

Traffic and resource management in content-centric networks : design and evaluation

Massimo Gallo

▶ **To cite this version:**

Massimo Gallo. Traffic and resource management in content-centric networks : design and evaluation. Other. Télécom ParisTech, 2012. English. NNT : 2012ENST0067 . pastel-01002134

HAL Id: pastel-01002134
https://pastel.hal.science/pastel-01002134

Submitted on 5 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

EDITE - ED 130

TELECOM ParisTech

2012-ENST-0067

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité "Informatique et réseaux"

présentée et soutenue publiquement par

Massimo GALLO

le 23/11/2012

Traffic and Resource Management in Content-Centric Networks : Design and Evaluation

---

Gestion du Trafic et des Ressources dans les réseaux centrés sur le contenu : Design et Évaluation

Directeur de thèse: **M. Dario ROSSI, Professeur**, *Telecom ParisTech*
Co-encadrement de la thèse: **M. Luca MUSCARIELLO, Docteur**, *Orange Labs R&D*

Jury

M. Marco AJMONE-MARSAN, Professeur, *Politecnico di Torino*	Rapporteur
M. Serge FDIDA, Professeur, *Université Pierre et Marie Curie*	Examinateur
M. Holger KARL, Professeur, *Universität Paderborn*	Rapporteur
M. Daniel KOFMAN, Professeur, *Telecom ParisTech*	Examinateur
M. James ROBERTS, Docteur, *INRIA - Rocquencourt*	Examinateur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Acknowledgement / Remerciements

I am very grateful to all the people, research groups, companies that supported me for the last three years during the Ph.D. preparation. The list is very long and I hope I won't forget anyone.

First of all I want to thank Luca, my "industrial" advisor, for the opportunity of undertaking the Ph.D. he gave me three years ago. He taught me to work with passion and curiosity, two fundamental qualities for a researcher. He has always been available for me and those three years allowed me to grow both personally and professionally. I hope there will be a way to collaborate with him in the future. Thanks to Dario, my academic advisor that always encouraged me to attend the LINCS's seminars and explore other research fields.

Thanks to the manuscript's reviewers Marco Ajmone-Marsan and Holger Karl for the time they dedicated to my thesis. Thanks also to the jury's members that accepted to evaluate my work.

Thanks to Giovanna that has been a valuable co-author and a third "un-official" advisor to me. Thanks to Diego that, besides being a co-author, probably convinced me that the Ph.D. at Orange Labs would be the right choice.

The years I spent in the Traffic and Resource Management team (TRM) at Orange labs have been rich in terms of technical and personal discussions. The list of people that left/joined the team during those years is very very long so If you have been part of this team while I were there, I thank you. Thanks to Prosper, the Future Internet research responsible at Orange Labs, that has always shown interest in my work. Thanks to Amel and Jean-Robin (in alphabetical order!) that shared the office with me for almost three years. They always tolerated my complaints and my bad Italian words! Thanks to all the Ph.D mate of TRM, they all became really good friends to me.

Thanks to the people that participated to the ANR CONNECT project with which I had fruitful and interesting discussions. In case I forgot someone, I would really like to thank all the people that I met at Orange Labs, Telecom ParisTech, Alcatel-Lucent Bell Labs and LINCS laboratory.

Now, some acknowledgments in Italian:

Grazie a Francesca e Luciano, i miei genitori, che mi hanno permesso di avere l'istruzione che ho avuto e di arrivare dove sono arrivato. Grazie a Je, che è sempre stata al mio fianco portando con se il proprio ottimismo e la propria spensieratezza (che io non ho!). Grazie anche a tutte le altre persone, amici, parenti, insegnanti, professori che, in un modo o nell'altro, hanno contribuito negli anni alla mia formazione personale e professionale.

Abstract

The advent of the World Wide Web has radically changed Internet usage from host-to-host to service access and data retrieval. The majority of services used by Internet's clients (The web, Video on Demand, IP Television, on-line gaming, social networking) are content-centric in the sense that they focus on named content objects rather than on host-location. Contrarily to its usage, the original Internet revolves around host-to-host communication for which it was conceived. Even if Internet has been able to address the challenges offered by new applications, there is an evident mismatch between the architecture and its current usage. In this context, many projects in national research agencies propose to re-design the Internet architecture around named data. Such research efforts are commonly identified under the name of Information Centric Networking (ICN).

This thesis focuses on the Content-Centric Networking proposition. We first analyze the CCN communication model with particular focus on the bandwidth and storage sharing performance. We compute closed formulas for data delivery time, that we use in the second part of the thesis as guideline for network protocol design. Second, we propose some CCN congestion control and forwarding mechanisms. In particular, we present a first window based receiver driven flow control protocol named Interest Control Protocol (ICP). Furthermore, we introduce a hop-by-hop congestion control mechanism in order to obtain early congestion detection and reaction. We also extend the original ICP congestion control protocol implementing a Remote Adaptive Active Queue Management mechanism (RAAQM) in order to efficiently exploit heterogeneous (joint/disjoint) network paths. Finally, we introduce a distributed forwarding mechanism that bases its decisions on per prefix and per interface quality measurement without impacting the system scalability. To support our results and test the proposed mechanisms, we have also published the source code of a CCN Packet Level simulator (CCNPL-Sim) under the GNU GPLv2 license.

Keywords: Information Centric Networks — Content-Centric Networking — Performance evaluation — Transport protocol — Congestion Control — Forwarding

Résumé

Dans les dernières années, l'utilisation d'Internet a sensiblement changé en passant d'un modèle de communication centré sur les machines (*host-to-host*) á un centré sur les contenus (*content-centric*). La plus part de services utilisés par les clients d'Internet aujourd'hui sont déjà centré sur les contenus même (la Vidéo á la demande, les jeu en ligne, les réseaux sociaux, etc ...) et pas sur leurs emplacement, comme dans le modèle host-to-host. Même si elle a été capable de répondre á touts les problèmes posé par les nouvelles applications, il y a une différence évidente entre l'architecture d'Internet et son utilisation actuelle.

Dans ce contexte, beaucoup de projets de recherche proposent un changement radicale de l'architecture de l'Internet, en mettent des contenu identifié par leur nom au centre du réseau. Ce group de proposition est généralement identifiés sous le nom de Information Centric Network (ICN). Cette thèse se focalise sur la proposition Content- Centric Network (CCN, aussi connu comme Named Data Networkong - NDN).

Dans une premier temps, nous analysons les performance du modèle de communication CCN en se concentrent sur le partage de la bande passante et de la mémoire. Nous proposons des formules pour la caractérisation du temps de transfert qu'on utilise comme référence pour une phase de design de protocoles dédié à CCN. Deuxièmement, nous proposons un protocole de contrôle de congestion et des mécanismes de forwarding pour l'architecture CCN. En particulier on présent un premier mécanisme de contrôle de congestion, Interest Control Protocol (ICP), qui utilise une fenêtre contrôlé avec le mécanisme Additive Increase Multiplicative Decrease (AIMD) au récepteur et réalise un partage efficace et équitable de la bande passante efficace en présence de mémoires de stockage temporaire dans le réseau. En complément avec le contrôle de congestion au récepteur, nous présentons un mécanisme distribué (hop-by-hop) pour obtenir une détection/réaction à la congestion plus rapide. Nous proposons aussi une modification pour ICP en implémentant le mécanisme Remote Adaptive Active Queue Management(RAAQM) pour exploiter efficacement tous les chemins disponibles dans le réseau qui peuvent présenter de caractéristique très hétérogènes. En fin, nous présentons un mécanisme de forwarding distribué qui base ses décisions sur des mesure de qualité d'interface par chaque préfixe disponible dans le tableaux de routage. Pour supporter nos résultats et tester les mécanismes proposés, nous publions un simulateur pour de réseaux de type CCN (CCNPL-SIM) sous la licence GNU GPL v2.

Mots-clés : Information Centric Networks — Content-Centric Networking — Evaluation de performances — Protocol de transport — Contrôle de congestion — Forwarding

Contents

Abstract iv

Résumé v

1 Introduction 1

1.1 Networking evolution 1

1.2 The Information Centric Network paradigm 3

1.3 Content-Centric Networking (CCN) 13

1.4 Thesis organization and contribution 17

1.4.1 Content Centric Networks modeling 17

1.4.2 Flow control and forwarding mechanisms 18

1.5 Publications and submissions 18

I Content Centric Networks modeling 21

Introduction 23

2 Storage sharing modeling 25

2.1 Related Works 25

2.2 System description 26

2.3 Least Recently Used replacement policy 27

2.3.1 Single LRU cache 30

2.3.2 Content requests aggregation 34

2.3.3 Network of LRU caches 35

2.3.4 Miss rate characterization 35

2.3.5 Miss rate characterization with request aggregation 36

2.3.6 LRU simulation results . . . 37
2.4 Random replacement policy . . . 42
2.4.1 Single cache model . . . 42
2.4.2 Comparing RND to LRU . . . 50
2.4.3 RND Simulation results: single cache . . . 52
2.4.4 In-network cache model . . . 54
2.4.5 RND Simulation results: network of caches . . . 56
2.5 Mixture of RND and LRU . . . 58
2.5.1 Large cache size estimates . . . 58
2.5.2 Mixed RND-LRU Simulation results . . . 59
2.6 Conclusion . . . 62

3 Bandwidth sharing modeling **63**
3.1 Model description . . . 63
3.2 Average content delivery time . . . 65
3.3 Simulation results . . . 66
3.4 Conclusion . . . 69

