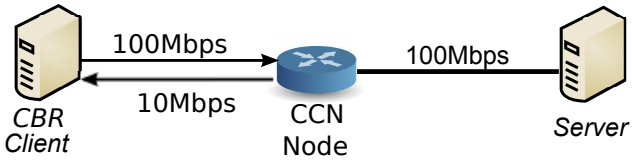


(c) Chunk queue behavior under different values of h, case 2

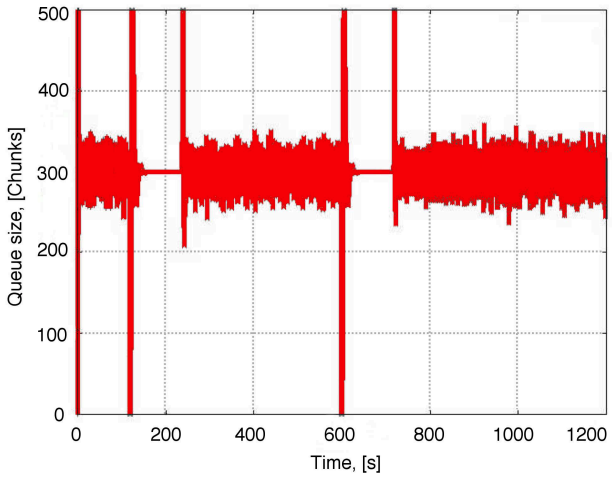


(d) Shaping rate behavior under different values of h, case 2

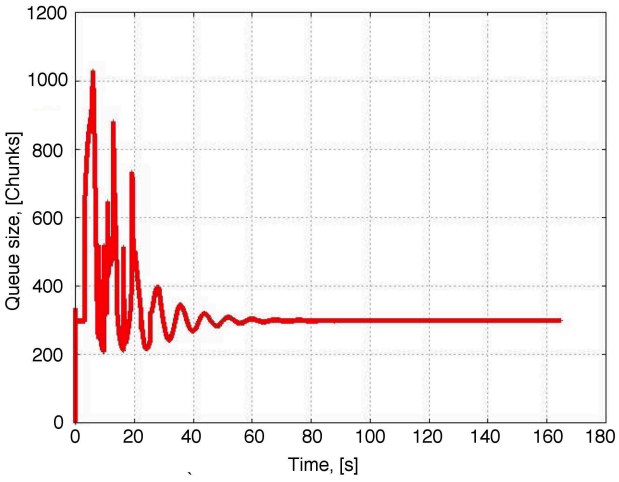
Figure 3.9: Chunk queue length and Interest shaping rate convergence speed as a function of h



(a) Topology for analyzing the Response Delay variations

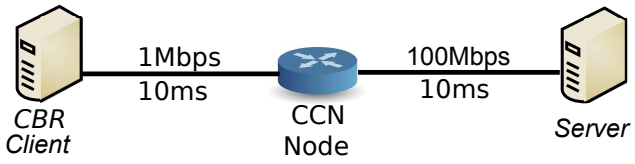


(b) Queue size as the function of time for Scenario 1

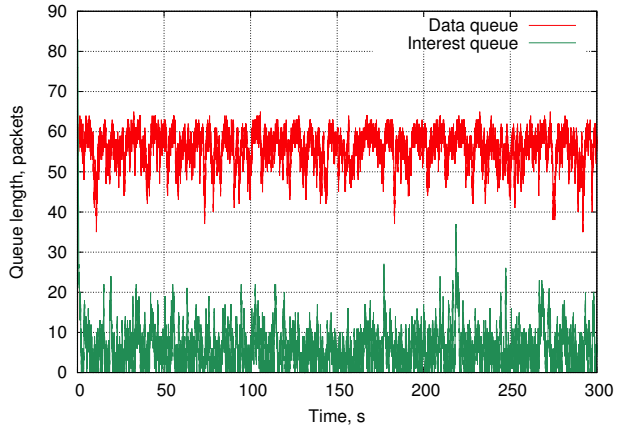


(c) Queue size as the function of time for Scenario 2

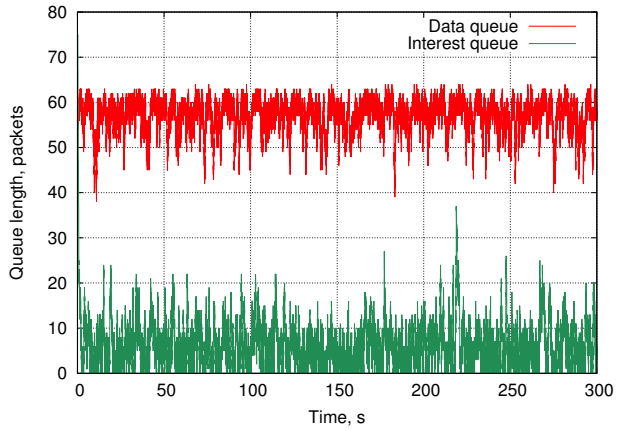
Figure 3.10: Influence of the Response Delay variations



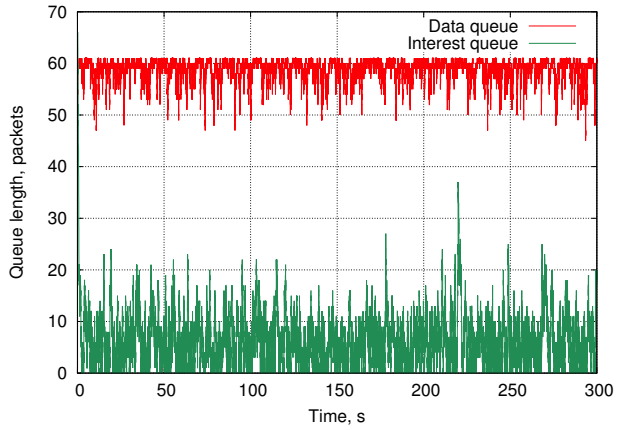
(a) Topology for analyzing different Response Delay distributions



(b) Queue size as the function of time with uniform distribution of RTT from 0 to 0.1s



(c) Queue size as the function of time with exponential distribution of RTT from 0 to 0.1s



(d) Queue size as the function of time with pareto distribution of RTT from 0 to 0.1s

Figure 3.11: Influence of the Response Delay variations

Chapter 4

Tolerance rate mechanism

4.1 Controlling the Interest rate

The shaping rate formula allows us to control the congestion of the transmission queue by carefully monitoring the rate of the Interest packets in the CCN nodes. However, there is no mean to prevent a Client from sending Interest at high or excessive rates in order to be privileged and retrieve his content faster. Therefore, it is important to define a tolerance mechanism that, in addition, can control the Clients as well as prevent the loss of Interest packets.

There exist two basic methods to throttle the rate of a client: rate-based and window-based. Each of them has their own merit and both approaches are feasible in CCN. In the case of window-based, the client's behavior is strictly defined by the presence or absence of Data packets. If no feedback arrives, it will be interpreted as congestion in the network and the client will automatically stop sending. However, window-based control schemes can lead to traffic bursts. Moreover, if the window is too small, the network resources are not effectively used. An adaptation of this method is investigated in [16] and represents a variant of Additive Increase Multiplicative Decrease (AIMD) for CCN. As the authors use Data Chunks to increase/decrease the congestion window, the estimation of the response time is crucial for this type of control. In [46], the authors also choose to use AIMD in order to adjust the Interest sending rate used by clients.

Ideally, Clients should fully saturate the link, send packets and keep the Chunks arriving continuously. To achieve this, the window-based control has to estimate an ideal window size that can be computed based on the bottleneck capacity and the Response Delay. The rate-based control has to know the bottleneck capacity or should receive an explicit rate value from the network in order to adapt its sending rate. One advantage of the rate-based control is that it does not stop sending in absence of feedback, but at the same time the client has to be aware of a specific sending rate. This rate might be known thanks to the introduction of a network feedback mechanism or by using an instantaneous measured value of the input Data rate.

For a CCN client it is important to choose an initial value for the sending rate. It may be a small fixed window for window-based approach or any specific rate for rate-based schemes.

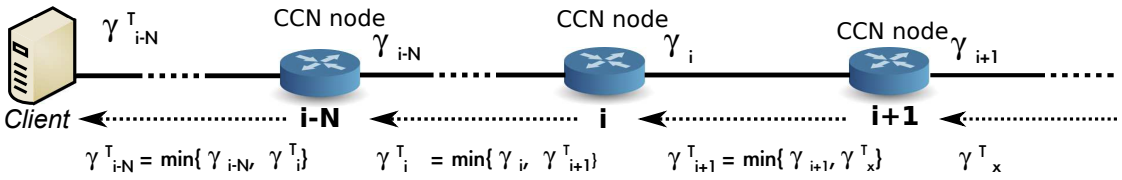


Figure 4.1: Network model

4.1.1 Tolerance rate

In this section we present the Explicit Interest Rate control mechanism that defines a tolerance with respect to the interest rate that a Client can generate. A rate-based scheme was preferred to a window-based mechanism such as AIMD because it is independent of the end user's strategy. In addition, it allows a more accurate control and better network resource utilization.

We define the tolerance rate as the maximum rate that the clients are not allowed to exceed. The principle of our solution is to adjust the tolerance rate to the rate of the bottleneck node. Every CCN router will control the rate of incoming Interests by periodically sending control packets with an explicit rate field advertised back to the Clients. This explicit rate field aggregates the bottleneck rate on that path up to this router and is advertised upstream towards the source.

In our model, two complementary rates are computed for each individual conversation: the shaping rate and the tolerance rate. Thanks to HoBHIS, every CCN router controls the rate of a conversation by shaping the rate of its associated Interests. In addition, control packets will be exchanged and updated by the routers along the path followed by a conversation, to convey the tolerance rate for every single Client. For sake of simplicity and without loss of generality, we will consider a network topology illustrated in figure 4.1. Each $A(t)$ seconds, CCN nodes transmit control packets with an explicit rate information back to the client. Each router calculates the maximum allowed rate for the Client and updates this field if it is found smaller than the actual value. As a consequence, the control packets carry the most conservative value for the rate, namely, the maximum allowed rate of the path. It is obviously not necessary to update the explicit rate faster than the delay of the control loop $A(t)$.

4.1.2 Computation of the Tolerance rate

Let Formula 4.1 represent the maximum shaping rate for conversation i at time t. The Interest queue contains Interests packets from many different conversations. The control function used to compute the shaping rate of a given conversation enforces the total Chunks queue length to converge to the objective r defined as a percentage of the buffer capacity.

$$\gamma_i(t) = \min[\max[C(t) + h\,\frac{r\rho_i(t) - e(t)}{A_i^*(t)}, 0], C_{int}(t)], \tag{4.1}$$

As we do not know a delay between the bottleneck router and the client, we can not directly use the shaping rate formula to compute the tolerance rate because it may lead to large Interest queue oscillations. Moreover, the shaping rate formula does not depend on the Interest queue length at time t and so the level of Interests in the queue is not controlled. As a result, it would be possible to have an Interest queue always congested.

Thus, the tolerance rate formula should: 1) depend on $\gamma(t)$; 2) use the filling level of Interest queue at time t.

Respecting the conditions listed above, we propose the following computation formula for the tolerance rate:

$$\gamma^T(t) = \gamma(t).(1 - \frac{e^I(t)}{B^I}) \tag{4.2}$$

The factor $(1 - \frac{e_I(t)}{B_I})$ should decrease the oscillations due to feedback delay and the variations of the shaping rate. Using formula 4.2 to calculate the explicit rate value, allows us to maintain the total arrival Interest rate aligned with the shaping rate.

4.1.3 Dealing with packet loss

We have seen earlier that we can face the losses of both Data and Interest packets. As one Interest corresponds exactly to one Chunk, the loss of one Interest reflects the loss of one data Chunk. Therefore, it is important to react as soon as possible to a loss, and therefore, a loss detection mechanism and the actions to be performed by a Client or CCN routers should be defined.

Each CCN router has a Pending Interest Table (PIT) containing information about the Interests that are waiting for the reception of the corresponding Chunks. Because of Interest aggregation it becomes useless for a source to retransmit. In order to avoid this situation, the current CCNx implementation considers the PIT timers to delete an entry waiting for a long time.

A few papers have addressed this problem. In [16] and [17] the timers are used by the Clients to allow the retransmission of Interests. At the same time, CCN routers use the PIT timers defined by the CCNx implementation. However, for the algorithm described in [16], the retransmission timer must be larger than the PIT timers of CCN routers, otherwise, all retransmissions become useless. Another contribution, [46], presents a mechanism of packet loss detection where a congested router sends a header of data packet to the content consumer. This approach reduces the delay of loss detection. Nevertheless, an implicit feedback is also used by default via a timeout that is interpreted as congestion.

In our solution, we recommend to use the features provided by the current CCNx implementation ([1]) because the PIT timers guarantee that the Interests that have been waiting for a long time will be deleted from the PIT after a timer expiration. It allows the Clients to detect a congestion via a timeout and react by the retransmission of the corresponding packet. However, the computation of the timer values remains to be solved.

4.2 Performance evaluation

The aim of this section is to analyze, through simulations, the performance of HoBHIS and Tolerance rate mechanism. We use our implementation of these mechanisms in ns3-based Named Data Networking Simulator (ndnSIM) that implements Named Data Networking (NDN) communication model.