4 Conclusion of PART I **71**

II Transport protocol and Forwarding Mechanisms for CCN **73**

Introduction **75**

5 Interest Control Protocol - ICP **77**
5.1 Related Works . . . 77
5.2 ICP Design . . . 78
5.3 Preliminary analysis of a single bottleneck . . . 81
5.4 Multi-bottleneck analysis . . . 85
5.5 Design guidelines and protocol assessment . . . 90
5.6 Conclusion . . . 96

6 Hop-By-Hop congestion control **97**
6.1 Related Works . . . 97
6.2 HBH Interest Shaping design . . . 98

6.3 HBH-ICP analysis . . . 100
6.4 Evaluation . . . 103
6.5 Conclusion . . . 107

7 Multipath transport protocol and request forwarding 109
7.1 Related Works . . . 109
7.2 Congestion control mechanism . . . 110
7.3 Dynamic Request forwarding . . . 116
7.4 Optimization framework . . . 118
7.5 Performance Evaluation . . . 121
7.6 Conclusion . . . 126

8 Conclusion of PART II 129

9 CCN Packet Level Simulator - CCNPL-Sim 131
9.1 Workload generator . . . 132
9.2 Simulation run . . . 136
9.3 Simulation outputs . . . 140

10 Conclusion 143

Appendices 149

A LRU cache modelling - proofs 149

B RND cache modelling - proofs 157

C RAAQM proofs 167

D CCNPL-Sim files 169

E Synthèse en français 171
E.1 Introduction . . . 171
E.2 Modélisation des réseaux CCN . . . 176
E.3 Protocoles de transport et mécanismes d'acheminement des requêtes pour CCN 189

Bibliography 205

List of Publications and Submissions **211**

Chapter 1

Introduction

1.1 Networking evolution

The first global telecommunication architecture conceived by mankind was the telephony network. Telephone devices appeared around 1840s, in their early forms, enabling voice transmission between two communication endpoints, connected with a wire. As the devices improved in quality over time, both real needs and economic incentives fostered the development of a global interconnection network. The problem that the architecture had to solve was to find a way to establish permanent circuits (paths) between telephones; to connect them. In the beginning, circuits were built by human operators that manually connected the caller with the other end. As the number of terminals (telephones), and hence of possible connections increased, mechanical and electrical circuit switching was developed to reduce the time needed to establish a circuit.

Computer systems appeared more or less one century after telephone devices. As for the phones, with the computer evolution there were enough interests (military and research) to build an interconnection network. The main goal behind the development of the Internet's architecture was to connect a small number of machines (used by numerous users) to transfer a small amount of data or share scarce expensive devices (such as card readers, supercomputers, etc.).

The only example of a global communication network at that time was the telephony network. Establishing paths between computer devices that needed to communicate seemed to be the best way to transfer data, and researchers concentrated in reducing the time needed to build permanent paths. However, the problem that needed to be solved by computer networks was different from the one solved by telephony. Instead of building permanent circuits, a computer network needed to efficiently move encoded information from one point to another, thus focusing on end-host transmission, not path construction.

The adopted solution was to send independent data packets, forwarded using the name of the

final destination (packet switching) contained in the packet itself. First packet switch networks like ARPANET (an interconnection between universities' computers and research laboratories in the USA) and MINITEL (a French chat online service accessible through telephone lines) appeared around late 1970s. An important step for the development of the network architecture was the introduction of the TCP/IP suite of protocols (Transmission Control Protocol and Internet Protocol, in the 1980s), which provided the fundamental building block for the Internet's development and is still used in today's network.

With the introduction of the World Wide Web (1990), the Internet usage started to shift from host-to-host to service access and data retrieval communication. Services like VoD, IPTV, on-line gaming and social networking represent the predominant usage of today's Internet. Despite the rapid and constant evolution of applications/services and phisical/access technologies the Internet Protocol (IP) remained basically unchanged since its introduction. The key of this succes was the simplicity of the IP protocol that allowed the deployment of many different services while supporting the usage of a wide range of new access and physical technologies. Fig.1.1 represents the Internet model (commonly known as *hourglass model*) in which IP is the so called *thin waist* that introduces minimal functionalities for transmitting addressed packets between two hosts.

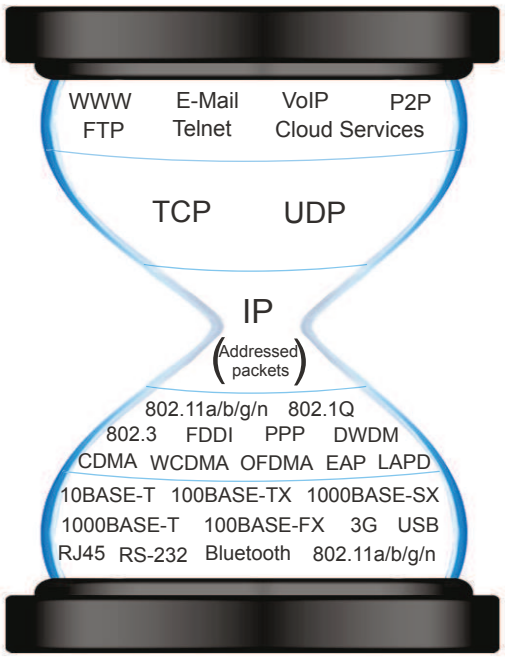


Figure 1.1: Internet *hourglass* model

Even if the original Internet's architecture has been able to address new applications' requirements, developing a large range of solutions (as for example DNS, NAT, Proxy, CDNs, IPv6, IP multicast, Mobile IP, etc...), its original objective is to solve the problem of host-to-host communications and not content dissemination/retrieval. The mismatch between Internet's main usage (content retrieval) and its architecture (host-to-host based) creates many problems for network operators, network administrators, application developers and network engineers in terms of security, availability and data location dependence.

Many research projects, focusing on Future Internet's architecture design, have been funded in the last years (e.g. US NSF GENI [1], EU FIA [2] and AsiaFI [3]). Information Centric Networking (ICN) proposals in particular address the previously discussed architectural problems, making named data, instead of its physical location, the central element of the architecture communication paradigm.

1.2 The Information Centric Network paradigm

The ICN paradigm consists of redesigning the Future Internet architecture, placing named data rather than host locations (IP addresses) at the core of the network design. In order to overcome the mismatch between the current use of the Internet and its original design, many ICN architectures have been proposed by the networking research community. In this section, we will introduce some notions and concepts relevant to the ICN paradigm.

In ICN architectures, data are identified by location-independent (often globally unique) names. Different types of naming schemes have been proposed so far, with the intent to achieve name persistence and provide an efficient and scalable look-up for billions of different possible names. Among the different naming propositions, two main categories can be identified: flat and hierarchical names. The flat naming scheme consists of identifying data with a flat label, using methods such as Distribute Hash Table (DHT) based look-up, to achieve scalability. Hierarchical naming scheme instead, uses a hierarchical set of labels (e.g. URIs) to exploit name prefixes aggregation as an answer to scalability requirements. Both naming scheme have advantages and disadvantages in terms of security, scalability, etc. and naming remains an open research issue.

Security, is also an important aspect of ICN architectures. Since data may come from any network element, the architecture must secure the content object rather than the communication path, contrarily to the security model commonly adopted in the current Internet architecture. Frequently in ICN, security is strictly related to naming and any ICN architecture proposes a slightly different security framework exploiting traditional security mechanisms as digital signature, certificate, etc.

As a shared principle, ICN architectures are aware of the named data they are delivering, many proposals introduce caching capability directly at network level. Every network element of

the ICN architecture, can therefore temporarily store a copy of the data packets that traverse it, and reply to successive requests for the same data. In this way, ICN architectures naturally offer a fast, reliable and scalable content delivery system, without using any external caching solution such as proprietary Content Delivery Networks (CDN), commonly adopted in the current Internet architecture.

Furthermore in the ICN communication process, senders (a cache or a content producer) do not send content directly to the user and data delivery is completely controlled by the receiver through requests or subscriptions to named data or services. The ICN architecture has the responsibility to resolve the data name in order to find the closest copy and deliver it to the receiver. ICN networks can resolve names in two ways: with a look-up-by-name in a distributed database or route-by-name the request to a nearby data copy.

Hereafter, we briefly recall some of the ICN architectures proposed so far.

Data-Oriented Network Architecture (DONA)

One of the first proposals of ICN architecture is DONA [4] that basically redesigns the Internet naming system. It replaces DNS names, with flat, self-certifying names and provides name-based primitives (above the IP layer) for the name resolution phase. The main objectives of DONA with the proposed naming and name resolution system, are the following:

- **Persistence:** Once a name is assigned to data by the DONA architecture, it should remain valid as long as the named data is valid. In other words, even if the data is moved or changes administrative domain, the name should remain the same.

- **Authenticity:** Users that receive named data in the DONA architecture, should be able to verify that the data came from the appropriate source and not from a misbehaving server/user.

- **Availability:** Users that need a service or a data in DONA, should be able to locate them with high reliability and low latency. DONA, needs to allow user requests to locate nearby copies of the desired data that can be stored in a nearby server.

Flat self-certifying names

DONA naming scheme consists in associate each datum or service with a *principal* that represents the owner or the purveyor of the named entity. Each principal is identified by a public-private key pair and represented by a hash of its public key P. Names under the responsibility of a principal, will be of the type P:L where L is a unique, flat identifier chosen by the principal. When a named data is sent to a receiver, it is associated with the principal's public key and signed using the

private key (a data is represented by the triplet <*data,public key,signature*>). Clients can verify data authenticity with the public key and data signature enclosed in the received object, assuring authenticity. Finally, data persistence, is granted by the fact that names do not refer to a particular location and can be hosted everywhere, maintaining the same name.

Name resolution

Given the name structure, DONA needs a mechanism to translate identifiers into locations and provide data availability. It uses the route-by-name paradigm for name resolution, relying on an infrastructure of new machines called Resolution Handlers (RH). RHs are organized in a hierarchical network that can be seen as an extension of the ASes structure with the same relationships between different ASes' RHs. Each RH has a registration table that maps a name to the couple composed by next-hop RH and distance from the copy. In order to translate the name into a network location, two simple primitives are made available in the RHs' network: FIND and REGISTER.

In order resolve object names, clients issue the FIND command that is routed through different RHs toward a nearby copy. When a RH receives a FIND command it forwards the FIND to the next-hop RH if there is an entry in the resolution table, otherwise, it forwards the FIND to its parent RH. If the FIND reaches a copy of the requested data, it initiates the communication between a nearby server and the requester. Packet exchanges that occur after a FIND, are transferred using standard TCP/IP routing, forwarding and transport protocols. In order to implement cache capabilities within the resolution process, RHs that receive a FIND can issue a new FIND packet (instead of forwarding the received FIND) for the same name. In this way the FIND response is forced to traverse the RH that receive the data, cache it in a local memory and directly reply to future FIND messages.

REGISTER commands on the contrary, are used by the network to set up the necessary state for the RHs' network to effectively route FIND messages. When a RH receives a REGISTER message, it will only forward the command if there are no entries for the registered name or the new REGISTER comes from a copy closer than the previous one. Moreover, REGISTER messages have a TTL and need to be periodically refreshed. The DONA architecture also provides methods to invalidate REGISTER operations such as UNREGISTER, in order to indicate that the data is no longer available at a certain location. Notice that REGISTERs must be authenticated in order to avoid fake registrations in the network such as registrations by entities that do not have the right to deliver that data. Fig. 1.2 shows registration and resolution phase in the DONA architecture.

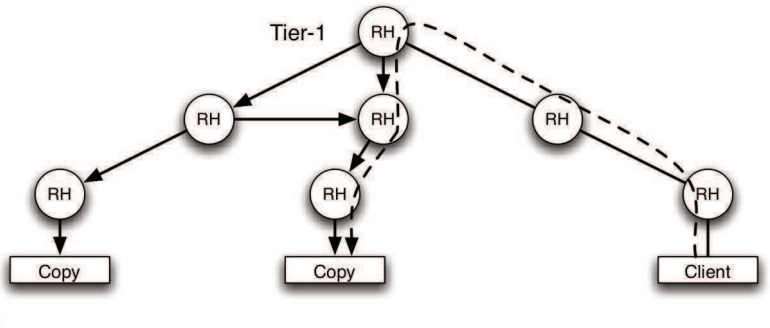


Figure 1.2: Registration (solid arrows) and resolution (dashed arrow) phases in the DONA's RHs structure. Source: [4]

Network of Information (NetInf)

Another important contribution to the ICN research field, has been given by the 4WARD European project [5] and its successor SAIL [6]. The two projects design an architecture framework and architecture elements for naming, name resolution, caching and transport in order to provide access to named data objects in an ICN. The main result of the two projects is NetInf (Network of Information) that describes the proposed ICN architecture.

Naming and security

In NetInf, data are called Named Data Objects (NDOs) and each of them is named using a unique, location-independent name, described using some meta-data information attached to the object. The name of each NDO is composed by three parts: a *scheme*, an *authority* and a *local* part. The scheme part consists in a Uniform Resource Identifier (URI), specifically developed for NetInf called Named Information (ni) and defined in [7]. The ni scheme, provides a flat space of not human-readable cryptographically identifiers. More specifically, names for the NDOs are obtained using a cryptographic hash function applied to the object itself. In order to guarantee name uniqueness, the hash function, needs to be collision free meaning that two different objects must result in two different hash values (and hence names). The authority part is optional and it is provided in order to assist the routing and forwarding processes, indicating the owner/publisher of the NDO. Notice that, since the authority part of the name is optional, each node within the NetInf network, needs to be able to deal with fully-flat names composed only by the hash of the object. The third and last part of a NDO's name is the local one that contains an identifier for the resource that is meaningful in the context of the authority.

Given the naming scheme, data authenticity (security) can be easily achieved as a hash of the object is directly included in the name. An application that needs to verify the binding between

the name and the received data, can simply compute the hash of the received object and compare it with the one contained in the name used for the object request.

Routing and Forwarding

The NetInf protocol, provides three fundamental primitives to request, publish or search Named Data Objects:

- **GET/GET-RESP**: The GET primitive is used to request an instance of a NDO using its identifier. A NetInf node that has the requested object, replies to the GET with a GET-RESP using the GET message identifier in order to bind the two messages.

- **PUBLISH/PUBLISH-RESP**: The PUBLISH primitive, is used to insert a new name in the NetInf architecture and optionally, register a physical copy of the object. The reply to the PUBLISH is the PUBLISH-RESP that communicates the state of the publication (published, existing object, etc..) to the publisher.

- **SEARCH/SEARCH-RESP**: Finally, the SEARCH primitive is used by a receiver to search objects in the NetInf architecture. It contains some keywords that are matched to the meta-data of the published objects, in order to find the desired content. The SEARCH-RESP is the response to a SEARCH and contains a set of NDO names that match the query keywords.

The fundamental NetInf primitive is the GET that needs to be routed/forwarded to a copy of the requested object. The request forwarding in NetInf consists in a hybrid routing/forwarding scheme including name-based routing and name-resolution based aspects.

The name-based part is primarily intended for edge domains. Each NetInf node belonging to an edge domain, has a ni forwarding table that is used to perform local name-based routing. In addiction to forwarding tables, a NetInf router can store objects using its local cache maintaining a list of NDOs that it serves locally. If the request reaches the border of the edge network without finding the object, the ni name is resolved through a global Name Resolution System (NRS) that can be implemented using a Multiple Distributed Hash Table (M-DHT). The NRS replies with the route that needs to be used to find the ni object.

When the GET reaches the requested object, the reply is sent back to the request originator using some soft state left in traversed routers. Fig. 1.3 shows an example of a GET message forwarding that, once arrived to the border of the edge network, is resolved using the global NRS.

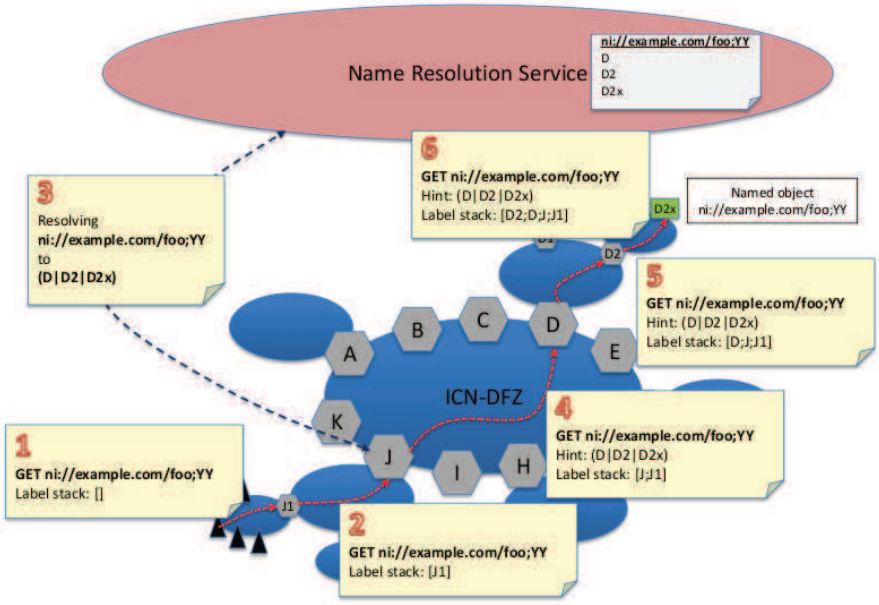


Figure 1.3: NetInf forwarding example. Source: [8]

Transport

NetInf nodes use the NetInf protocol that makes use of lower-layer networking technologies such as TCP/IP or even a new dedicated and optimized transport protocol to transfer objects between them. In order to be able to use different lower layer technologies between nodes, NetInf defines the Convergence Layer (CL). The CL is directly built on top of the different lower-layer technologies adopted, allowing NetInf nodes to establish hop-by-hop communication without the need of any specific NetInf's node identifiers.

Using the CL, NetInf nodes request and receive NDOs that are considered to be Application Data Units (ADUs). However, additional segmentation can be required at the CL while transporting an NDO from a node to another (i.e. IP fragmentation). Hence, NDOs are the smallest addressable and verifiable objects in the NetInf architecture framework. In general, it is up to the application (or the publisher) to define if the NDO refers to a large object (i.e. an entire data video) or a fragment of it. If many NDOs belongs to a larger object, a special indexing NDO can provide the list (or the way to retrieve it) of NDOs needed to reconstruct the original object.

Publish and Subscribe Internet Routing Protocol and Publish Subscribe Internet Technology (PSIRP/PURSUIT)

The Publish-Subscribe Internet-working Routing Paradigm (PSIRP), also known as Publish Subscribe Internet Technology (PURSUIT), is an ICN publish and subscribe architecture for the Future Internet. PSIRP/PURSUIT (simply PURSUIT in the following) architecture, is the outcome of the two homonym European projects [9], [10].