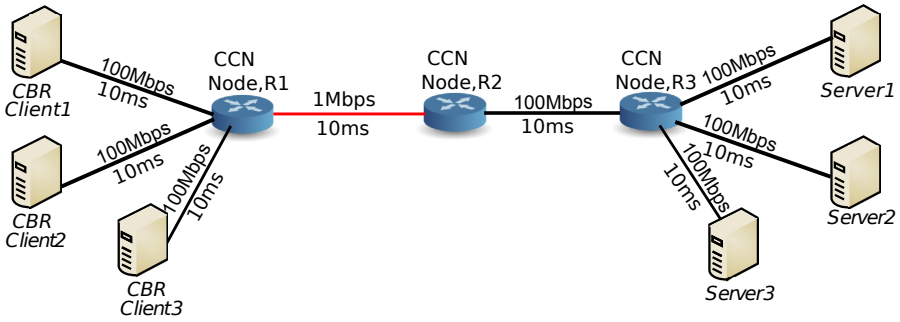
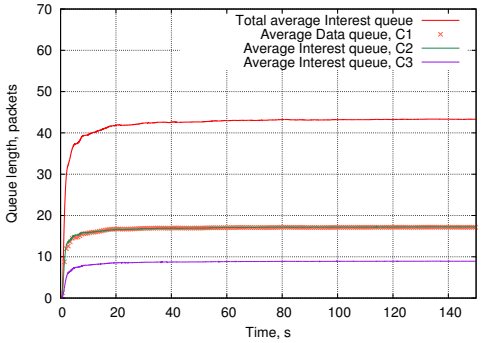
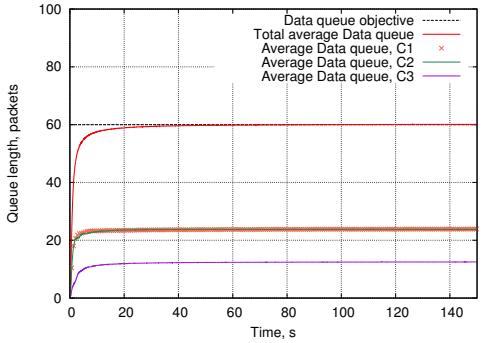


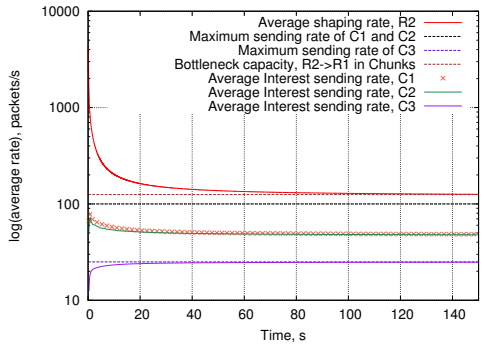
Figure 4.2: Network topology to study the Tolerance rate mechanism



(a) Interest queue length under shaping and Interest rate control



(b) Chunk queue length under shaping and Interest rate control



(c) Conversation rates

Figure 4.3: Hop-by-Hop Interest shaping and Tolerance mechanism in operation

4.2.1 Tolerance mechanism

In this section we analyze the performance of our feedback mechanism that is used to enforce the tolerance rate. The simulation topology is presented in Figure 4.2. In this scenario, Clients 1, 2 and 3 are asking for the content from Server 1, 2 and 3 respectively. The shaping mechanism is implemented in each CCN node. Every CCN node has a timer set to a Response Delay value and send the control packets according to the algorithm defined in Section 4.1. The servers represent the rest of CCN network and generate the random delay from 0 to 0.1 s for every Chunk packet. All of the clients are not greedy and have the maximum sending rate for 100 Interests/s for Clients 1 and 2 and 25 Interests/s for Client 3 they can not exceed. When the explicit rate value is retrieved by the Clients,

they adjust their sending rate to this value. Their rates are updated every $A(t)$ seconds with the values obtained from the control packets. We are interested in observing the Interest and Chunk buffer states in bottleneck node R2. The buffer parameters are: buffer size = 100 Chunks, r = 60 Chunks. The control packet's size is equal to the Interest packet size that is around 30 Bytes, the Chunk payload size is set to 1000 Bytes.

The results are shown in Figures 4.3(a) and 4.3(b). As it can be seen from the Figures, the total average Chunks queue length converges to the threshold r = 60 Chunks as expected by HoBHIS. We can see that the conversations fairly share the buffer capacity. The Clients 1 and 2 have the same percentage of buffer capacity attributed to them because they are emitting with the same rates. As the sending rate of Client 3 is slower than the rates of other two Clients, it does not need the same amount of buffer capacity that Clients 1 and 2. Thus, we see that the part of router ressources that is not used by Client 3 is attributed to Clients 1 and 2. We observe the same behavior of the Interest queue where the resources are fairly shared between the conversations. Figure 4.3(c) presents the client rates and the shaping rate of the bottleneck node. The rate of Client 3 achieves its maximum value at 25 Interests/s. As the Clients 1 and 2 are able to emit faster than Client 3, their tolerance rate is higher. The sum of Client rates is equal to the bottleneck node's shaping rate. Finally, as a result, we observe optimal resource utilization and no packet loss.

4.3 Conclusion

We presented a first comprehensive solution for congestion control in CCN networks. This problem is of utmost importance and has not been globally addressed in the past. Our framework is grounded on our original HoBHIS mechanism that was the first introduced to provide a hop-by-hop shaping mechanism. It nicely exploits the flow balance enforced in CCN between Interest and Chunk packets. It mostly consists in monitoring active conversations sharing the transmission buffer of a CCN node face in order to dynamically adjust their Interest sending rate and enforce the Chunk queue length to converge to a defined objective. This mechanism is implemented in each CCN node. We extended the design of HoBHIS in order to address the important concerns that might occur in CCN. We first demonstrated the fairness of resource sharing among competing conversations. Second, we introduced an explicit Interest rate feedback mechanism designed to control the Client behavior and prevent a potential risk of network congestion. Each node will compute a Client tolerance rate that is returned upstream towards the source in order to collect the bottleneck rate of the path. A thorough evaluation was conducted using different simulation scenarios. We observed that the results fully satisfy the design objectives and we can conclude that HoBHIS is an efficient and operational solution to the problem of congestion control in CCN.

Chapter 5

Managing complex scenarios

In chapter 3 we have seen that HoBHIS allows us to control the congestion of the transmission queue by carefully monitoring the rate of the Interest packets in the CCN nodes. However, there is no mean to prevent a Client from sending Interest at high or excessive rates in order to be privileged and retrieve his content faster. Therefore, it is important to define a tolerance mechanism that, in addition, can control the Clients as well as prevent the loss of Interest packets. In chapter 4 we introduced the Explicit Interest Rate control mechanism that defines a tolerance with respect to the interest rate that a Client can generate. A rate-based scheme was preferred to a window-based mechanism such as AIMD because it allows a more accurate control and better network resource utilization.

The structure of this dissertation is a step-by-step design and evaluation of the congestion control scheme. The aim of this chapter is to study in details the behavior of our mechanisms in different possible advanced scenarios that may happen in CCN. We study such situations as multicast, traffic-split, cross-traffic and influence of caching. It is important to analyse the reaction of our solutions and to propose the strategies to manage all of the cases. The objective is to improve our solutions in order to develop a unified control that will take into account all possible situations.

5.1 Multicast

This section is dedicated to the study of the behavior of HoBHIS in the situation of multicast conversations. This scenario is important as it often happened that the same content is retrieved by multiple Clients and optimizations are required in such situations.

Two or more Interests asking for the same content must be aggregated by a CCN router. These Interests might come from heterogeneous Clients and environments, for instance having different link capacity. However, only a single copy of such request will be kept in the buffer and processed by HoBHIS. An important issue being to define which parameters should be used in formula (3.5) to compute the shaping rate for an aggregated set of requests.

According to the state of the art, we have to adjust to the most conservative environment, namely, we shall use the smallest bandwidth and the largest output queue length from the heterogeneous conversations. The shaping rate will be calculated as follows:

$$\gamma(t) = \min\{C_1, ..., C_N(t)\} + h.\frac{r - \max\{e_1, ..., e_N(t)\}}{A^*(t)} \tag{5.1}$$

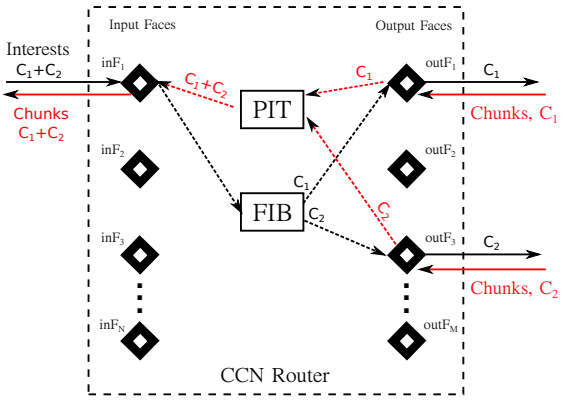


Figure 5.1: Representation ot the traffic split in CCN network

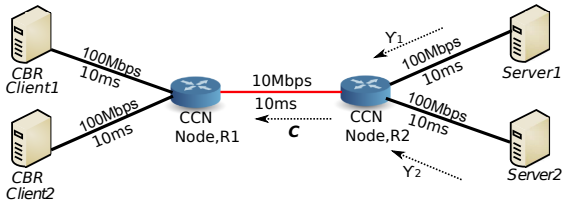


Figure 5.2: Representation ot the traffic split in CCN network

Where N represents the number of Clients asking for the same Chunk.

If all links have the same capacity, then all buffers will be controlled similarly and the queue lengths will converge to the objective r as expected by HoBHIS. We provided a detailed performance evaluation in Section 3.1.5.

5.2 Traffic split

In this section we discuss traffic split and its influence on our scheme. **Traffic split** is a situation where the incoming Interest traffic is splitted between different output faces such that it is shown in Figure 5.1. In this figure, the Interests are arriving to the input face inF_1 and asking for different content C_1 and C_2. After a FIB lookup for the output faces, the traffic is splitted between $outF_1$ and $outF_3$. Later, the returning Chunks will be forwarded and mixed on inF_1. This situation is important because the shaping rate is based on the common Chunk queue length of inF_1 but applied to the independent output faces $outF_1$ and $outF_3$. In the worst case, this may cause the overflow of the output Chunk buffer because the expected threshold r won't be respected.

5.2.1 Problem description

In this section we provide a simple example in order to better explain the importance and consequences of traffic split. The Figure 5.2 depicts such a scenario. Two Clients, Client 1 and Client 2, are asking for the contents C_1 and C_2 located in Servers 1 and 2 correspondingly. The Interests of two conversations are mixed on the same output face of router R_1 and share the link capacity between $R_1 - R_2$. The mixed traffic arrives then to R_2 where it is splitted in order to reach the Server 1 for C_1 and the Server 2 for C_2.

Table 5.1: Notations

$C(t)$	available bandwidth to send the chunks at time t
$C_{int}(t)$	available bandwidth to send the Interests at time t
$\gamma(t)$	shaping rate at time t
$\gamma^T(t)$	tolerance rate at time t
$A(t)$	delay from Interest to the related content
$A^*(t)$	predicted delay from Interest to the related content
$e(t)$	number of queued Chunks at time t
$e^I(t)$	number of queued Interests at time t
$e_i^I(t)$	number of queued Interests of conversation i at time t
B	buffer size
B_I	Interest buffer size
r	queue threshold
F	number of active conversations
h	design parameter

HoBHIS is implemented in every router of the path. The output link $R_2 \rightarrow R_1$ is the bottleneck to send the returning Chunks. We describe the chunk queue behavior of the router R_2. The notations are given in Table 5.1.

In order to maintain the chunk queue length around the objective value r, HoBHIS computes the Interest shaping rate and supposes that the returning Chunk rate will be around this value. In our example depicted in Figure 5.2, the returning Chunk rate should ideally be as follows:

$$\gamma_{in}(t) = \gamma_1(t) + \gamma_2(t) = C + h\frac{r - e(t)}{A^*(t)} = \gamma(t) \tag{5.2}$$

Where $\gamma_{in}(t)$ is the total incoming Chunk rate to the router R_2, and $\gamma(t)$ is the Interest shaping rate.

Let us check if the Condition 5.2 is satisfied; for our example. According to HoBHIS, the shaping rate on the independent output faces for C_1 and C_2 will be computed as follows:

$$\begin{aligned} \gamma_1 &= C + h\frac{\frac{e_1(t)}{e(t)}r - e_1(t)}{A_1^*(t)} \\ \gamma_2 &= C + h\frac{\frac{e_2(t)}{e(t)}r - e_2(t)}{A_2^*(t)} \end{aligned} \tag{5.3}$$

Suppose that $A_1^*(t) = A_2^*(t) = A$.

$$\gamma_1 + \gamma_2 = 2C + h\frac{r - e(t)}{A} \neq \gamma \tag{5.4}$$

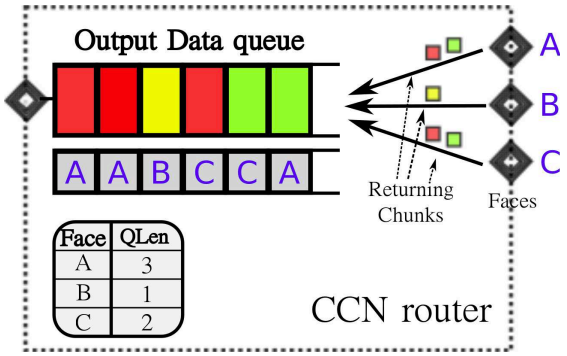


Figure 5.3: Solution to manage traffic split

Thus, in general case with M conversations whose Interests are splitted between different faces :

$$\gamma_{in} = MC + h\frac{r - e(t)}{A} \neq \gamma \tag{5.5}$$

As we can see, the Condition 5.2 is not respected. The returning Chunk rate is higher than an expected value. It is easy to demonstrate that the Chunk queue is converging to $r' > r$.

5.2.2 A solution to manage traffic split

As we have seen above, the traffic split has an influence on the behavior of HoBHIS. It happens because the output faces are managed independently. Thus, we need to share the output Data link capacity according to the number of output interfaces to send Interests in order to satisfy Condition 5.2. Moreover, the solution should be directly included in the shaping rate formula and should not consider any traffic split maintaining mechanism. We also want our solution to be conversation-independent and should not keep the huge history about sent Interests.