Naming and security

In the PURSUIT architecture, as in most of the proposed ICN architectures, each Information item or any piece of data that can be used within the network, has to be specifically named. The PURSUIT architecture uses flat names called Rendez-vous Identifiers (RIds) that uniquely identify Informations. Notice that, such Identifiers and the associated semantic are meaningful only in the context of the application/service that uses them.

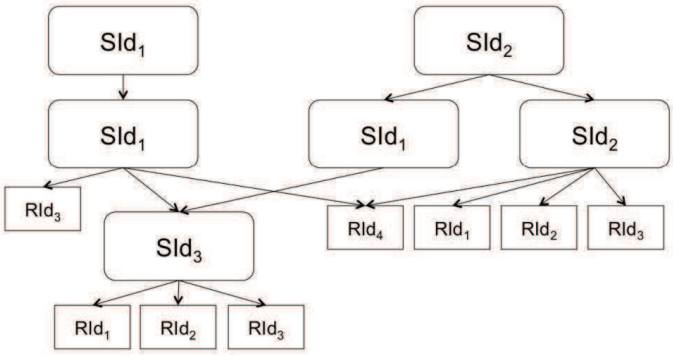


Figure 1.4: PURSUIT naming structure. Source: [11]

In addition to Information item identifiers, the PURSUIT architecture defines scopes. Scopes are identified through Scope Identifiers (SIds) and represent a set of Information items disseminated in the network. SIds, are treated as information items and can group other sub-scopes. Notice that, each Information Item is placed in at least one scope and, a single RId, can be published under different scopes as it is used for many applications/services.

Given SIds and RIds, each information item (a scope or a physical data) used in the PURSUIT architecture, can be identified using a set of identifiers, starting from a root scope i.e. */SId1/SId1/SId3/RId1*, that identifies the information item RId1 (Fig.1.4 shows an example of the PURSUIT naming structure).

As for the majority of the ICN proposals, in PURSUIT, security is important in order to guar-

antee that the data delivered by the network corresponds to what the user asked (subscribed) for. To assure security, the PURSUIT architecture uses self-certifying identifiers that are directly derived from the public key so that any network node has the ability to establish the content's validity and avoid DoS attacks. RId and SId identifiers in particular, have a structure similar to the one adopted by DONA (i.e. Public key:Label).

Publish and subscribe communication model

As its name suggests, the communication model adopted within the PURSUIT architecture is Publish/Subscribe. Network users can publish a piece of information in a particular scope in order to make it available. Clients, can then subscribe to individual information items (or even to entire scopes) to receive updates on the specified information (scope).The publish/subscribe model is implemented by the PURSUIT architecture through different network primitives:

- **create_scope**: used to create a scope starting from parent scopes (i.e. /SId1/SId1 in Fig.1.4 for SId3) or to create a root scope if no other root scopes have the same identifier;
- **advertise_info**: creates a new piece of information under the specified scope hierarchy;
- **publish_info**: publishes a piece of information under the specified scope hierarchy and, based on the dissemination strategy (configurable through the primitive), send it to the subscribers;
- **unpublish_scope / unpublish_info**: used to delete all the references to a scope (and its child scopes and information) or to an information item from the architecture;
- **subscribe_to_scope / subscribe_to_info**: subscribes a user to a particular scope (and optionally to its child scopes) or to an information item;
- **unsubscribe**: invalidates the subscription command related to a scope or to an information item.

Even if the basic communication model is publish/subscribe, using this communication model, the architecture can also support traditional request-response services. This is realized including a response identifier in the subscription that will be used by the server to publish the requested information.

Core functions

The PURSUIT architecture, uses three fundamental core functions in order to disseminate and manage the information published by users.

- **Rendez-vous**: the rendez-vous is the fundamental network function that allows to verify the match between information publication and users' subscriptions. In the PURSUIT architecture, users are connected to attachment points that forward primitives (publish, subscribe, etc.) to local Rendez-vous Nodes (RNs). The main role of RNs is to keep track of publications and receive subscriptions in order to find if there is a match between them. RNs are distributed across the network and organized using, for instance, a Distribute Hash Table (DHT) that partitions the name space among different RNs. In this way, when a RN receives a publication or a subscription, it verifies the identifier and forwards the message to the responsible RN, if it cannot be locally managed;

- **Topology Management**: once there is a match between a publish and a subscribe, the topology management layer is responsible for finding the optimal forwarding paths (a delivery graph) to deliver the information from the publishers to the subscribers, taking into account current network conditions. Every administrative domain, is controlled by a single topology manager that collects the network information (topology, load, etc.) of its local domain and exchanges it with other domain managers;

- **Forwarding**: the last function needed to deliver data from publisher(s) to subscriber(s) is the forwarding one. The delivery graph is constructed by the topology management function and encoded using specific forwarding identifiers constructed through the Bloom-filter based Relay Architecture (BRA). Given the forwarding identifiers, the forwarding layer is responsible of making sure that the information is correctly delivered to subscriber(s), avoiding wrongly forwarded data. Notice that, as for other ICN networks, any network's node involved in the content forwarding process, can opportunistically cache the information in order to re-use it in the future.

Content Based Networking (CBN)

Another important research contribution to ICN is given by Content Based Networking [12], [13]. CBN designs a content-based publish/subscribe mechanism, coupled with a name-based routing and forwarding scheme. In the CBN architecture, as in most of the ICN proposals, nodes are not identified using a unique network address and packets does not need to be specifically addressed to any node. That is, messages from sender to receivers, are driven by the CBN network using the content of the message itself. Even if its design is Information Centric, CBN is slightly different from other ICN architectures as it is not a proposal for the future Internet's architecture. CBN is not a global communication infrastructure like other ICN proposals, but it is tailored for a class of application domains such as news distribution, publish/subscribe event notification, system

monitoring and many others.

Content-based names and predicates

In CBN, the content of a message is described through a set of attribute/value pairs that constitute the flat content-based address (name) of the message. Nodes wishing to receive data messages, advertise selection predicates that are a logical disjunction of conjunctions of constraints over the values of messages' attributes. Here is an example of a message and a selection predicates (that match the message):

message: [class="warning", severity=1, device-type="hard-drive"]
selection predicate: [(class="warning") $\vee$ (alert-type="software-error" $\wedge$ severity=1)]

Notice that contrarily to other ICN architecture designs, in CBN, messages are not assigned to a unique identifier, and a selection predicate expressed by a node may match more than one message and vice-versa.

Routing scheme

In the CBN architecture, a content-based routing mechanism [13] is implemented on top of a broadcast layer (see 1.5). The broadcast layer constructs acyclic broadcast trees that provide minimal paths between source and potential clients. The content-based layer instead, is responsible for routing messages only to clients that are interested in it, pruning the broadcast trees provided by the broadcast layer.

The routing table of a router, associates a selection predicate to each interface, including the local one (that represents the selection predicate of the local applications) and is updated through a "push" or a "pull" mechanism. Receivers push their interest in the network through *Receiver Advertisements* (RAs) packets that carry the predicate indicating the messages of interest for that node. A router instead, uses the *Sender Request* (SR) packet in order to pull predicates that indicating the receivers' interests and update its routing table. Receivers reply to the SR with *Update Replies* (URs) that carry their current content-based address combined with those of downstream nodes.

Forwarding scheme

Each node in the CBN architecture, maintains a content-based forwarding table in order to properly forward messages to interested users. The forwarding table binds a single content-based address to each node's interface and is constructed starting from the information contained in the routing

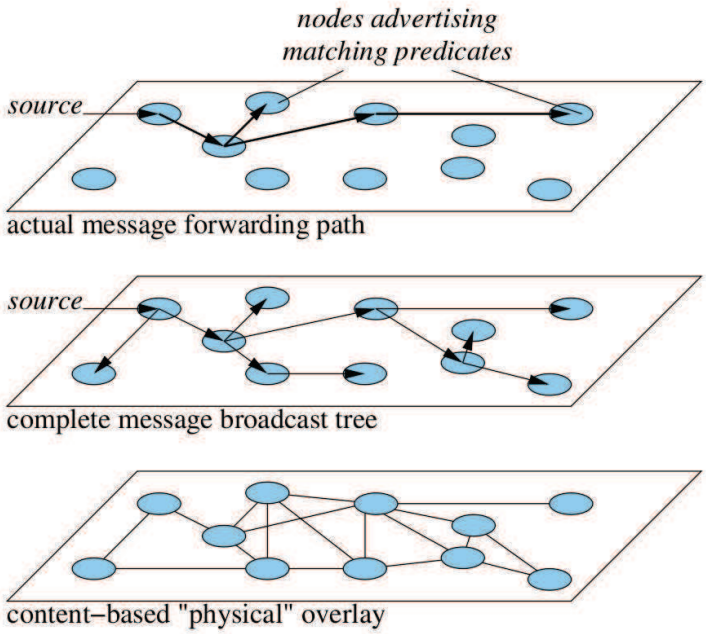


Figure 1.5: High level CBN routing scheme. Source: [13]

table. A message is broadcasted from a server through its broadcast tree. When a message is received by a CBN node, a look-up of its content-based address in the forwarding table selects the subset of interfaces on which the node should forward the message.

1.3 Content-Centric Networking (CCN)

A recent proposition by the Palo Alto Research Center (PARC), Content-Centric-Networking [14], elaborates some ideas and principles previously introduced by other ICN proposals. CCN proposes a clean-slate Internet architecture including naming and forwarding functionalities, taking care of security, content dissemination and mobility issues. The CCN design mainly focuses on layers 3 and 4 of the Internet model (see Fig.1.1), maintaining the simplicity of IP in order to become the *thin waist* of the future Internet Architecture.

Recently, projects focused on CCN called Named Data Networking (NDN) [15] and Content-Oriented Networking: a New Experience for Content Transfer (CONNECT) [16], has been funded by the respective National research agency (American's National Science Foundation - NSF and Agence Nationale Recherches - ANR).

The work described in this dissertation is mainly focused on the principles proposed in CCN even if its results has broader applicability in the context of general ICN proposals and next gen-

eration CDNs. Hereafter the description of the CCN's architecture.

Hierarchical names and security

Instead of identifying machines as in the today's Internet, the CCN architecture (as many ICN proposals), identifies data objects with names. Data objects are then split in many packets, each one identified using a globally unique name. CCN's names are composed by a variable number of components, organized in a hierarchical structure. Hierarchical structured names, with respect to flat names, have the advantage that they can be aggregated in prefixes, significantly reducing the size of the forwarding table on a CCN's router, in order to speed-up the forwarding process (detailed in the next section).

As presented in [14], Fig.1.6 shows an example of the naming scheme in which a video file, is identified by the name */parc.com/video/WidgetA.mpg/_v<timestamp>/* and its chunks with names of the type */parc.com/video/WidgetA.mpg /_v<timestamp>/_s_id* where _s_id represents the segment (data packet) identifier.

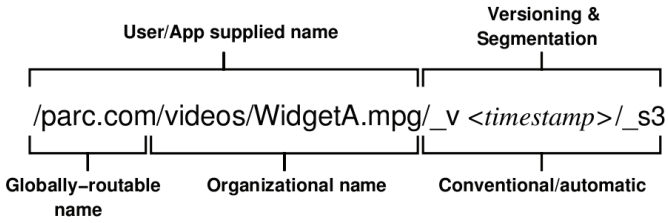


Figure 1.6: CCN's hierarchical name example. Source: [14]

In order to assure integrity and authenticity of the content object, CCN does not use self certifying names as other ICN proposals. Instead, it directly authenticates the binding between the name and the identified data. The content publisher wishes to make its data available to users, distributes them as a mapping triple: $M_{(N,P,C)} = (N, C, Sign_P(N, C))$ where C represents the content object, P the publisher's public key and N the name chosen by the publisher to represent C. When a consumer receives $M_{(N,P,C)}$ in response to a query for N, no matter from whom or from where, it can verify the data's integrity and authenticity, using the public key of the publisher (retrieved using some certificate system or embedded in the content object's meta-data).

Forwarding

In the CCN architecture there are two different packet types:

- **INTEREST**: used to request a particular data using its content name N. It contains the

content name N of the data, some meta data and a random nonce value used to prevent interest loops in the network.

- **DATA**: any node that receives an INTEREST packet, can reply to it with a DATA packet. It contains the content name of the data, signature and security information as well as the requested data. A DATA can be compared to a single packet in the current Internet architecture, but its size is not strictly defined in CCN. Notice that a DATA satisfies a single INTEREST so that the receiver does not receive packet that it does not request.

In CCN the communication is initiated by the users that send an interest packet in order to query the network for a particular named data. Interests are then routed by name toward possible data location(s). More precisely, once a router receives an interest, it will first check in its Content Store (CS, a local cache in the router) to see if it has a copy of the requested content. If so, the node can reply directly to the interest with the data, if not, it checks the Pending Interest Table (PIT). The PIT table is used to take trace of the forwarded interest, storing the content name and the set of interfaces from which the interest was received. If an interest for the same name N has been previously forwarded upstream, the received one is stopped, avoiding multiple requests for the same chunk (allowing native multicast), and the interface from which it was received is locally recorded. Finally, if a match is not found in the CS nor in the PIT, the router performs a longest prefix match look-up in the Forwarding Information Based (FIB, similar to the one used in IP). If there is a match in the FIB, the interest is forwarded and a new entry is added in the PIT table. Finally, if none of the three structures (CS, PIT, FIB) return a match, the interest is discarded by the node.

Unlike interests, data packets are not routed in the network and exploit the information left by the interest in the PIT to find the route back to the requester(s). When a CCN node receives a data packet, it first checks if the data is already stored in the CS, if so it discards the packet. After that, if there is a matching PIT entry (meaning that an interest previously requested the data), the data packet is forwarded to the interfaces indicated by the PIT entry, and eventually stored in the local CS (managed with a standard replacement policy, i.e. LRU, RND, etc.). If there is no match in the PIT, the data is simply discarded. Notice that, with the described data forwarding mechanism, the path followed by the data in downstream, is constructed by the interest in upstream, creating a symmetry between the up and the downstream (not always true in the current Internet architecture). As depicted in [14], Fig. 1.7 shows the forwarding engine that includes FIB, PIT and CS of a CCN node with three interfaces, one internal (a local application) and two external (wired and wireless).

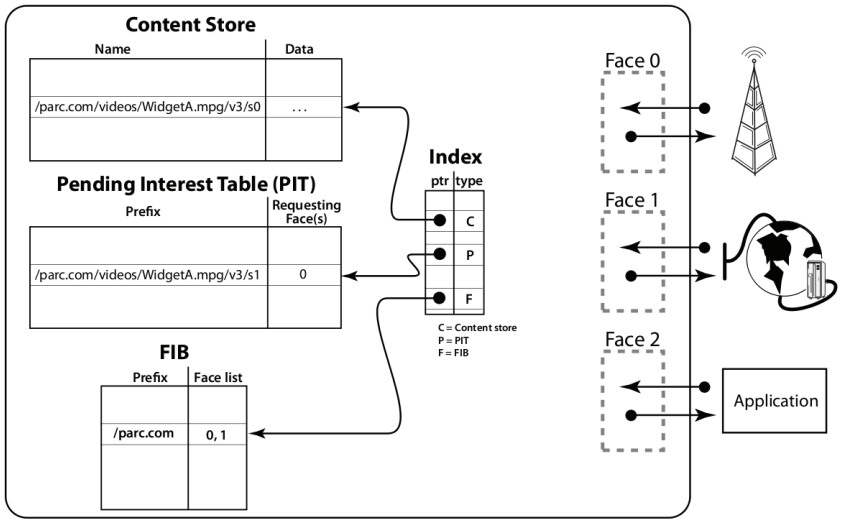


Figure 1.7: CCN forwarding. Source: [14]

Routing

As previously said, in the CCN architecture, content objects' names are used instead of machine identifiers (IP addresses). Hence, a routing protocol for CCN, should distribute name prefixes rather than IP subnets (a subnet can be seen as a IP address'prefix), in order to update the Routing Information Base (RIB) that is then used (as in today networks) to populate the FIB. As the CCN routing should not differ much from the one successfully adopted in IP, slight modification of the existing solutions such as OSPF or IS-IS (Intra-domain) and BGP (Inter-domain), could be adopted as CCN's routing protocol. However, routing in CCN remains object of current research.

Traffic and Resource Management

In the current Internet architecture, transport layer functions such as flow control, congestion avoidance, error detection, etc. are implemented at the host level as the communication is end-to-end, involving exactly two hosts, namely the sender and the receiver. On the contrary, in CCN as in many of the ICN proposals, the role of the end-hosts is completely different. The communication session, is content-centric rather than end-to-end, involving the receiver that requests the data and many intermediate nodes that participate to the content retrieval process as they store part of the requested data. As a consequence, the transport layer is based on a single communication endpoint: the receiver.

The original CCN architecture, specifies no traffic control mechanisms. Receivers are simply allowed to emit interest packets in order to receive data as a response. However, similarly to TCP's

sender variable window size, a CCN host can have multiple outstanding interests in parallel, in order to fill the bandwidth-delay product and achieve the theoretical maximum throughput of the network.

1.4 Thesis organization and contribution

This dissertation is organized in two main parts. The first one is devoted to *Content Centric Networks modeling* while the second one focuses on *Flow control and forwarding mechanism* design for CCN.

1.4.1 Content Centric Networks modeling

The objective of the first part of this dissertation, is to provide an analytical framework to assess CCN architecture performance. This preliminary analysis is fundamental in order to deeply understand the CCN architecture and to drive flow control and forwarding protocols' design.

In Chapter 2, we analyze two cache replacement policies namely Least Recently Used (LRU) and Random (RND). We derive analytical characterization of the cache performance in the context of CCN and, more generally, any network of caches. We write asymptotic formulas for the RND and LRU cache miss probability assuming infinite cache size and specific content object popularity. Furthermore, we compare our analytical results with packet level simulations verifying the model accuracy. We show that the performance of LRU and RND replacement policy is close, especially when the cache are deployed close to the repository. For this reason, and due to the fact that RND needs less memory access than LRU, we suggest that RND can be used when the request rate is high, and memory speed becomes crucial. Finally, in this direction, we analyze the performance of mixed LRU-RND networks.

In Chapter 3, we introduce the bandwidth sharing model using max-min as a fairness objective. Exploiting the results of Chapter 2, we study the interplay between bandwidth and storage resource sharing, and provide a user performance analysis tool for the the CCN architecture. Furthermore we compare our formula with packet level simulations to confirm the model accuracy. Our results highlight that most popular content objects, can exploit the proximity of the content, only if the links from the user to the cache that store the information does not constitute a bottleneck. In this case, caches are useless and the user perceived performance is the same, independently from the content object's popularity.

Chapter 4 concludes the first part.

1.4.2 Flow control and forwarding mechanisms

Due to its content awareness, CCN presents some differences with respect to the traditional TCP/IP communication as receiver-driven transport, natural multipath communication, etc. In the second part of the dissertation, we try to design efficient flow control and forwarding mechanisms for CCN.