To do so, the shaping rate formula should contain the following parameter:

$$\gamma_i(t) = \kappa_i C + h\frac{\rho_i r - e(t)}{A} \tag{5.6}$$

Where κ_i is computed as follows:

$$\kappa_i = \frac{N_{OutFace_j}}{e(t)}, \tag{5.7}$$

$N_{outFace_j}$ is the number of Chunks arrived from an $OutFace_j$ and actually being located in the output data queue.

In this solution we do not need to observe the traffic split and maintain different modes of our scheme to manage special situations. Instead, we always suppose that there is a traffic split and maintain per-interface queue length. The only information we need is to know from which interface we just received a Chunk.

Figure 5.3 illustrates the unified solution taking into account the traffic split. The Chunks of three conversations : red, green and yellow, are mixed in the same data output buffer. The different Chunks of red and green conversations are coming from the interfaces A and C. The yellow conversation arrives from the interface B. As it is depicted in the figure, every interface of the CCN router should maintain a structure like a table Face/QLen illustrated in the picture in order to know how many Chunks we get from each interface and what part of the buffer capacity they are occupying. The maximum number of lines in this table equal to the total number of interfaces in the router minus 1. The number of such tables is equal to the number of router interfaces. When a Chunk arrives into the Data queue from any given interface, the counter $QLen$ for this interface is increased by 1 and decreased when the packet is dequeued. At the same time, we should maintain the markers or labels that show us the interface from which this packet has been received. It can be implemented as a simple queue structure and have the same size and behavior as the Output Data queue. This structure will keep the markers of the input interfaces for every Chunk in the Data output buffer. A marker is added when a Chunk is enqueued and deleted when the packet is dequeued. The order of markers repeats the order of Chunks in the queue.

The presented solution also takes into account the situation where any given conversation is splitted between different faces. In this case, the packets will have the different interface markers and HoBHIS won't be confused computing the shaping rate. Thus, the final shaping rate formula becomes as follows:

$$\gamma_i(t) = \min[\max[\kappa_i C(t) + h\,\frac{\rho_i r - e(t)}{A_i^*(t)}, 0], C_{int}(t)], \tag{5.8}$$

with k_i

$$\kappa_i = \frac{N_{OutFace_j}}{e(t)}, \tag{5.9}$$

$$\sum_{j=1}^{N_{faces}} \kappa_j = 1 \tag{5.10}$$

where i is the conversation and j is the output interface where a given Interest is located at the moment of the computation of its shaping delay.

5.3 Cross-traffic

Previously we have discussed the problem of traffic split that happens when the Interests arrive to the same input face and then splitted between different output faces. In this section, we discuss the dual case where the Interests arrive to different input faces and then mixed into the same output face of a CCN router.

Figure 5.4 illustrates this situation. In this figure, the Interests asking for the content C_1 and C_2 are arriving to the input faces inF_1 and inF_2 correspondingly. They are then forwarded to the output interface $outF_2$ where they are mixed in the transmission buffer of this interface. The returning Chunks will so be splitted between the inF_1 and inF_2. It is easy to see that the output Chunk queues are independent from each other. Thus, we

should compute the shaping rate independently. But this is not a case when the Interests are mixed in the same queue. One of the conversations saturating its output Chunk queue will penalize other conversations.

Suppose that the Chunk queue of inF_1 is congested. According to HoBHIS, the shaping rate for the conversation C_1 will tend to 0. At the same time, even if the Chunk queue of inF_2 is behaving correctly, the Interests of the conversation C_2 will be blocked by C_1. That is not suitable behavior and we should carefully manage this situation.

5.3.1 Managing cross-traffic

In this section we propose a possible solution to manage cross-traffic. It consists in maintaining on each output interface one virtual queue per input interface. Thus, the total maximum number of virtual queues will be equal to the total number of possible input interfaces. We can manage them independently according to the shaping rate values. Thus, if C_1 is blocked, we can still forward the Interests of C_2 according to its shaping rate value.

5.3.2 Problem description and solution

In this section we try to clearly demonstrate why it is important to manage the cross traffic. We compare the behavior of HoBHIS by default, i.e. in case where the cross traffic is not managed, and using our solution proposed in section 5.3.1.

We consider two traffics i and j going through a CCN router like it is depicted in Figure 5.4. In the first case we mix interests of both conversations in the same output buffer so that one conversation depends on another. The second case consists in using the virtual queues and managing the conversations independently.

Suppose now that we always have Interests in the buffer, i.e. every time we have something to send. To keep it simple, we consider the Interests of same size and constant Response delay $A(t) = A$. Figure 5.5 illustrates an output Interest queue containing two conversations i and j that will be delayed by the shaper prior to be sent. As the Interests are arriving from different input interfaces, the shaping rate for theses conversations will be computed as follows:

$$\gamma_i(t) = C_i + h\frac{r - e_i(t)}{A} \tag{5.11}$$

$$\gamma_j(t) = C_j + h\frac{r - e_j(t)}{A} \tag{5.12}$$

Let us illustrate this in figure 5.5. We observe N_i packets of conversation i before j. The shaping delay for each of them is:

$$\tau_i(t) = \frac{1}{\gamma_i(t)} \tag{5.13}$$

Ideally, τ_j should also be computed with a similar formula:

$$\tau_j(t) = \frac{1}{\gamma_j(t)} \tag{5.14}$$

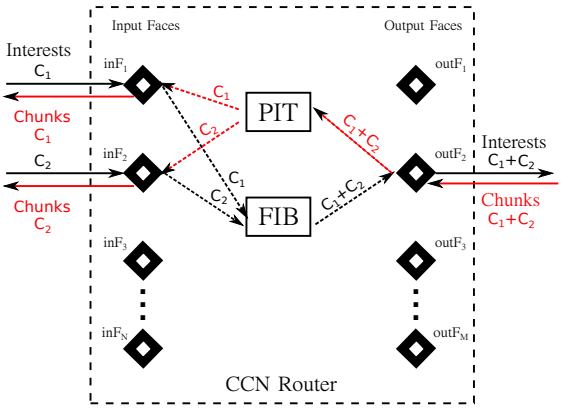


Figure 5.4: Representation of cross-traffic in CCN network

But in reality, packet j is waiting $N_i * \tau_i$ before being delayed on τ_j by the shaper. To keep it readable, we will omit the dependence of time. Taking the traffic i into account we have a real delay for packet j:

$$\tau'_j = \frac{1}{\gamma_j} + \frac{N_i}{\gamma_i} > \tau_j \tag{5.15}$$

It is easy to see that the real shaping delay for j is much higher than the expected one.

We now introduce the virtual queues and manage the traffic like it is depicted in Figure 5.6. Let us study the real and expected shaping delay for packets of the conversation j (the conversation i will have the same behavior).

$$\tau'_j = \begin{cases} \frac{1}{\gamma_j}, & if\ \tau_j \leq \tau_i \\ \frac{1}{\gamma_j} + (N_i\tau_i - \tau_j) & if\ \tau_j > \tau_i \end{cases}$$

Merging this two cases we obtain:

$$\tau'_j = \frac{1}{\gamma_j} + \max\{0, (N_i\tau_i - \tau_j)\}, \tag{5.16}$$

where τ_j is computed thanks to the formula 5.14. N_i is the number of Interests of conversation i that we can send over τ_j.

Generalising this formula to the case where the shaping delay of conversation i is changing during τ_j, we obtain:

$$\tau'_j = \frac{1}{\gamma_j} + \max\{0, (\sum_{k=0}^{N_i} \tau_{i,k} - \tau_j)\}, \tag{5.17}$$

It is easy to see that the result from formula 5.16 is smaller than the one of formula 5.15. It means that virtual queues provide a more efficient use of router resources.

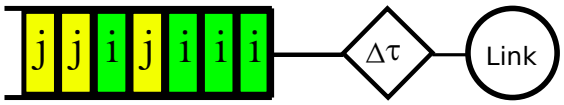


Figure 5.5: Output Interest queue with mixed Interests of two conversations i and j

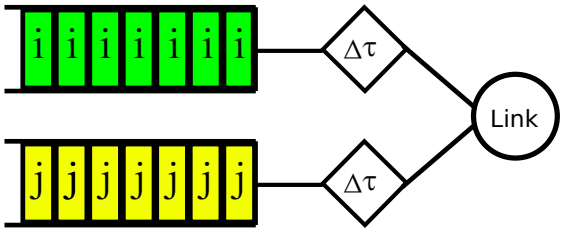


Figure 5.6: Virtual Interest queues

5.4 Influence of caching

In previous chapters we discussed a congestion caused by high returned data rate. However, we did not take into account the content returned from cache. Caching in CCN may also cause congestion in routers because the output data rate from the cache is upperbounded by the delay of cache lookup only and mostly equal to the arrival Interest rate. Thus, if the arrival Interest rate is high, the output Chunks queue will quickly be overloaded by the content going from cache. Thus, even if all Chunks of any given conversation are located in the cache, we can not send them all immediately due to possible buffer overflow. This situation is especially dangerous for bottleneck nodes but has never been carefully explored in the existing works.

As we have seen earlier, our hop-by-hop Interest shaping mechanism is applied to the output Interest queues and, thus, can not correctly manage the Chunks returned from cache. Therefore, in this section we extend our mechanism to solve this problem. We propose a scheme to protect the output Chunk queue from congestion caused by the cache.

We propose to maintain the output cache queues per interface and shape Chunks instead of Interests. We also consider that the cached content is the popular one and should be privileged. However, we must take into account others conversations and buffer sizes. Thus, we add a condition to check the presence of the asked content in the Content Store. The proposed solution is depicted in Figure 5.7. For every Interest arrived from face i (A, B or C in the Figure), we check if we can lookup the corresponding content in the cache. The condition is the filling level of the cache queue corresponding to the input face. If the corresponding cache queue is full, the Interest is forwarded normally. Otherwise, we check the Content Store for asked Chunk. To improve memory utilization, the cache queues may contain only the pointers to the given Chunks instead of the complete packets. This saves the memory, however, if any given packet has been replaced in the cache before to be sent, it may lead to packet loss.

5.4.1 Computation of the shaping rate

In this section we explain the shaping rate computation process. The situation discussed in this section may be presented as a particular case of traffic split and will have the

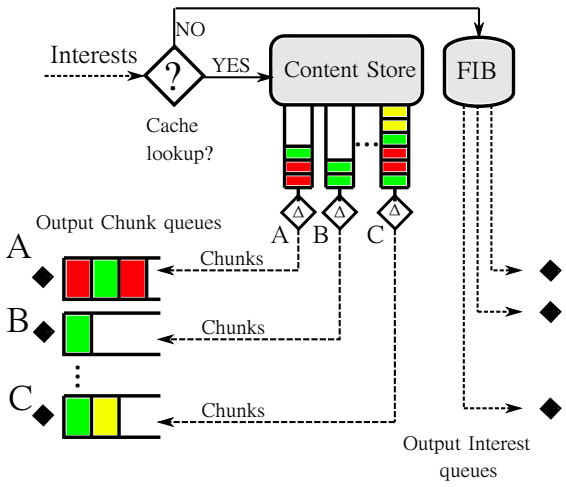


Figure 5.7: Representation of shaping scheme for Chunks coming from the Content Store

similar control as the Formula 5.6 in Section 5.2:

$$\gamma_i(t) = \kappa_i C + h\frac{\rho_i r - e(t)}{A^*_{min}}, \tag{5.18}$$

where A^*_{min} is the parameter that theoretically tends to 0 because the content is getting from cache. It means that the sending rate from the cache should tend to ∞ that is not possible in practice. Moreover, such a large output rate will quickly saturate the output Chunk buffer and lead to congestion. If this parameter is very small, it means that the cache is not optimally used. We propose to select a value that is less than the minimum $A(t)$. Choice of this parameter should depend on how much we want to privilege the cached content.

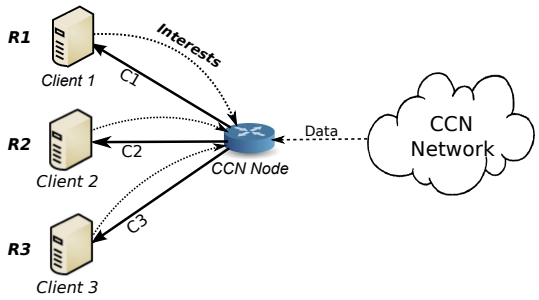
5.5 Performance evaluation

In this section we evaluate the performance of the proposed solutions using our implementation in *ndnSIM*. We analyse our solution to Multicast and evaluate the effectiveness of the schemes proposed in this chapter.

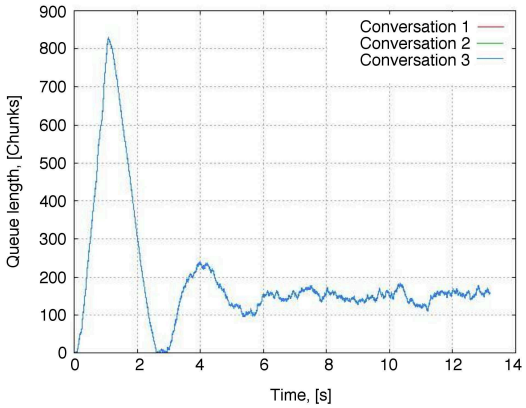
5.5.1 Multicast

The aim of this part is to study the behavior of HoBHIS when Interest aggregation is in operation. The network topology for this experiment is depicted in Figure 5.8(a). In this scenario three Clients ask for the same content continuously sending Interests with the same rate equal to 60Mb/s. Key parameters are set as: maximum buffer size is 500 Chunks, r=150 Chunks and h=0.7. All other parameters are unchanged. We study three possible scenarios:

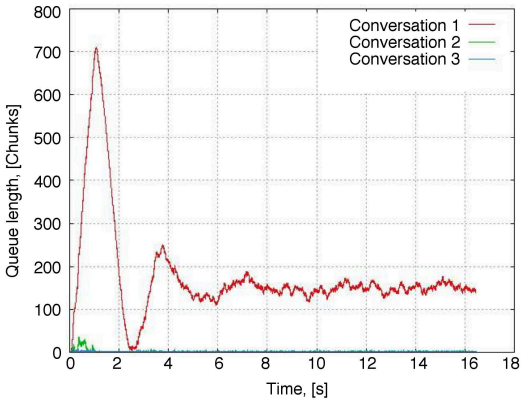
1. The Clients have the same rates (60 Mb/s) and the same link capacities (10Mb/s).
2. The Clients have different link capacities $C_1 < C_2 < C_3$ ($10Mb < 20Mb < 30Mb$) but the same rates (60Mb/s)



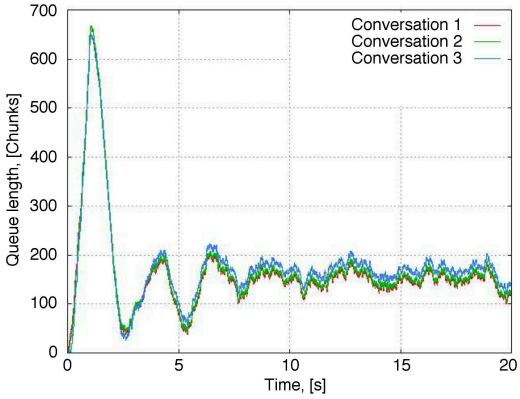
(a) Network topology



(b) Queue length over time for Scenario 1 (same rates / same badwidths)



(c) Queue length over time for Scenario 2 ($C_1 < C_2 < C_3$)



(d) Queue length for Scenario 3 ($R_3 < R_2 < R_1$)

Figure 5.8: Multicast

3. The Clients have differents rates $R_3 < R_2 < R_1$ ($30Mb/s < 50Mb/s < 100Mb/s$) and the same link capacities (10Mb/s)

We are interested in the queue length of each active conversation at time t. We observe the transmission buffers associated to the output faces of a CCN node. We do not consider any caching policy nor routing mechanism. The random value of the response delay parameter $A(t)$ is uniformly distributed between 0 and 1 s.

5.5.1.1 Scenario 1: The Clients have the same rates and the same link capacities

In this scenario, the clients ask for the same content with their Interest sending rates set to 60Mb/s. The output link capacity of the CCN node is also equal to 10Mb/s for every output interface. We are interested in the queue length of each conversation on the corresponding faces. The results are illustrated in Figure 5.8(b). Thanks to the Interest aggregation, every queue converges to r ($r = 150$ Chunks). So we have exactly the same queue's behavior for every conversation.

5.5.1.2 Scenario 2: The Clients have different link capacities but the same rates

In this scenario, to compute the shaping rate we use the smallest bandwidth between all the links and the biggest queue length as defined in Section 5.1 (cf. Formula 5.1). We are interested in the queue length of each conversation. Figure 5.8(c) represents the results of this experiment. We observe that the queue of conversation 1 with the minimum bandwidth converges to r while the queues of conversations 2 and 3 remains empty. No losses are generated and the bandwidth of the smallest link is effectively used.

5.5.1.3 Scenario 3: The Clients have differents rates but the same link capacities

The last scenario is similar to Scenario 1 but with different client sending rates. Figure 5.8(d) represents the queue size for each output interface of a CCN node. We observe that all queues converge to r thanks to the Interest aggregation and the shaping performed in CCN nodes.

5.5.2 Traffic split

In this part, we analyse the performance of HoBHIS in traffic split condition. We compare the behavior of HoBHIS with and without the solution proposed in Section 5.2. The simulation scenario is depicted in Figure 5.9(a). In this scenario, Clients 1 and 2 are asking for the content from Server 1 and 2 respectively. HoBHIS is implemented in every router of the path. The client rates are set to 1000 Interests/s, Data buffer size is set to 100 Chunks and Data queue threshold $r = 60$ Chunks. We suppose a constant Response delay $A(t) = A$. As the Tolerance mechanism is not used for this scenario, Interest buffer size is set to large value in order to avoid the losses of Interest packets caused by buffer overflow. We are interested in the Chunk queue length of the bottleneck router R_2.

Figures 5.9(b) and 5.9(c) represent the simulation results. Figure 5.9(b) illustrates the behavior of HoBHIS without traffic split solution while Figure 5.9(c) use the scheme presented in Section 5.2. We can observe that without correctly managing the traffic split, the Data queue is converging to a different threshold $r' > r$. In Figure 5.9(c) the shaping

rate is computed thanks to the formula 5.8. This control takes into account traffic split and makes the Data queue converge to the objective r. At the same time the fairness between the conversations is also achieved.

Let us now change the rate of one of the clients. Suppose now that the Client 2 is transmitting Interests two times slowly than Client 1. The results of such a scenario are shown in Figure 5.9(d). We still observe the expected optimal and fair behavior of HoBHIS.

5.5.3 Cross-traffic

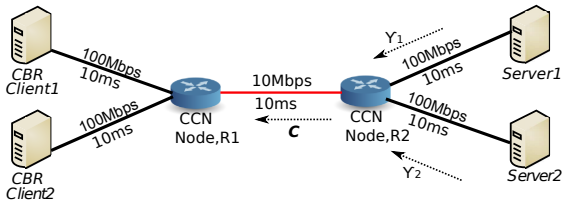
We now analyse the performance of HoBHIS in presence of cross-traffic. The network topology is presented in figure 5.10(a). Two Clients 1 and 2 are asking for the content from Server 1 and 2 respectively. We see that the Interests will be mixed in router R_2 and share the same output link capacity while the returning Chunks will be splitted between different interfaces. HoBHIS is implemented in each router. In order to keep it simple, we do not use the Tolerance rate mechanism for this simulation scenario. The rates of the clients are set to 150 Interests/s each. The HoBHIS parameters are as follows: Chunks buffer size = 100 Chunks, Interest queue length = 100000000 Interests (i.e. big enough to enqueue everything during the simulation), r = 60 Chunks, h = 0.7. We suppose a constant Response delay. We are interested in observing the chunk queue lengths of the bottleneck links R2->R1 and R2->R4.

Figures 5.10(b) and 5.10(c) represent the results for this simulation scenario without and with virtual queues respectively. In Figure 5.10(b) we observe that the chunk queue does not converge to its objective. That is due to large real shaping delay that differs from the one computed by HoBHIS (rf. Section 5.3.2). Figure 5.10(c) shows the chunks queue lengths over time using the virtual queues. We observe that they are converging to the objective as expected by HoBHIS. At the same time, the fair ressource sharing is respected. We can conclude that the virtual queues is an efficient solution to manage the behavior of our shaping mechanism in presence of cross traffic.

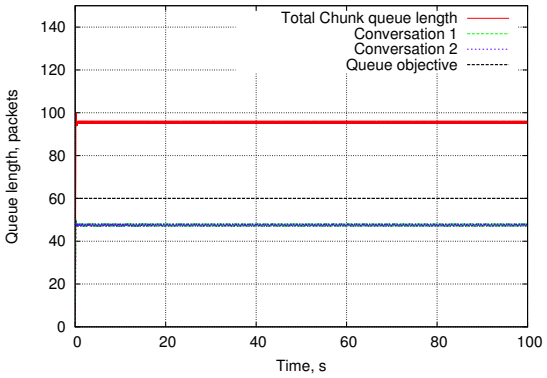
5.5.4 Influence of caching

In this section, we study the influence of caching on the behavior of our mechanisms. The simulation scenario is presented in Figure 5.11(a). Four Clients are asking for different content at different time. The Client 1 is active from 0 to 50s, the Client 3 starts at 0 and stops at 70s that is a little bit later than Client 1. The Client 2 becomes active at 55s and asks for the same content that the Client 1. It stops then at 200s while Client 4 is still active from 55s to 300s. During the period when Client 1 is active, the CCN node $R2$ caches the Chunks of this conversation that passes through the router. Thus, the Client 2 mostly retrieves the corresponding Chunks from the cache. The Response time for content delivered from Servers is uniformly distibuted between 0s and 0.1s. We are interested in observing the queue behaviors in the bottleneck router R2. HoBHIS and the Tolerance rate mechanisms are used in every router of the path. The buffer parameters are: B = 100 Chunks, h = 0.1, r = 60 Chunks. We use the same buffer parameters for cache buffer and A_{min} is fixed to 0.0001s. The Clients initial rates are set to 500 Interest/s and controlled by Tolerance rate mechanism.

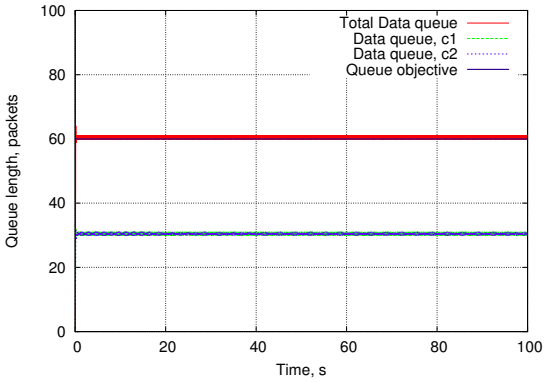
The results for this scenario for time period [0; 55s] and [55; 300s] are presented in Figures 5.11(b) and 5.11(c) respectively. We observe in figure 5.11(b) that the Clients 1



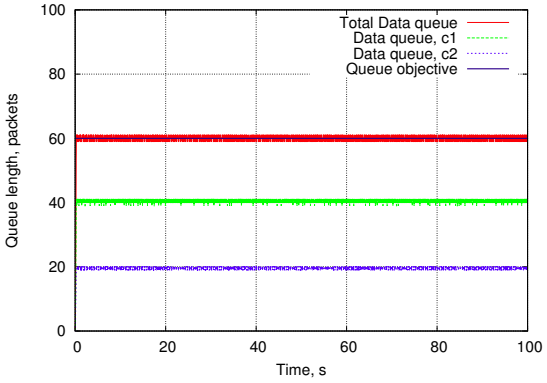
(a) Topology



(b) Chunk queue length under shaping

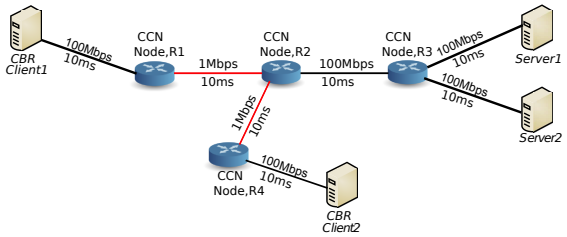


(c) Chunk queue length under shaping, different client rates, modification for traffic split

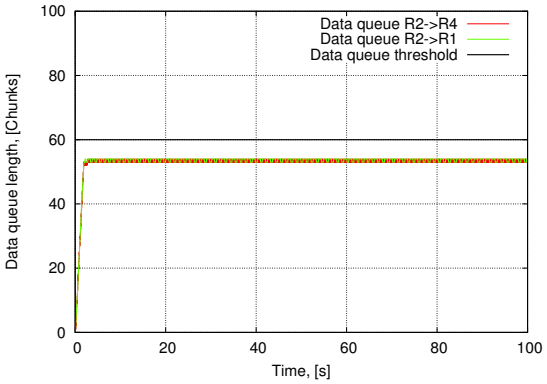


(d) Chunk queue length under shaping, different client rates, modification for traffic split

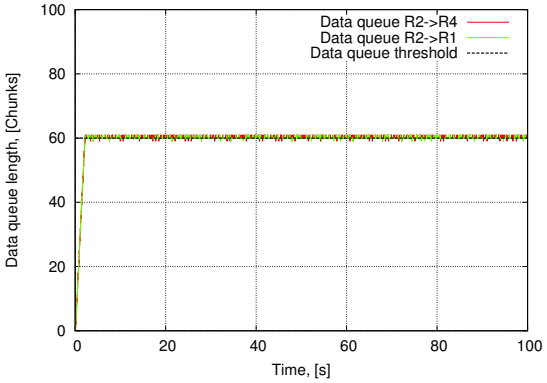
Figure 5.9: HoBHIS and traffic split



(a) Topology



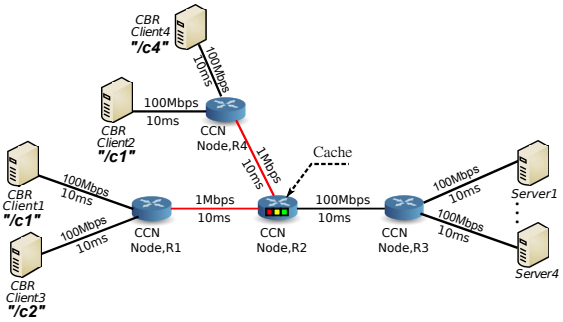
(b) Chunk queue length under shaping without virtual queues



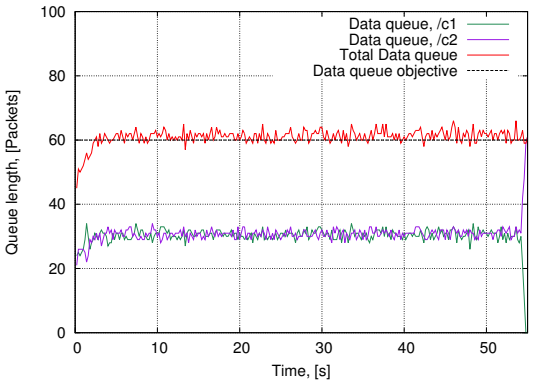
(c) Chunk queue length under shaping using virtual queues

Figure 5.10: HoBHIS and cross-traffic

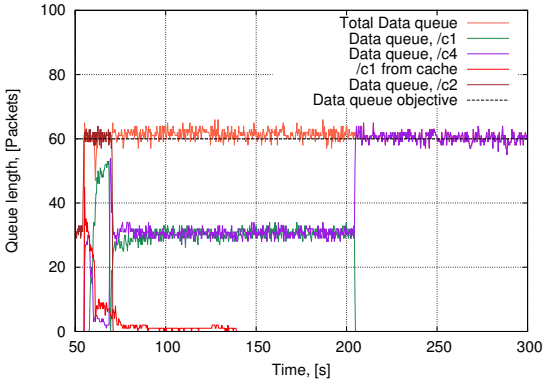
and 3 are fairly sharing the router resources as expected by HobHIS. The objective of the time period [0;55s] is to cumulate some packets in the Content Store of router $R2$. The Figure 5.11(c) shows the results for the time period [55; 300s] where the Clients 2, 3 and 4 are active. We observe that the data queue of Client 3 is oscillating around the Data queue objective. This is because the Client 3 is not sharing the same face with Clients 2 and 4. The content of Client 2 is retrieved from both cache and server. As the content from cache is served faster, it uses the part of router resources that is higher than one for the Client 4. When there is no more Chunks to retrieve from Content Store, both of Clients are sharing the same amount of resources and are converging to the same threshold. When Client 2 becomes inactive, Client 4 occupies the rest of the resources and its queue length is converging to the Data queue threshold as expected by HoBHIS. Note that during the entire simulation, the total queue length is maintained around the objective.