In Chapter 5, we present a first design of a CCN flow control protocol namely Interest Control Protocol (ICP). ICP is a window, receiver-based flow control protocol with the Additive Increase Multiplicative Decrease mechanism successfully used in TCP, that uses a delay measure as a congestion indication. The ICP design objective, is to control a single path to the repository, in which contents can be retrieved from the original server or any intermediate cache. We show the ICP convergence to the efficient and max-min fair equilibrium presented in the previous part and perform simulations to show the ICP behavior.

In Chapter 6, we introduce an hob-by-hop interest shaping mechanism coupled with ICP. The objectives of the hob-by-hop interest shaping mechanism are to accelerate congestion reaction and reduce the loss rate. Moreover, we show that ICP coupled to the proposed HBH mechanism converges to the same equilibrium of ICP (without the HBH).

In Chapter 7, we extend our first flow control proposition, ICP, with a Remote Adaptive Active Queue Management (RAAQM) mechanism to efficiently control different paths (possibly disjoint), during the same content retrieval process. The proposed protocol is (as ICP) a window, receiver-based flow control protocol with the AIMD mechanism, that uses a probabilistic window decrease with a different probability associated to each controlled path. We also prove the protocol stability with and without constant propagation delays in the single path case and perform extensive simulations to assess the stability and to guide parameter setting. Furthermore, we present a distributed forwarding algorithm that uses the average number of pending interests as paths' performance metric to make the forwarding decision. Finally, we compare the proposed mechanism to the optimal forwarding strategy. Chapter 8 concludes the second part.

In Chapter 9 we present the structure of the Content Centric Network Packet Level Simulator (CCNPL-Sim) developed in C++ and used to verify both the analytical and the design part. Finally, Chapter 10 concludes this dissertation.

1.5 Publications and submissions

The content of this dissertation has been partially published in international conferences and workshops. The results of the first part are presented in [75], [76] and [77]. Furthermore, an article

combining and extending the results of [75] and [77] has been submitted to the Computer Networks journal and is currently under review ([78]). Moreover, the extension of [76] has been submitted to the Performance Evaluation journal [79]. The content of the second part is presented in [80] and [81].

During the thesis we also considered the problem of per-application storage management Content Centric Networks. However, the outcome of this work is not presented in this thesis and an article has been submitted and is currently under review ([82]).

Part I

Content Centric Networks modeling

Introduction

Recent ICN proposals like CCN, NetInf, etc. introduced caching mechanisms directly embedded into the network, potentially equipping each network device with a cache. In this context, a preliminary understanding of transport and caching issues in ICN appears to be necessary in order to analyze the architecture performance, to quantify potential benefits and to guide optimized protocol design. In the first part of this dissertation, we analyze the user perceived performance in the CCN architecture (more generally any caching network). In order to do so, we need to consider the complex interplay between the two resources influencing the user performance namely *storage* and *bandwidth*.

In Chapter 2 we first address performance issues of CCN nodes running LRU and/or RND replacement policies. Even for a single storage node of the CCN architecture, the analysis of cache dynamics is not straightforward. When the scenario under study is a network of caches operating under CCN principles, different modeling issues arise. The output process of the first node, that is the process of missing chunk requests, needs to be precisely modeled in order to define the modified request process feeding upstream caches. In this scenario, we provide asymptotic formula for the miss rate, and hence for the storage performance analysis for the LRU and RND caching policies, in the case of a heavy-tailed content popularity (the Zipf distribution).

Once studied the storage sharing performance of a network of caches, we need to take into account limited bandwidth resources as it is the case for the CCN architecture. The problem is that in CCN, as expected and shown in Chapter 2, packets belonging to the same content can be retrieved from different nodes. For this reason, during the same download process, packets can experience different network delays and hence different download times, as they are downloaded from different nodes. In Chapter 3 we consider the max-min bandwidth sharing (which aims at fair rate sharing) and provide an analytical tool for the CCN performance analysis.

Chapter 2

Storage sharing modeling

2.1 Related Works

In the context of Web caching there have been previous attempts to modeling content-level cache dynamics, most of them related to a single cache scenario under Least Recently Used (LRU) replacement policy. The LRU replacement policy, consists in moving the latest requested object in front of a list of stored objects and, removing elements from the tail of the same list when a new element needs to be inserted. Analytical models of LRU caches exploit the analogy between the LRU miss probability, and the tail of the search cost distribution for the Move-To-Front (MTF) searching algorithm as described by King in [17]. In [18] Flajolet et al., derived an integral expression for the Laplace transform of the search cost distribution function, that needs to be numerically integrated with complexity proportional to the cache size and the number of content items of the catalog. An asymptotic analysis of LRU miss probabilities for Zipf and Weibull popularity distributions proposed by Jelenković et al. in [19], provided simple closed formulas. A more recent work by the same author [20], provides an analytical characterization of the miss probability and thus miss rate under Poisson assumptions of content requests' arrivals. Extensions to correlated requests are obtained in [21, 22] by Jelenković et al., showing that short term correlation does not impact the asymptotic results derived in [19]; the case of variable object sizes is also considered in [23] by the same author. The average miss probability for a LRU cache when requests are issued according to a general, may be non-stationary, stochastic point process is also derived in [24].

Contrarily to LRU, the RND replacement policy, is less studied in the literature. As its name suggests, the RND replacement policy consists in choosing at random an element in the cache when a new one need to be inserted. The analytical evaluation of the RND policy has been first initiated by Gelembe in [25] for a single cache where the miss probability is given by a general expression.

Despite the big amount of cache's studies, almost all of these prior works are devoted to the analysis of a single cache and we are interested in network of caches in the context of CCN/ICN architectures. A network of LRU caches has been analyzed by Towsley et al. in [26] using the approximation for the miss probability at each node obtained in [27]. Authors assume that the output of a LRU cache follow the Independence Reference Model; miss probabilities can then be obtained as the solution of an iterative algorithm that is proved to converge. The interconnection of LRU caches using the Leave a Copy Down mechanism (each cache of a given level stores the transmitted object) is analyzed in [28]. Mixed replacement policies are analyzed in the context of hierarchical cache in [29], and hybrid policies with good performance and implementation cost are proposed. In [30] Che et al., applied a single LRU cache approximated model to study a cache coordination technique. The same approximated model has also been used by Laoutaris et al. in [31] and more recently in [32] by Roberts et al.

2.2 System description

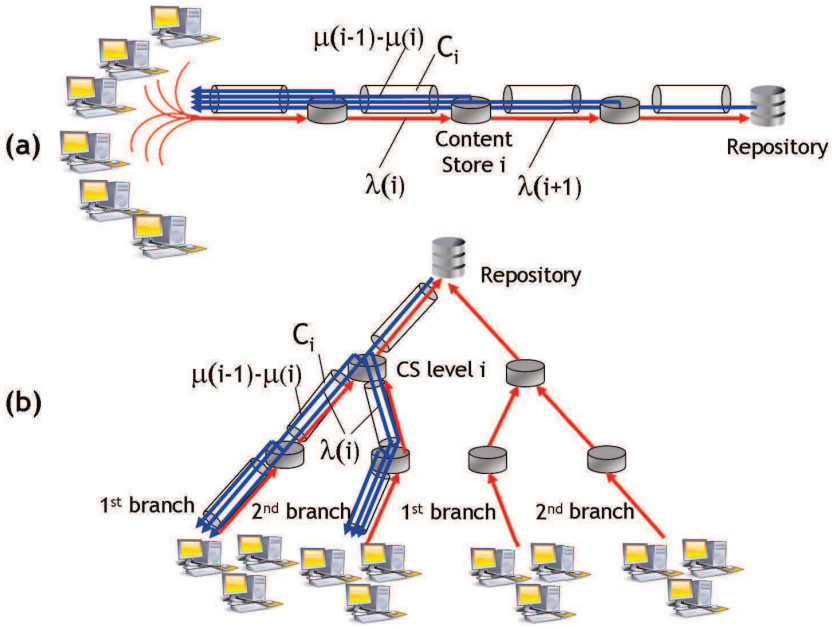


Figure 2.1: Network topologies: line (a), binary tree (b).

The ICN architecture we focused on, is the one described in [14] and previously introduced in Sec. 1.3. Even if we consider a specific architecture, our storage sharing model has broader applicability and can can be used for any network of cache.

We consider Content items uniquely identified by a name, and permanently stored in one (or more) repository (a network node that permanently store the whole content). Users can retrieve them using a receiver-driven transport protocol based on per-packet *interest* queries triggering data packet delivery. A name-based routing protocol guarantees that queries are properly routed towards data repository. Every intermediate node keeps track of pending queries, in order to deliver the requested packets back to the receiver through the reverse path of the request (as in[14]). In addition, exploiting this mechanism, intermediate nodes perform request aggregation (also denoted as filtering), i.e. avoid forwarding multiple requests for the same packet while the first one is pending. Finally, each router is equipped with a local cache managed with an LRU/RND replacement policy that store data packets using the In-Path Caching (**IPC** each valid, received data is cached) scheme in order to satisfy future requests for the same packet.

2.3 Least Recently Used replacement policy

Model Description

In the above described system, data may come from the repository or from any hitting cache, along the path towards the original repository. Packets of the same content can therefore be retrieved from multiple locations experiencing different round trip times (RTTs) and affecting the delivery performance. The resulting throughput and hence content delivery time are strongly affected by this notion of average distance between the user and the packets of the requested content item, which we will explicitly define by the variable *Virtual Round Trip Time* (VRTT) in analogy with connection-based transport protocols like TCP.