(a) Topology



(b) Chunk queue lengths from 0 to 55s



(c) Chunk queue lengths from 55 to 300s

Figure 5.11: Influence of caching on HoBHIS

5.6 Conclusion

In this Chapter we studied some complex scenarios for HoBHIS and proposed efficient solutions to manage them. We first analysed three important cases that can occur in CCN: traffic-split, cross-traffic and influence of caching on the performance of congestion control mechanisms.

The traffic split is a situation where the Interests arriving to the same input face inF are splitted between different output faces. The problem with this case is that the Chunks returning from different paths will be accumulated on the same output face (that is the incoming face for Interests, inF). The cross-traffic is a dual situation to traffic-split. In this case the Interests arriving to different faces are then mixed into the same output face of a CCN router. Finally, the expensive caching capabilities of CCN may influence the congestion control mechanisms because the rate of Chunks taken from cache of a CCN router is not limited and may quickly saturate the output buffer of this router.

We then analyzed the multicast scenario where the same content is retrieved by multiple clients and studied different scenarios evolving from this case. A solution to manage this situation has been proposed and is based on using the minimum of the link capacities and the maximum queue length at time t.

Finally, we have proposed a unified solution that allows to use the shaping rate formula that adopts dynamically to all complex situations. The performance evaluation of the proposed solutions has been evaluated through various simulation scenarios using our implementation in ndnSIM. The results obtained show that the shaping mechanism performs as designed and is able to dynamically manage complex cases.

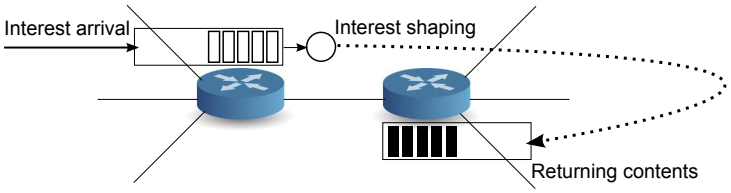
Chapter 6

Proactive ICN Congestion Control

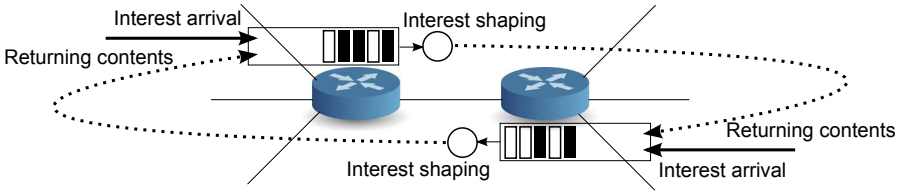
The communication paradigm of NDN [66] has two prominent features: 1) all traffic is receiver-driven; 2) content retrieved in response to an interest traverses exactly the same links in reverse order. These two unique characteristics make hop-by-hop interest shaping a better option for NDN congestion control than traditional TCP-like mechanisms. TCP congestion control *reacts* to congestion after data packets are lost. By contrast, interest shaping *proactively* prevents data packet loss by regulating the interest rate in the first place. Dropping interest packets early wastes fewer resources than dropping data packets late. More significantly, end-to-end congestion control is severely handicapped in NDN. Extensive content multihoming and caching make it very difficult to identify interests belonging to a single "flow"and sharing the same congestion path. Performing interest shaping in a hop-by-hop manner can significantly alleviate this problem, especially if the shaping scheme does not rely on flow identification. It also enables sophisticated forwarding strategies such as congestion-aware rerouting to higher-cost (but uncongested) paths. Hence, incorporating hop-by-hop interest shaping with a backpressure mechanism appears to be a more viable option for NDN congestion control.

A number of schemes have been proposed along this path (e.g., [52, 46, 17]) but all these interest shapers consider single or multiple unidirectional flows. As Figure 6.1(a) shows, if we assume that interests are sent in just one direction or the bandwidth consumption of the interests in the reverse direction is negligible, then the design of the interest shaping algorithm is quite straightforward: we simply pace the interests so that the contents they bring back will saturate the reverse link but not overload it. However, in practice, a link will see interests and contents flowing in both directions simultaneously. This is especially true for NDN since extensive in-network caching obscures the difference between clients and servers. Also content names in interests can be long since it includes transactional information in many applications (e.g., [33]). As each interest packet fetches exactly one data packet, interests may consume a non-negligible fraction of the link bandwidth. In light of these factors, the interest shaping algorithm in one direction can no longer assume that the entire reverse link bandwidth is available for returning contents. It needs to shape the interests properly so that enough room is left on the reverse link for interests in the reverse direction. The same logic applies to the interest shaper on the other side of a link and they have a recursive interdependence as shown in Figure 6.1(b). Since it is non-trivial to determine how much room should be left by the shaper, an imprudent algorithm may cause starvation or link under-utilization.

In this chapter, we present our joint work with Cisco Systems and North Carolina State University (NCSU). We examine the interdependence between interests and contents in bidirectional flows, and study its impact on the design of hop-by-hop interest shapers in



(a) Simplified scenario



(b) Practical scenario

Figure 6.1: Interdependence between interests and contents in reverse directions and its impact on hop-by-hop interest shaping

NDN. Section 6.1 formulates this issue as an optimization problem and obtains optimal solutions under various scenarios. Section 6.2 presents a practical shaping algorithm based on these solutions, to be executed at the output interface of NDN routers. Section 6.3 evaluates the performance of our algorithm in conjunction with simple AIMD clients via simulation. Finally, Section 6.4 concludes the paper.

6.1 Problem Formulation

As depicted in Figure 6.2, let i_1 and i_2 denote the interest arrival rate in each direction. They are shaped down to s_1 and s_2 respectively by the interest shapers at the routers. Let r_1 and r_2 denote the average size ratio between contents and interests in each direction and let c_1 and c_2 denote the link capacity in each direction. Note that both r_1 and r_2 must be greater than 1. We can then formulate the interest shaping problem under steady state as follows:

Objective:

$$\max \; u(s_1) + u(s_2) \tag{6.1}$$

Subject to:

$$0 \leq s_1 \leq i_1 \tag{6.2}$$

$$0 \leq s_2 \leq i_2 \tag{6.3}$$

$$s_1 + r_2 s_2 \leq c_1 \tag{6.4}$$

$$r_1 s_1 + s_2 \leq c_2 \tag{6.5}$$

where $u(\cdot)$ is the utility function discussed later. The objective of this optimization is to maximize network utility (Eq. 6.1) subject to demand (Eq. 6.2 and 6.3) and bandwidth (Eq. 6.4 and 6.5) constraints. As Figure 6.3 shows, the feasible region of this optimization

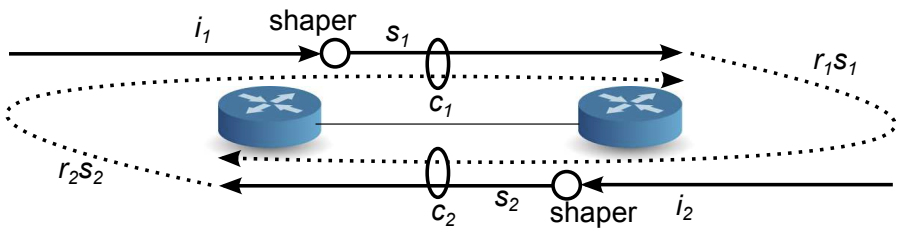


Figure 6.2: Modeling hop-by-hop interest shaping

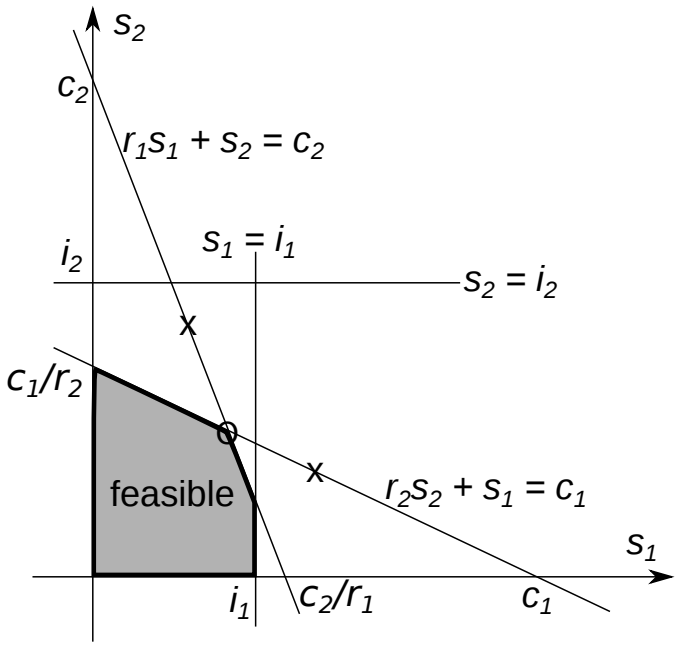


Figure 6.3: The feasible region of the optimization problem is convex.

problem is convex. Hence, if the utility function constitutes a concave objective function, the formulation is a convex optimization problem which is mathematically tractable.

Note that the utility function we pick has profound impacts on the solution we would obtain. The utility function must maximize data throughput while maintaining certain degree of fairness between the traffic in both directions. As mentioned above, reverse interests compete with forward data packets for bandwidth and the shaper needs to appropriately divide resources between them. It has been shown in [?] that logarithmic utility functions can achieve proportional fairness between competing flows. Hence, we present below a closed-form solution to the described optimization problem under logarithmic utility functions (equivalent to maximizing the product of $s1$ and $s2$).

First, if we temporarily assume infinite loads in both directions, we can lift the constraints in Eq. 6.2 and 6.3. Second, from Figure 6.3 the optimal solution of the problem lies on the boundary of the feasible region, so we can convert the inequality constraints in Eq. 6.4 and 6.5 into equality constraints. Now let us begin by solving the optimization problem under each equality constraint independently. Solving for Eq. 6.4:

$$\begin{array}{ll} \text{maximize} & s_1 s_2 \\ \text{subject to} & s_1 + r_2 s_2 = c_1 \end{array}$$

The optimal solution for this problem can be determined using Lagrange multipliers:

$$\begin{cases} s_1 = \frac{c_1}{2} \\ s_2 = \frac{c_1}{2r_2} \end{cases} \tag{6.6}$$

This optimal point has been labeled by an "x"in Figure 6.3. Similarly, solving for Eq. 6.5:

$$\begin{array}{ll} \text{maximize} & s_1 s_2 \\ \text{subject to} & r_1 s_1 + s_2 = c_2 \end{array}$$

The solution (also labeled by an "x" in Figure 6.3) is:

$$\begin{cases} s_1 = \frac{c_2}{2r_1} \\ s_2 = \frac{c_2}{2} \end{cases} \tag{6.7}$$

Now we consider the solution to the original problem under infinite demand (i.e., the optimization problem consisting Eq. 6.1, 6.4 and 6.5). From Figure 6.3 we can see that the optimal solution primarily depends on how the two bandwidth constraint lines cross each other. If one of the optimal points is within the feasible region, then the optimal solution is just that point (Eq. 6.6 or 6.7). If neither optimal point lies within the feasible region (the case shown in Figure 6.3), the optimal solution is given by the intersection of the two lines (labeled by a circle):

$$\begin{cases} s_1 = \frac{r_2 c_2 - c_1}{r_1 r_2 - 1} \\ s_2 = \frac{r_1 c_1 - c_2}{r_1 r_2 - 1} \end{cases} \tag{6.8}$$

It is trivial to show that the only case where both optimal points lie within the feasible region occurs for the degenerate case of $r_1 = r_2 = 1$, in which case both points coincide and represent the optimal solution.

Now we reintroduce the condition of finite load (Eq. 6.2 and 6.3). In practice, it is possible that the traffic load in one direction is inherently low (e.g., the first hop link from a client may have lots of outgoing interests but few incoming interests). Under such cases, the reduced load in one direction should result in increased shaping rate in the opposite direction. The limiting case is unidirectional demand in which case:

$$i_2 = 0 \tag{6.9}$$

$$s_1 = \frac{c_2}{r_1} \tag{6.10}$$

Hence, the instantaneous shaping rate should be variable. We seek proportional fairness between the two-way traffic only if both directions have excessive demand and are competing for the link capacity in both directions. If the load in one direction is inherently low, our scheme is work-conserving and will let the traffic in the other direction grab as much bandwidth as it can. An adaptive algorithm that adjusts the shaping rate between Table 6.1 and Eq. 6.10 based on the fluctuating demand will be presented in the next section.

case	s_1	s_2
$\frac{c_1}{c_2} < \frac{2r_2}{r_1r_2+1}$	$\frac{c_1}{2}$	$\frac{c_1}{2r_2}$
$\frac{2r_2}{r_1r_2+1} \leq \frac{c_1}{c_2} \leq \frac{r_1r_2+1}{2r_1}$	$\frac{r_2c_2-c_1}{r_1r_2-1}$	$\frac{r_1c_1-c_2}{r_1r_2-1}$
$\frac{c_1}{c_2} > \frac{r_1r_2+1}{2r_1}$	$\frac{c_2}{2r_1}$	$\frac{c_2}{2}$

Table 6.1: Optimal solution under infinite loads in both directions

6.2 Practical Algorithm

As we have shown, obtaining the optimal interest shaping rate is mathematically tractable if the shaper has knowledge of the content/interest size ratio (r_1 and r_2), link capacity (c_1 and c_2) and demand (i_1 and i_2) in both directions. Due to the symmetric routing of interests and contents in NDN, the shapers can independently measure r_1 and r_2 by observing the interests and contents arriving at and leaving the interface. If we assume that link bandwidths c_1 and c_2 are static parameters, they can also be made known to the shapers easily. However, the offered demand (i_1 and i_2) is constantly varying and cannot be accurately measured. Further, the shaper at one end of a link cannot know the interest load on the other side without additional message exchange. Hence, we have designed an adaptive algorithm that does not require accurate knowledge of the offered demand on both sides, which we present below.