Assumptions and notation

In this section, for the LRU analysis, we make the following assumptions:

- We consider a set of M different content items distributed in K classes of popularity ($m = M/K$ contents for each class) , i.e. content items of class k are requested with probability q_k, $k = 1, \ldots, K$. In the rest of the LRU analysis, we assume a Zipf popularity distribution, hence $q_k = c/k^\alpha$, with parameter $\alpha > 1$ and $1/c = \sum_{j=1}^{K} 1/j^\alpha$,

- Content items are segmented into packets and have different sizes: σ denotes the average content size in terms of number of equally sized packets;

- A node i in the network has a cache of size $x(i)$ packets.

N	Number of network nodes (or tree levels)
M	Number of content items ($m = M/K$ in each class k)
$x(i)$	Cache size at node (or tree level) i
$\lambda, \lambda(i)$	Total content request rate at first node, at node $i > 1$
$\mu(i)$	Miss rate at node $i > 1$
λ_k	Content request rate for class k w/o filtering
λ_k^f	Content request rate for class k with filtering
σ	Average content item size in number of packets
$q_k(i)$	Content popularity distribution of requests at node i
$p_k(i; x(1)..x(i)) \equiv p_k(i)$	Miss probability for class k at node i
$p_k^f(i; x(1)..x(i)) \equiv p_k^f(i)$	Miss probability with filtering for class k at node i
$R(i)$	Round trip delay between client and node i
$\text{VRTT}_k(x(1)..x(N)) \equiv \text{VRTT}_k$	Virtual round trip delay of class k w/o filtering
$\text{VRTT}_k^f(x(1)..x(N)) \equiv \text{VRTT}_k^f$	Virtual round trip delay of class k with filtering
$\text{RVRTT}_k(i; x(1)..x(i)) \equiv \text{RVRTT}_k(i)$	Residual VRTT_k at node i w/o filtering
$\text{RVRTT}_k^f(i; x(1)..x(i)) \equiv \text{RVRTT}_k^f(i)$	Residual VRTT_k at node i with filtering
Δ	Filtering time window
T_k	Content delivery time of class k
W	Number of parallel packet requests expressed by clients

Table 2.1: Notation

- Users use a receiver-driven transport protocol which requires the expression of an interest per packet and use a pipelining interest window of size W.

- We define virtual round trip time of class k, VRTT_k, the average time that elapses between the dispatch of a packet request and the packet reception in steady state. This variable plays a similar role to what the round trip time is for TCP connections in IP networks.

$$\text{VRTT}_k(x(1)..x(N)) = \sum_{i=1}^{N} R(i)(1 - p_k(i; x(1)..x(i)) \prod_{j=1}^{i-1} p_k(j; x(1)..x(j)) \tag{2.1}$$

with $k = 1, \ldots, K$. $\text{VRTT}_k(x(1)..x(i))$ is defined as a weighted sum of the round trip delays $R(i)$ to reach node i from the user, where weights correspond to the stationary hit probability $(1 - p_k(i; x(1)..x(i))$ to find a packet of class k at node i, given that a miss is originated by all previous nodes.

To simplify notation, for the rest of the section, we will omit the cache size $x(1)..x(i)$ dependence in the miss probability, Virtual Round Trip Time, etc. For example, we will write the miss probability at node i as $p_k(i) \equiv p_k(i; x(1)..x(i))$, and VRTT as $\text{VRTT}_k \equiv \text{VRTT}_k(x(1)..x(N))$.

Content requests process

The content request process is structured in two levels, *content* and *packet*. In this section we build a fluid model of such two levels request process that captures the first-order dynamics of the system in steady state. The main assumption behind is that lowest level nodes in Fig.2.1 aggregate requests issued by a large number of users.

The request arrival process is modeled through a Markov Modulated Rate Process (MMRP) [33] as depicted in Fig.2.2. Requests for content items in class k are generated according to a Poisson process of intensity $\lambda_k = \lambda q_k$, and the content to be requested is uniformly chosen among the m different content items in the given popularity class. A content item request coincides with the request of its first packet. Once a packet is received, a new packet request is emitted and so on until the reception of the last packet (in case of interest window $W = 1$). Notice that the model, also applies to the more general case of a window $W > 1$ packets requested in parallel.

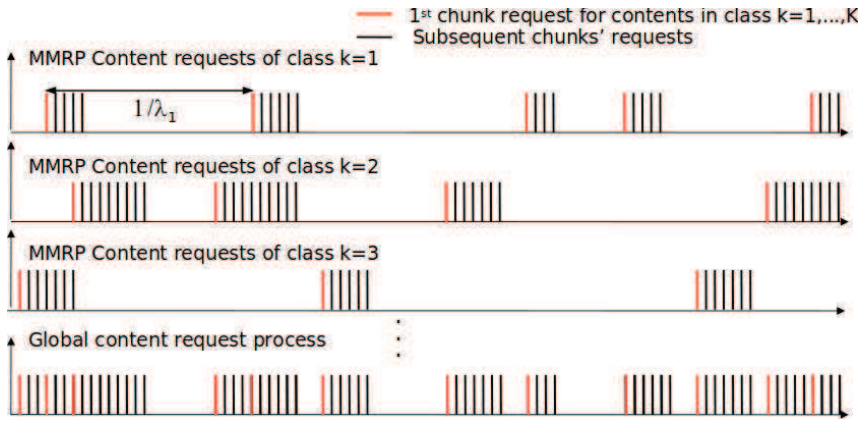


Figure 2.2: Single class content request process and global content request process.

Let us assume that the number of packets requested, N_{ch} and hence the size of content items is geometrically distributed with mean σ, i.e.

$$\mathbb{P}(N_{ch} = s) = \frac{1}{\sigma}\left(1 - \frac{1}{\sigma}\right)^{s-1}, \quad s = 1, 2, 3, ...$$

This means that we can never have zero packets in a content item request. The inter-arrival between two subsequent packet requests is deterministic and equal to the average Virtual Round Trip Time of class k, VRTT_k.

Observation 2.3.1. *Virtual Round Trip Times are not constant in practice as the hit/miss probabilities may vary over time. However, we look at the system in steady state under a stationary content*

request process where the average hit/miss probabilities and hence $VRTT_k$ have converged to a constant value.

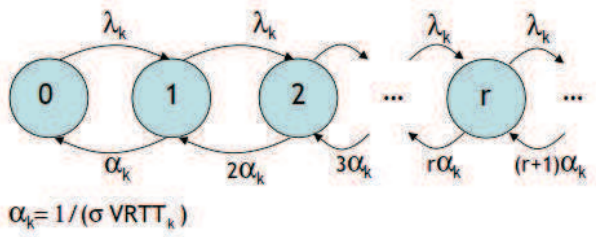


Figure 2.3: Superposition of content requests in class k.

The superposition of different content requests defines the MMRP process (Fig.2.2), whose underlying Markov chain that models the number of ongoing content download for a given class k (indicated with r) is represented in Fig.2.3. Note that besides the limited number of different content items per popularity class (m) there can be multiple ongoing requests for the same content item as the request process aggregates a large number of users' requests and thus r is in the interval $[0, \infty]$.

The choice of a MMRP model is natural within the context of packet-based content centric networks: in fact, *at content level* the Poisson assumption is motivated by previous work on Internet traffic modeling at session level, whereas it results to fail at packet level [34]. *At packet level* we suppose to look at system dynamics in *steady state* where the content store dynamics, have converged to a stationary value. The large number of content requests served by the aggregation network justifies the fluid assumption behind the MMRP [33]. It is worth to remark that we do not assume any other temporal correlation in the input process as in the IRM (Independence Reference Model), but we model the temporal correlation induced by content request aggregation (filtering). Such feature allows to keep trace of the ongoing requests at each node and avoid forwarding packet requests whether a request for the same packet has been already sent.

2.3.1 Single LRU cache

Let us now characterize the miss probability at steady state of class $k = 1, ..., K$ at the first cache, under the above described content request process. In [20], authors give an analytical characterization of the miss probability and hence of the miss rate for the case of a Poisson content request process when the cardinality of the set of different contents M tends to infinity. In this section we extend their result to the case of:

- a MMRP content request process (content/packet levels) with content request rate λ_k,

- K popularity classes partitioning with the same number of contents $m = M/K > 1$ in each one,
- content items of average size σ packets,
- pending request aggregation (filtering) over a time interval Δ.

Proposition 2.3.2. *Given a MMRP content request arrival process as described in previous section with intensity λ_k, popularity distribution $q_k = \frac{c}{k^\alpha}$, $\alpha > 1$, $c > 0$, and average content item size σ, be $x \equiv x(1) \equiv x > 0$ the cache size in number of packets, then the stationary miss probability for packets of class k, $p_k(1;x)$, is given by*

$$p_k(1) \equiv p_k(1;x) \sim e^{-\frac{\lambda}{m} q_k g x^\alpha} \tag{2.2}$$

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

Before evaluating the miss process of the cache, let us first give the intuition behind the proof. A request for a given packet generates a cache miss when more than x different packets are requested after its previous request, implying that the given packet has been removed from the cache before the arrival of the new request.