From Eq. 6.10, we can compute the *maximum* interest shaping rate max_s_1 (which occurs when $i_2 = 0$). We can also determine the *minimum* interest shaping rate (min_s_1) from Table 6.1. To determine the actual shaping rate, we measure the incoming interest rate and use it as an estimation of the load on the other side. Assuming a similar shaper is running on the other side of the link, if there is sufficient demand in the reverse direction, then the observed incoming interest rate (obs_s_2) should be no less than the rate given by the s_2 column in Table 6.1 (we call it the expected minimum incoming interest $expmin\_s_2$). Hence, if $obs\_s_2 \geq expmin\_s_2$, the shaping rate is set to min_s_1. Otherwise, the shaping rate is calculated as follows:

$$min\_s_1 + (max\_s_1 - min\_s_1)(1 - \frac{obs\_s_2}{expmin\_s_2})^2 \tag{6.11}$$

This equation adjusts the outgoing interest shaping rate between min_s_1 and max_s_1. If $obs\_s_2 = 0$, the shaping rate become max_s_1. As we measure higher incoming interest rate, we reduce the shaped outgoing interest rate until it hits min_s_1. We observe that the quadratic control used here is more conservative and robust than a linear control.

Figure 6.4 shows how the proposed interest shaper is implemented on each interface of an NDN router. The outgoing packets are first classified into interests and contents. Content packets are passed directly to the link output queue without shaping. Interest packets join a separate shaper queue, the output of which is fed into the link output queue. The shaping rate of this queue is dynamically computed as per Eq. 6.11.

It is important to note that hop-by-hop interest shaping alone is inadequate to solve the entire congestion control problem. If a client issues more interests than the network can handle, the excessive interests are discarded by the shaper. This interest loss must be signaled back to the client so that it can slow down its request rate. In this paper, we use the simplest drop-tail policy for the interest queue in the shaper and reject interests with negative acknowledgments (NACK). A similar NACK mechanism has been proposed in [64]. Compared with explicit congestion notification in the current Internet, we believe that using NACK to signal congestion in NDN networks with hop-by-hop interest shaping

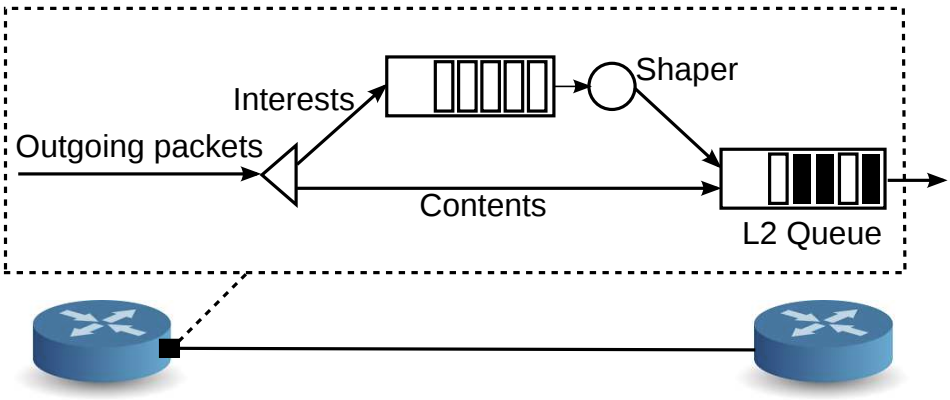


Figure 6.4: Shaper implementation

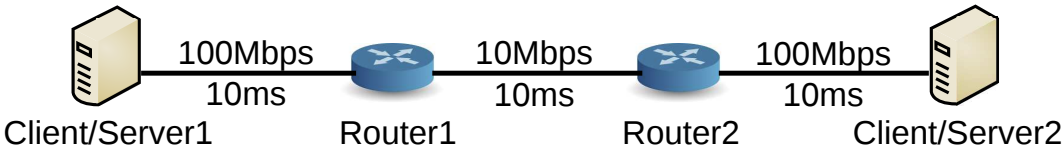


Figure 6.5: Baseline topology

has some unique advantages. When a neighboring router forwards an interest across a link, enough bandwidth has been accounted for in the reverse direction of this link to accommodate the expected returning content. If this interest is then rejected with a NACK towards the client, this NACK takes the place of the accounted-for content. As long as NACKs are smaller on average than contents, there should be enough bandwidth in all the downstream links for NACKs and they should never get lost due to congestion. Hence, using our shaping scheme, NACKs are a much more reliable and timely method of congestion notification than existing proposals (e.g., the timeout mechanism in [16]).

Clients should react to congestion-triggered NACKs and throttle their outstanding interest rate. Intermediate routers can also react to this congestion signal and implement some sophisticated forwarding strategies such as dynamic rerouting to alternative paths, or mid-stream throttling of flows. In this paper, we demonstrate our shaping algorithm with a simple window-based control on clients only. We defer the design of more sophisticated router reaction to our future work.

6.3 Performance Evaluation

We have implemented our proposal in ndnSIM and evaluated its performance across a number of different scenarios. In our current implementation, each shaper estimates r_1 and r_2 by monitoring the sizes of the interests and contents passing in both directions. The average size of interests (or contents) is calculated by smoothing out the observed samples similar to TCP round-trip time (RTT) estimation. The use of an exponentially weighted moving average (EWMA) allows errors in size measurement to correct themselves. Further, the algorithm only relies on a reasonable estimate of the average interest and content sizes, so individual flows can have wildly different ratios without impacting our algorithm. We also allow for 2% headroom in the shaping to accommodate traffic

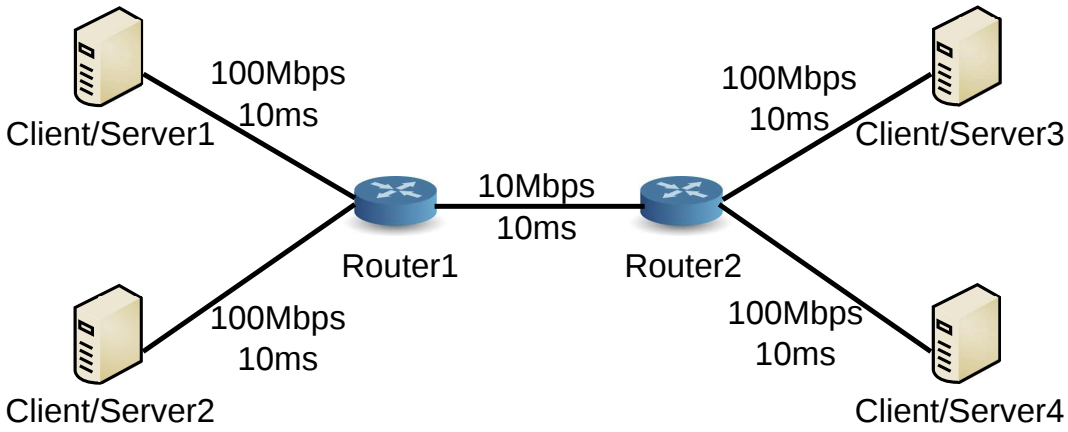


Figure 6.6: Dumbbell topology

burstiness caused by heterogeneous packet size or RTT. The sizes of the shaper queues as well as the Layer 2 queues are all set to 60 packets. In-network caching is not enabled at this point. All simulations last for 70s and the session start times are randomly picked between 0s and 5s. Metric measurement ignores the first 10s to eliminate any transient effect during the warm-up phase. All test cases have been repeated 12 times and 95% confidence intervals are calculated.

In the baseline scenario shown in Figure 6.5, clients at the two ends of the network issue interests for the contents served at the other end. Content payload size is fixed at 1000B and the interest size is 24B (which slowly increases to 28B due to the increasing number of digits in the content name: $/prefix/1$, $/prefix/2$, ...). Table 6.2 shows the simulation results under this scenario as well as a few other scenario that will be discussed later. As can be seen from the table, the hop-by-hop interest shaper effectively controls congestion so that the packet loss rate at the bottleneck link (due to queue overflow) is zero. About 0.015% of interests issued by the clients are rejected by the shapers with NACK and they serve as congestion signals propagating back to the clients. The throughputs achieved by the clients on both sides of the bottleneck are close to 9.56Mbps. This illustrates the bandwidth consumed by interests and the motivation for our optimization: to achieve 9.56Mbps of data throughput in each direction, we need an additional $9.56 \times 26 \div 1000 \approx 0.24856$Mbps of interest rate in each direction. Therefore, the total throughput of interest and content traffic is 9.80856Mbps. Taking into account the 2% headroom reserved by the shaper, our shaping algorithm achieves the optimal possible throughput.

Let us now investigate a few variations of the baseline scenario. First, we evaluate the shaper under randomized packet size. Recall that the shaper estimates r_1 and r_2 by smoothing packet size samples. When the sizes vary over time, the shaper may have inaccurate estimation of r_1 and r_2, leading to sub-optimal shaping behavior. However, as shown in Table 6.2, when the interest packet size is uniformly distributed between 27B and 62B, and the content payload size between 600B and 1400B, the shaper still achieves zero packet loss and we only experience minor throughput loss. Second, we evaluate the scenario where the content/interest size ratios in two directions are asymmetric (i.e., $r_1 \neq r_2$). In this scenario, the content payload size on Client/Server2 is set to 500B while the other side still has 1000B payloads. Due to the reduced payload size, Client/Server1 needs to send interests at twice the original rate to achieve the same data throughput. Hence, we experience lower data throughput at Client/Server2 because more link bandwidth is allocated to the interests from Client/Server1. Finally, we simulate the asymmetric link

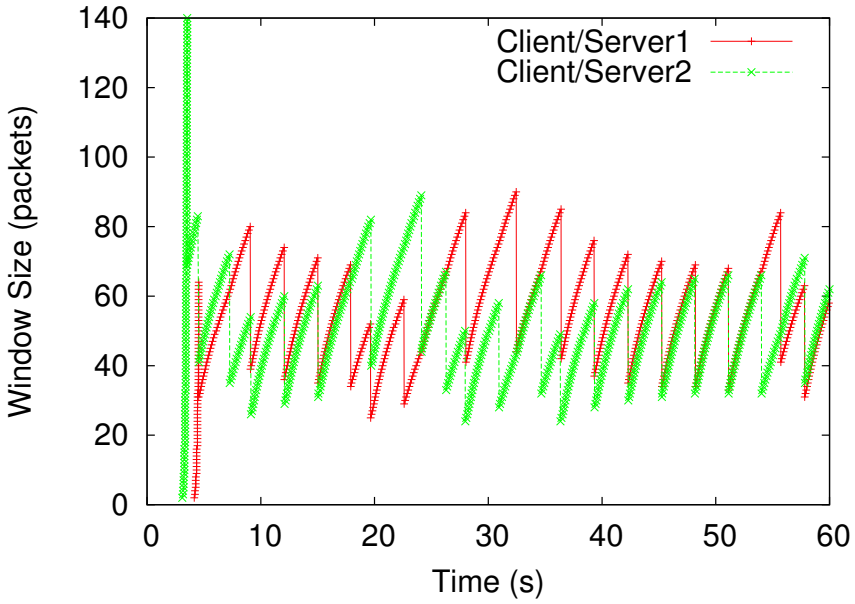


Figure 6.7: Client window size evolution under homogeneous RTT

bandwidth scenario (i.e., $c_1 \neq c_2$), as is commonly found in today's residential access networks. Here the bottleneck link bandwidth from Router1 to Router2 is reduced to 1Mbps while the reverse link remains at 10Mbps. As simulation results show, our hop-by-hop interest shaper still manages to control congestion under this highly asymmetric scenario. Client/Server2 achieves approximately 0.72Mbps data throughput out of the 1Mbps link, leaving the rest to the interests from Client/Server1 that bring back 9.77Mbps of contents. This also outlines the importance of the logarithmic utility function which avoids starvation of either side.

Next, we evaluate our scheme under the dumbbell topology shown in Figure 6.6. Three different scenarios have been simulated. First, we launch two clients on the left side of the bottleneck link. They retrieve contents from the two servers on the right side respectively. Here the two flows have homogeneous RTTs of 60ms and their performance is shown in Table 6.3 along with other scenarios under the same topology. As Figure 6.7 shows, the two flows roughly converge to fair bandwidth sharing despite the randomized session start time. Note that, in this scenario, there are no interests coming from the right side of the bottleneck link, and therefore all this bandwidth can be used for the returning contents. Our shaping algorithm has correctly adapted to this situation, as evidenced by the sum of the throughputs of these two flows being around 9.8Mbps.

We reran this test but set the link latency between Router2 and Client/Server4 to 20ms so that the two flows now have heterogeneous RTTs. The results are also shown in Table 6.3. Figure 6.8 compares the evolution of the bottleneck queue length over time between the homogeneous RTT case and the heterogeneous RTT case. As the figures show, we have minimum queues (oscillating between zero and one packet) under homogeneous RTTs since the interests going out from Router1 to Router2 are perfectly paced by the shaper. With heterogeneous RTTs, we see more fluctuations in the queue length due to the burstiness of the traffic incurred by RTT heterogeneity. However, even in this case the queue is well controlled (maximum of 17 packets). Note that our shaper is an open-loop controller that aims at long-term fairness and stability of the system.

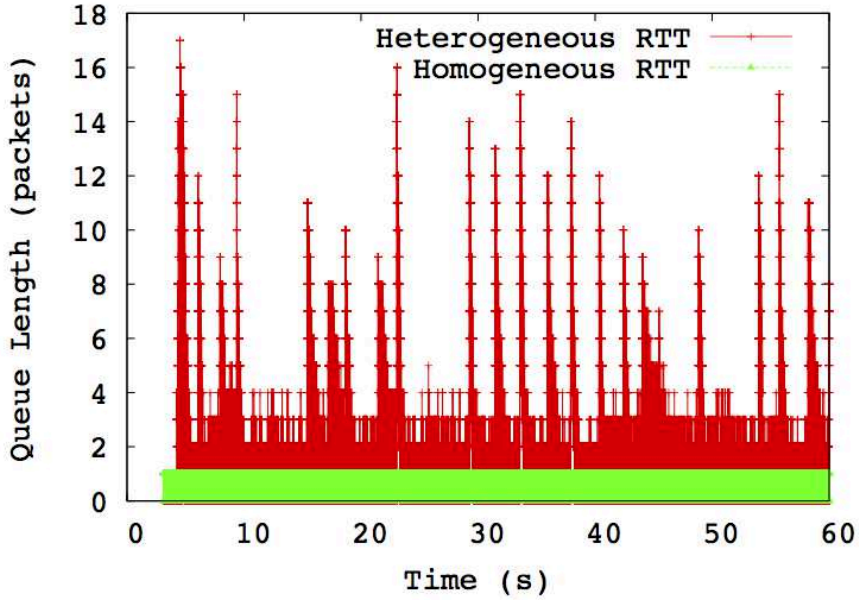


Figure 6.8: Comparison of bottleneck queues

It accounts for the steady-state average rates and leaves their variance to be handled by the headroom parameter and the Layer 2 queue buffers. We believe that with proper parameter tuning, our open-loop controller is able to absorb the traffic burstiness and make the system converge to the optimal steady state, as demonstrated by our simulations. An alternative is a closed-loop controller that adjusts the shaping rate based on the instantaneous queue length. However, such a solution may be much more complicated and costly since the queue states must be communicated between the two shapers of each link via some messaging or piggybacking technique.

Finally, we simulate the following heterogeneous RTT scenario: one client resides on Client/Server1 and retrieves contents from Client/Server3. Another client resides on Client/Server4 and retrieves contents from Client/Server2. Since the link latency between Router2 and Client/Server4 is 20ms, these two flows sharing the bottleneck link in reverse directions have different RTTs. The results in Table 6.3 show that our hop-by-hop interest shaper works perfectly in this scenario, too.

In summary, our proposed shaping algorithm has effectively controlled data congestion, kept data queue sizes low, and achieved near-optimal data throughput with zero packet loss across all the test cases we have simulated.

6.4 Conclusion

In conclusion, we have presented a hop-by-hop interest shaping algorithm for NDN to avoid network congestion and achieve optimal network resource utilization. Our proposed solution accounts for the interdependence between interests and contents in opposite directions and is capable of optimally sharing link bandwidth without extra message exchange. We evaluated the performance of our mechanism when combined with simple AIMD clients using ndnSIM, and studied its behavior under varying conditions of bandwidth, load, and

packet size ratios. Our future work includes a more comprehensive evaluation of our interest shaper, the design of more sophisticated backpressure mechanisms as well as congestion-aware multipath routing.

Scenario	Data Throughput (Mbps)		Packet Loss Rate (%)		Interest Rejection Rate (%)	
	Client/Server1	Client/Server2	Router1	Router2	Client/Server1	Client/Server2
Baseline	9.558421 ± 0.001261	9.559624 ± 0.001550	0	0	0.0150 ± 0.0006	0.0153 ± 0.0011
Randomized Packet Size	9.432117 ± 0.005931	9.434337 ± 0.007859	0	0	0.0180 ± 0.0014	0.0167 ± 0.0015
Asymmetric Size Ratio	9.373692 ± 0.014214	9.326215 ± 0.000921	0	0	0.0074 ± 0.0006	0.0155 ± 0.0006
Asymmetric Link Bandwidth	9.774441 ± 0.001723	0.719525 ± 0.000139	0	0	0.0119 ± 0.0005	0.0576 ± 0.0000

Table 6.2: Simulation results over baseline topology

Scenario	Data Throughput (Mbps)		Packet Loss Rate (%)		Interest Rejection Rate (%)	
	Client/Server1	Client/Server2	Router1	Router2	Client/Server1	Client/Server2
Homogeneous RTT	5.142089 ± 0.505369	4.692407 ± 0.505271	0	0	0.0515 ± 0.0112	0.0620 ± 0.0129
Heterogeneous RTT	5.209043 ± 0.384781	4.624094 ± 0.380328	0	0	0.0513 ± 0.0092	0.0428 ± 0.0067
	Client/Server1	Client/Server4	Router1	Router2	Client/Server1	Client/Server4
Heterogeneous RTT	9.565575 ± 0.000762	9.419777 ± 0.007525	0	0	0.0148 ± 0.0004	0.0116 ± 0.0005

Table 6.3: Simulation results over dumbbell topology

Chapter 7

Conclusion and Future work

The evolution of the Internet has triggered a significant activity exploring new architectures among which the concept of Information-Centric-Networks (ICN) has emerged. Considering the various ICN solutions, Content-Centric Networking (CCN) is the one that received most attention. The design of CCN is progressing albeit many important issues still deserve a careful analysis and design.

This architecture differs from the traditional host-based communication principle in many ways. One important change is that network addresses are replaced by content names to distribute information in CCN networks. Moreover, extensive in-network caching capabilities are introduced to benefit from the observation of the traffic flowing in the network. As a consequence, data packets can be delivered by any CCN router in addition to the source, according to various caching algorithms.

All these architectural features demand congestion control that would be completely different from that in TCP/IP. We proposed a first comprehensive solution for congestion control in CCN networks. This problem is of utmost importance and has not been addressed in this context.

7.1 Contribution 1: Hop-by-hop Interest shaping mechanism

In Chapter 3 we presented HoBHIS, the first hop-by-hop Interest shaping congestion control mechanism designed to avoid the congestion that can occur in the output interface of a CCN node. It nicely exploits the flow balance enforced in CCN between Interest and Chunk packets. This mechanism is implemented in each CCN node and mostly consists in monitoring active conversations sharing the transmission buffer of a CCN node face in order to dynamically adjust their Interest sending rate and enforce the Chunk queue length to converge to a defined objective.

7.2 Contribution 2: Tolerance mechanism

As a consequence of the CCN design principles, we have to regulate the stream of chunks as well as the stream of interests in order to avoid congestion and to improve network performance. The important concern in CCN is related to the ability offered to a user to send interests without any limiting rate factor, creating a risk for buffer overflow and performance degradation. In Chapter 4 we introduced a new rate-based mechanism aiming

at controlling the interest sending rate of a content receiver (a client or source of interest). The scheme introduces an exchange of control information between CCN nodes and the client. CCN nodes periodically send control packets with an explicit rate specifying the maximum sending rate for each Client on the path used by this conversation. In order to derive an optimum explicit rate, CCN routers use a control function based on the shaping rate computed by HoBHIS.

7.3 Contribution 3: Important extensions to HoBHIS

In Chapter 5 we first studied more complex scenarios and proposed efficient solutions to manage three important cases that can appear in CCN: traffic-split, cross-traffic and influence of caching on the performance of congestion control mechanisms.

The traffic split is a situation where the Interests arriving to the same input face inF are splitted between different output faces. The problem with this case is that the Chunks returning from different paths will be accumulated on the same output face (that is the incoming face for Interests, inF). This is not trivial to estimate the shaping rate for this case. The cross-traffic is a dual situation to traffic-split. In this case the Interests arriving to different faces are then mixed into the same output face of a CCN router. Finally, the expensive caching capabilities of CCN may influence the congestion control mechanisms because the rate of Chunks taken from the cache of a CCN router is not limited and may quickly saturate the output buffer of this router.

We then analyzed the multicast scenario where the same content is retrieved by multiple clients and studied different scenarios evolving from this case. A possible solution to manage this situation has been proposed and is based on using the minimum of the link capacities and the maximum queue length at time t.

Finally, we proposed a unified solution that allows to use our shaping rate formula that adapts dynamically to all complex situations.

7.4 Performance analysis

In this dissertation we first demonstrated analytically the convergence property of our algorithms. We then performed various experiments with different settings and progressive complexity using our implementation of the mechanisms in different network simulators. According to our knowledge, at the beginning of this work there was no version of CCN in ns2. We started the implementation of our solutions to ns2. Thus, the first performance evaluation of the mechanisms has been done by using this simulation environment. Later, we ported our implementation to ns_3-based Named Data Networking Simulator (ndnSIM) that implements Named Data Networking (NDN) communication model. Next simulation experiments has been performed with this environment. The source code of our implementations is publicly available at *www.github.com*.

We analyzed the single and multiple conversation scenarios in a single router model as well as for a more complex network case. Therefore, we have demonstrated the behaviour of our algorithm in more complex conditions. We have seen that the shaping mechanism performs as designed.

We observed that the results fully satisfy the design objectives and we can conclude that HoBHIS is an efficient and operational solution to the problem of congestion control

in CCN.

7.5 Future work

Our work can be extended in various directions. In this dissertation, we have demonstrated the promising results and effectiveness of our solutions. Hovewer, it would certainly be interesting to consider different types of traffic and services and analyze if differentiating between them will provide additional value.

Also, an effective admission control would be desired to limit misbehaving clients keeping the network utilization below the saturation point, while still optimally utilizing the network resources.

Another interesting point may concern the multipath extension for the proposed schemes. Instead of limiting the Client sending rate, the Interest packets might be redirected to other faces of the CCN router in order to achieve better utilization of network resources. One proposal of such an adaptive forwarding has been presented in [65, 64]. Hovewer, the need and effectiveness of an Interest reforwarding should be verified in case of our congestion control solutions.

Study of bidirectional traffic and sharing of the resources between Interests and Chunks in the same direction may also be an interesting point to follow this work.

Finally, it would be interesting to study the specific application case of our mechanisms, for example in mobile, vehicular etc. networks.

Appendix A

HoBHIS implementation in ns3-based ndnSIM

In this section we briefly describe the implementation of our solutions to ndnSIM.

A.1 What is ndnSIM?

ndnSIM, [3], is a simulator that implements Named Data Networking (NDN) communication model and represents an ns_3 module. This simulator is optimized for simulation purposes and easy to use, modify and extend. Even if this implementation is relatively recent, it becomes very popular in the research community. We use this simulator for implementing our congestion control schemes presented in this dissertation.

A.2 Design goals

The implementation of our contributions has the goal to be an open source package enabling the researchers to run the experiments in order to verifiy the results presented in our papers, compare their proposals with ours, propose and add some new features.

A.3 Design overview

In this section we briefly present the design of our implementations. We explain what abstractions of ndnSIM and ns_3 have been used as a base. Finally, we present added objects and features.

A.3.1 HoBHIS

We started by implementing our first solution: Hop-by-Hop Interest Shaping mechanism (HoBHIS). We first have to add Interest shaping functionnality to the ndn::Face abstraction that defines basic functionality of Ndn face. We have implemented the HobhisNetDeviceFace derived from ndn::NetDeviceFace which is permanently associated with one ns_3 NetDevice object and this object can not be changed for the lifetime of the face. HobhisNetDeviceFace adds shaping fonctionnalities, computes the shaping rate, schedules

the Interests according to HoBHIS algorithm and provide some per-flow statistics about shaped Interests.

At the time of writing the congestion control functionnalities was not extended in ndnSIM and it was not possible to esily obtain the NDN statistics or modify the functionnalities of the transmission queues. That is why we have also added an NDNDropTailQueue class derived from ns3::Queue. This queue is mostly for keeping Chunks because the Interests are managed by the shaping algorithm in HobhisNetDeviceFace. NDNDropTailQueue provides a simple DropTailQueue but keeps the interesting statictics such as number of flows, queue size per flow, total queue size. In our implementation we do not only need this class for statistics but also because this information is used by HoBHIS for computation of the shaping rate.

In order to test our mechanism with different traffic distributions we have added a functionnality of generating the traffic following different distribution laws to ndn::Producer. We used such random number generators provided by ns_3 as Uniform, Exponential, Pareto, Sequential etc.

A.3.2 Tolerance Rate

The second part of our implementation consists in realisation of Tolerance Rate mechanism. For that we are using the ndn::ForwardingStrategy abstraction and core implementation for Interest and Data forwarding. We have derived an IRControl class from ndn::ForwardingStrategy. This class enables sending the control packets with corresponding tolerance rate back to the clients each $A(t)$ seconds according to the algorithm. The tolerance rate is computed by HobhisNetDeviceFace and used in IRControl. On receiving the control packet with tolerance rate, the clients should adjust their Interest sending rate to this value. We added this functionnality and created a new type of Consumer.

A.4 Complex scenarios

To test the complex scenarios presented in Chapter 5 we have implemented the corresponding functionnalities to HobhisNetDeviceFace and ForwardingStrategy to test the influence of caching.

A.5 How to use

The usage of the implemented solutions is pretty easy. As the code is publicly available, the installation procedure is specified on the correspondig page of *www.github.com*. The user can find some example scripts of HoBHIS in files: hobhis-chain.cc and hobhis-fairness.cc and run them with different parameters. To see the possible configurations he/she can ask the script for help using –PrintHelp option.

One can install the implemented objects on the nodes directy in the ns_3 topology configuration script.

A.5.1 Example of using Hobhis and Tolerance Rate mechanism in a CCN router

In this section we show how to easily activate Tolerance Rate mechanism and HoBHIS in ns_3 configuration script. The Tolerance Rate mechanism could easily be enabled during the configuration of the NDN nodes:

```
//Creation of the ndnHelper
ndn::StackHelper ndnHelper;
//Installing the ForwardingStrategy
ndnHelper.SetForwardingStrategy ("ns3::ndn::fw::IRControl");
```

The next step is to enable our shaper. We implemented the special function EnableHobhis in order to simplify the installation of the shaper. The following parameters need to be specified :

1. Enable shaper (true/false, default = false)
2. Client/server (whether the node is client or server? This parameter should be enabled on router nodes only).
3. Interest buffer size (the size of shaper queue).
4. Target (the objectif r to which we want our queue to be converged).
5. Convergence speed (design parameter h).
6. Dynamic adjustment of design parameter.

Example:

```
ndnHelper.EnableHobhis (true, false, 100, 60, 0.1, false);
```

Once all necessary configurations are finished user can run his simulation script.

A.6 Summary

We presented a brief description of the implementation of our solutions to ndnSIM. Currently, the source code for HoBHIS (Chapter 3) is publicly available on:
https://github.com/Be1thaz0r/HoBHIS. The source code used for Chapters 4 and 5 is currently in final preparation for uploading and will be published soon. Based on number of emails that we have got from people working on CCN/NDN from everywhere, we can conclude that our implementation is known and used by the researchers to better undertand the functionnality of our solutions.

Appendix B

Activity

Published and submitted papers

- Natalya Rozhnova, Serge Fdida, *An effective Hop-by-Hop Interest shaping mechanism for CCN communications*, IEEE INFOCOM NOMEN workshop, 2012, Orlando, Florida
- 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*, Best paper award, In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking (ICN '13).
- Natalya Rozhnova and Serge Fdida, *A Hop-by-Hop Interest shaping mechanism for Content-Centric Networking*, poster session, LINCS workshop, 2014, Paris, France.
- Natalya Rozhnova, Serge Fdida, *An extended Hop-by-hop Interest shaping mechanism for Content-Centric Networking*, GLOBECOM 2014, Austin, Texas.

Talks and presentations

- *Congestion Control in CCN*, talk for master students in Networking, UPMC, 2012, Paris, France
- *An effective Hop-by-Hop Interest shaping mechanism for CCN communications*, LINCS, 2012, Paris, France
- *Congestion Control issues for CCN*, North-Eastern University, 2013, Boston, USA
- Presentation of a poster, Y. Wang, A. Narayanan, D. Oran, and I. Rhee "Hop-by-hop Interest Shaping under Two-way Traffic", IEEE INFOCOM NOMEN workshop 2013, Turin, Italy
- Managing the Information-Centric Networking reading group, LINCS, Paris, France

Collaboration and internship

- ANR CONNECT project, France

- Laboratory of Information, Networking and Communication Sciences (LINCS), Paris, France
- *Congestion control for Named-Data Networking*, internship, Cisco Systems, Nov. 2012 - May 2013, Boxborough, MA, USA. Supervisor: Dave Oran
- Collaboration with Dr. Edmund Yeh, North-Eastern University, Boston, MA, USA

Teaching

- ***Simulation, Emulation and Virtualization***, Université Pierre et Marie Curie (UPMC), Paris, France.
  Master students.Total responsibility : development of the subjects of TPs, implementing of necessary tools and programs, teaching, correction of the student reports (the score for the TP represents 40% of students' final grade). Practical Classes (TP).
- ***Routing in Networks***, Université Pierre et Marie Curie (UPMC), Paris, France.
  Master students. Tutorial classes (TD) on Addressing, Forwarding, Ethernet, RIP, OSPF, BGP, Multicast, Peer-to-peer networks.
- ***Project on Networking***, Université Pierre et Marie Curie (UPMC), Paris, France.
  Master students. Proposing the subjects and plans of projects, and supervising of two groups of students working on different projects (3-4 students per group).
- ***Networking***, Université Pierre et Marie Curie (UPMC), Paris, France.
  3 year Licence students. Signal processing: Signal digitization, Modulation, Coding, Error control

Bibliography

[1] *CCNx project.* http://www.ccnx.org.

[2] *NDNx project.* http://named-data.net.

[3] *NS-3 based Named Data Networking simulator.* http://ndnsim.net.

[4] *Publish-Subscribe Internet Routing Paradigm (PSIRP) project.* http://www.psirp.org.

[5] *Scalable and Adaptive Internet Solutions (SAIL).* http://www.sail-project.eu.

[6] *The FP7 4WARD Project.* http://www.4ward-project.eu.

[7] *Fibonacci numbers and their applications*, in Proceeding of the first International Conference Numbers and their Applications, University of Patras, 1984.

[8] *NS3 DCE CCNx Quick Start.* http://www.nsnam.org/overview/projects/direct-code-execution/, 2013.

[9] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, *Design considerations for a network of information*, in Proceedings of the 2008 ACM CoNEXT Conference, CoNEXT '08, New York, NY, USA, 2008, ACM, pp. 66:1–66:6.

[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, *A survey of information-centric networking*, Communications Magazine, IEEE, 50 (2012), pp. 26–36.

[11] S. Arianfar, P. Nikander, and J. Ott, *On content-centric router design and implications*, in Proceedings of the Re-Architecting the Internet Workshop, ReARCH '10, New York, NY, USA, 2010, ACM, pp. 5:1–5:6.

[12] S. Arianfar, J. Ott, L. Eggert, P. Nikander, and W. Wong, *A transport protocol for content-centric networks*, in 18th IEEE International Conference on Network Protocols (ICNP'10), Kyoto, Japan, 2010.

[13] S. Arianfar, P. Sarolahti, and J. Ott, *Deadline-based resource management for information-centric networks*, in Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking, ICN '13, New York, NY, USA, 2013, ACM, pp. 49–54.

[14] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, *An Empirical Study of Receiver-based AIMD Flow-Control Strategies for CCN*, in IEEE ICCCN'13, July 2013.

[15] ——, *CCN and TCP co-existence in the Future Internet: Should CCN be compatible to TCP?*, in The Fifth IFIP/IEEE International Workshop on Management of the Future Internet, May 2013.

[16] G. Carofiglio, M. Gallo, and L. Muscariello, *ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking*, in IEEE NOMEN'12, Mar. 2012.

[17] G. Carofiglio, M. Gallo, and L. Muscariello, *Joint Hop-by-hop and Receiver-driven Interest Control Protocol for Content-Centric Networks*, in ACM ICN'12, Aug. 2012.

[18] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, *Multipath Congestion Control in Content-Centric Networks*, in IEEE NOMEN'13, Apr. 2013.

[19] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, *Modeling data transfer in content-centric networking*, in Teletraffic Congress (ITC), 2011 23rd International, 2011, pp. 111–118.

[20] G. Carofiglio, V. Gehlen, and D. Perino, *Experimental evaluation of memory management in content-centric networking*, in Communications (ICC), 2011 IEEE International Conference on, 2011, pp. 1–6.

[21] A. Carzaniga and A. L. Wolf, *Forwarding in a content-based network*, in Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, New York, NY, USA, 2003, ACM, pp. 163–174.

[22] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, *Wave: Popularity-based and collaborative in-network caching for content-oriented networks*, in Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on, March 2012, pp. 316–321.

[23] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, *A survey on content-oriented networking for efficient content delivery*, Communications Magazine, IEEE, 49 (2011), pp. 121–127.

[24] V. J. D. Smetters, *Securing network content*, PARCTechReport, (2009).

[25] C. Dovrolis, D. Stiliadis, and P. Ramanathan, *Proportional differentiated services: delay differentiation and packet scheduling*, Networking, IEEE/ACM Transactions on, 10 (2002), pp. 12–26.

[26] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, *Summary cache: A scalable wide-area web cache sharing protocol*, IEEE/ACM Trans. Netw., 8 (2000), pp. 281–293.

[27] S. Fdida and M. Diallo, *The network is a database*, in Proceedings of the 4th Asian Conference on Internet Engineering, AINTEC '08, New York, NY, USA, 2008, ACM, pp. 1–6.

[28] J. Garcia-Luna-Aceves, *Name-based content routing in information centric networks using distance information*, in Proceedings of the 1st International Conference on Information-centric Networking, INC '14, New York, NY, USA, 2014, ACM, pp. 7–16.

[29] M. Gritter and D. R. Cheriton, *An architecture for content routing support in the internet*, in Proceedings of the 3rd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 3, USITS'01, Berkeley, CA, USA, 2001, USENIX Association, pp. 4–4.

[30] T. Henderson, E. Sahouria, S. McCanne, and R. Katz, *On improving the fairness of tcp congestion avoidance*, in Global Telecommunications Conference, 1998. GLOBECOM 1998. The Bridge to Global Integration. IEEE, vol. 1, 1998, pp. 539–544 vol.1.

[31] 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, 2013, ACM, pp. 15–20.

[32] V. Jacobson, *Congestion avoidance and control*, in Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM '88, New York, NY, USA, 1988, ACM, pp. 314–329.

[33] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, *VoCCN: Voice over Content-Centric Networks*, in ACM ReArch'09, Dec. 2009.

[34] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, *Networking Named Content*, in ACM CoNEXT, Dec. 2009.

[35] 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., 37 (2007), pp. 181–192.

[36] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts, *Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing*, SIGMETRICS Perform. Eval. Rev., 33 (2005), pp. 217–228.

[37] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, *Popularity-driven coordinated caching in named data networking*, in Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS '12, New York, NY, USA, 2012, ACM, pp. 15–26.

[38] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, *Adaptive web caching: Towards a new global caching architecture*, Comput. Netw. ISDN Syst., 30 (1998), pp. 2169–2177.

[39] Z. Ming, M. Xu, and D. Wang, *Age-based cooperative caching in information-centric networks*, in Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on, March 2012, pp. 268–273.

[40] P. P. Mishra and H. Kanakia, *A hop by hop rate-based congestion control scheme*, SIGCOMM Comput. Commun. Rev., 22 (1992), pp. 112–123.

[41] Y. Moret and S. Fdida, *Eraqles an efficient explicit rate algorithm for abr*, in in GLOBECOM, 1997, pp. 801–805.

[42] J. Moy, *OSPF Version 2*. IETF RFC 2328, Apr. 1998.

[43] L. Muscariello and M. Gallo, *Content Centric Networking Packet Level Simulator*. https://code.google.com/p/ccnpl-sim/, 2011.

[44] A. Narayanan and D. Oran, *Content routing using internet routing protocols: Can it scale?* In ICNRG side meeting in IETF-82, Nov. 2011.

[45] D. Oran, *OSI IS-IS Intra-domain Routing Protocol*. IETF RFC 1142, Feb. 1990.

[46] S. Oueslati, J. Roberts, and N. Sbihi, *Flow-aware Traffic Control for a Content-Centric Network*, in IEEE INFOCOM, Mar. 2012.

[47] J. Pan, S. Paul, and R. Jain, *A survey of the research on future internet architectures*, Communications Magazine, IEEE, 49 (2011), pp. 26–36.

[48] PARC, *An Overview of Routing Solutions*. http://www.ccnx.org, Mar. 2014.

[49] S. Podlipnig and L. Böszörmenyi, *A survey of web cache replacement strategies*, ACM Comput. Surv., 35 (2003), pp. 374–398.

[50] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, *Modelling and evaluation of ccn-caching trees*, in Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I, NETWORKING'11, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 78–91.

[51] D. Rossi and G. Rossini, *Cache Performance of Content Centric Networks under Multipath Routing (and More)*, tech. rep., Telecom ParisTech, 2011.

[52] N. Rozhnova and S. Fdida, *An Effective Hop-by-hop Interest Shaping Mechanism for CCN Communications*, in IEEE NOMEN'12, Mar. 2012.

[53] ——, *An extended Hop-by-Hop Interest shaping mechanism for Content-Centric Networking*, in Globecom'14, 2014.

[54] L. Saino, C. Cocora, and G. Pavlou, *CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking*, in IEEE ICC 2013 - Next-Generation Networking Symposium (ICC'13 NGN), June 2013.

[55] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, and L. Tassiulas, *Storage planning and replica assignment in content-centric publish/subscribe networks*, Comput. Netw., 55 (2011), pp. 4021–4032.

[56] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, *A trace-driven analysis of caching in content-centric networks*, in Computer Communications and Networks (ICCCN), 2012 21st International Conference on, July 2012, pp. 1–7.

[57] M. Ul Haque, K. Pawlikowski, A. Willig, and L. Bischofs, *Performance analysis of blind routing algorithms over content centric networking architecture*, in Computer and Communication Engineering (ICCCE), 2012 International Conference on, July 2012, pp. 922–927.

[58] M. Vojnovic, J.-Y. Le Boudec, and C. Boutremans, *Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times*, in INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, 2000, pp. 1303–1312 vol.3.

[59] J. Wang, *A survey of web caching schemes for the internet*, SIGCOMM Comput. Commun. Rev., 29 (1999), pp. 36–46.

[60] L. Wang, A. K. M. Mahmudul Hoque, C. Yi, A. Alyyan, and B. Zhang, *OSPFN: An OSPF Based Routing Protocol for Named Data Networking*, Tech. Rep. NDN-0003, July 2012.

[61] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, *Could in-network caching benefit information-centric networking?*, in Proceedings of the 7th Asian Internet Engineering Conference, AINTEC '11, New York, NY, USA, 2011, ACM, pp. 112–115.

[62] 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 ACM ICN'13, Aug. 2013.

[63] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, *On the role of routing in Named Data Networking*, Technical Report NDN-0016, NDN, December 2013.

[64] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, *A Case for Stateful Forwarding Plane*, Tech. Rep. NDN-0002, July 2012.

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

[66] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, kc claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, and E. Yeh, *Named Data Networking Project*, Tech. Rep. NDN-0001, Oct. 2010.