Thanks to the memory-less property of the Poisson process (content level) and of the geometric content size distribution (packet level), the number of packet requests for a given class k in the open interval (τ_{n-1}, τ_n) is independent from the number of packet requests in (τ_n, τ_{n+1}), where $\{\tau_1, \tau_2, ...\}$ is the sequence of points at which a miss in class k occurs. In addition, the distribution of the number of requests between two consecutive requests for the same packet can be computed from that of the number of requests among two subsequent requests for any two packets of class k. We define $r_{i(c_k)}(u,t)$ as the number of requests in the open interval (u,t) of a packet i of content c_k in class k, $r_{i(c_k)}(u,t) \sim$ *Poisson*$(\lambda q_k/m)$ and $B^{i(c_k)}(u,t) = \mathbb{1}_{\{r_{i(c_k)}(u,t)>0\}}$ the Bernoulli variable associated to the event that at least one packet i belonging to a content item in class k is requested in the open interval (u,t) with $\mathbb{P}[B^{i(c_k)}(u,t) = 1] = 1 - e^{-\frac{\lambda q_k}{m}(t-u)}$. $S(u,t)$ denotes the number of different packets requested over all popularity classes in (u,t),

$$S(u,t) = \sum_{k=1}^{K} \sum_{c_k=1}^{m} \sum_{i(c_k)=1}^{N_{ch}} B^{i(c_k)}(u,t) \tag{2.3}$$

In following lemmas we will also use $S^{i(c_k)}(u,t)$ defined as the number of different requested packets between two subsequent requests for packet i of content c_k, in class k in (u,t),

$$S^{i(c_k)}(u,t) = \sum_{k'=1}^{K} \left(\sum_{c'_k=1}^{m} \sum_{j(c'_k)=1}^{N_{ch}} B^{j(c'_k)}(u,t) \right) \tag{2.4}$$

We start by proving the following auxiliary results, Lemmas 2.3.3 and 2.3.4.

Lemma 2.3.3. *Under above assumptions, as $K \to \infty$, $t \to \infty$*

$$1/g \equiv \lim_{t\to\infty} \frac{\mathbb{E}[S(0,t)]^\alpha}{t} = \lambda c \sigma^\alpha m^{\alpha-1} \Gamma\left(1 - \frac{1}{\alpha}\right)^\alpha. \tag{2.5}$$

Proof. see Appendix A. □

Let also denote by $\{t_n^{i(c_k)}\}_{n>0}$ the time sequence of requests for the given packet i (of content c_k in class k) and $\{\tau_n^{i(c_k)}\}_{n>0}$ the inter-arrival time sequence, where $\tau_n^{i(c_k)} = t_{n+1}^{i(c_k)} - t_n^{i(c_k)}$. The following lemma characterizes the miss process for a given packet i (of content c_k) in class k and states the independence of $S^{i(c_k)}$ over time intervals among two misses of the given packet.

Lemma 2.3.4.

$$\mathbb{P}\left(\begin{array}{c} \textit{a miss on packet } i(c_k) \\ \textit{occurs at} \quad t_1^{i(c_k)}, ..., t_n^{i(c_k)} \end{array} \right) = \mathbb{P}[S^{i(c_k)}(0, \tau^{i(c_k)}) > x]^n \tag{2.6}$$

where $\tau^{i(c_k)}$ is exponentially distributed with parameter λ_k/m (the inter-arrival among two requests for the same packet within class k).

Proof. see Appendix A. □

The next proposition characterizes the stationary distribution of $S^{i(c_k)}$ among two miss events for packet $i(c_k)$, $k = 1, ..., K$.

Proposition 2.3.5. *If $q_k = \frac{c}{k^\alpha}$, $\alpha > 1$, $c > 0$, $t_n^{i(c_k)}$ is the n-th arrival time of a given packet $i(c_k)$ belonging to class k, then*

$$\lim_{x\to\infty} \mathbb{P}[S^{i(c_k)}(t_n^{i(c_k)}, t_{n+1}^{i(c_k)}) \geq x] \sim \mathbb{P}[\tau^{(k)} \geq g x^\alpha] \quad = e^{-\lambda q_k g x^\alpha} = e^{-\frac{m}{k^\alpha \Gamma(1-1/\alpha)^\alpha} \left(\frac{x}{\sigma m}\right)^\alpha} \quad x \to \infty. \tag{2.7}$$

with $1/g = \lambda c \sigma^\alpha m^{\alpha-1} \Gamma\left(1 - \frac{1}{\alpha}\right)^\alpha$, and $\tau^{(k)}$ denoting the inter-arrival among two subsequent requests of packets of class k.

Proof. see Appendix A. □

The main statement of Prop.2.3.2 follows from the next theorem which is based on Lemmas 2.3.3, 2.3.4 and Proposition 2.3.5, allows to conclude on the miss rate at the cache from the superposition of the miss sequences of each packet.

Theorem 2.3.6. *$\forall k = 1, ..., K$ let $i(1_k), i(2_k), ..., i(m_k)$ denote the i^{th} packet of content $1, 2, ..., m$ in class k. Given that all packets have the same size, each one can be expressed as a function of cache size x as $i(1_k) = \lfloor \delta x \rfloor$. Then, if we denote by $\mathcal{M}(t_l^{i(c_k)})$ the l^{th} event of a miss at time $t_l^{i(c_k)}$ for packet i of content c in class k we have that*

$$\mathbb{P}[\bigcup_{k=1,...,K, c=1,...,m} \left(\mathcal{M}(t_1^{i(c_k)}), \mathcal{M}(t_2^{i(c_k)}), ..., \mathcal{M}(t_l^{i(c_k)})\right)] = e^{-\lambda c g \frac{k_1}{\delta^\alpha}} e^{-\lambda c g \frac{k_2}{\delta^\alpha}} \cdots e^{-\lambda c g \frac{k_K}{\delta^\alpha}}, \tag{2.8}$$

that is the probability to have misses for the packet i of the content c in class k at ($t_1^{i(c_k)}$, $t_2^{i(c_k)}$, ..., $t_k^{i(c_k)}$) is equal to the product of the probabilities of having that miss sequence for one packet of each content in each class.

Proof. see Appendix A. □

As a consequence of the previous theorem, the miss sequences for contents in popularity classes of the order of x are asymptotically mutually independent. Thus the miss process at content level, which constitutes the input process for caches at second hop (level), has an intensity

$$\mu_k = \lambda_k \exp\left\{-\frac{\lambda}{m} q_k g x^\alpha\right\}, \qquad \forall k = 1, ..., K. \tag{2.9}$$

Observation 2.3.7. *The output process of the first node can be well approximated by a renewal process. Consider the sequence of packets inter-arrivals of a given class k. After a packet miss a request is sent upstream and is followed by a packet miss of the same class with probability $p_k(1)$ or by a hit with probability $1 - p_k(1)$. The next miss arrives after a geometric number h of hits with probability $(1 - p_k(1))^h p_k(1)$. The next miss can be approximated by a geometric sum of exponential independent inter-arrivals exponentially distributed of parameter λ_k with a support translated by gx^α which might be seen as a critical value that indicates that two consecutive misses cannot take place before gx^α.*

2.3.2 Content requests aggregation

In the CCN architecture, a packet request is issued and forwarded through the network until the given packet is found. A fundamental feature to avoid packet requests flooding is interest aggregation in CCN, which is a mechanism taking trace of pending packet requests at each node and preventing the dispatch of new requests for the same packet. This mechanism needs to be accounted when studying the interaction between transport and caching in such systems as it impacts both throughput and network caches behavior. Let us define RVRTT_k :

$$\text{RVRTT}_k(i) = \sum_{j=i}^{N} (R(j) - R(i-1))(1 - p_k(j)) \prod_{l=i}^{j-1} p_k(l). \tag{2.10}$$

RVRTT_k is the residual virtual round trip time of class k at hop i, that is the time that elapses on average until the reception at node i of the packet of data requested. Clearly, $\text{VRTT}_k = \text{RVRTT}_k(1)$.

Observation 2.3.8. *In steady state, the effective timescale aggregation for packet requests of content of class k is $\Delta_k(i) = \min(\Delta, \textit{RVRTT}_k(i))$, where Δ is the time window in which requests for the same packet are aggregated and hence not propagated.*

Indeed, one can reasonably assume Δ is taken larger than the virtual round trip time in order to effectively aggregate requests, but in practice the request aggregation is done until the on-the-fly request is satisfied and the data packet is received. From that time on, the packet is stored in cache, and whether removed by the replacement policy a new packet request is forwarded. The request aggregation or filtering clearly impacts the miss rate of a given cache. In fact, when a packet of class k request arrives at one cache, it can generate a *hit* if the packet is found in cache, otherwise it generates a *miss*. In the latter case, the request is filtered only if a previous request for the same packet have been emitted and the packet has not yet been received (e.g. the time elapsed by the previous packet request emitted is smaller than Δ_k). Let us now compute the filtering probability and thus the filtered miss rate at the lowest level of caches in Fig.2.1(b).

Proposition 2.3.9. *Given the content request process previously described, the filtering probability associated to class k at the first hop is,*

$$p_{filt,k}(1) = \frac{1 - b_k}{1 - (1 - 1/\sigma) b_k} \quad k = 1, ..., K \tag{2.11}$$

with $b_k = e^{-\Delta_k \lambda_k}$, and $\Delta_k = \min(\Delta, \textit{VRTT}_k)$.

Proof. see Appendix A. □

2.3.3 Network of LRU caches

In this section we apply the results of the single LRU cache model described above to the study of the topologies in Fig.2.1 and derive analytical expressions for the miss probabilities/rates at every hop (level). As in previous sections, we consider the case $m = M/K$ and $q_k = c/k^\alpha$, $\forall k = 1, ..., K$.

Notice that similarly to the single cache scenario, to simplify notation, we will omit the cache size dependence and, for example, write $p_k(i)$ instead of $p_k(i; x(1)..x(i))$ to indicate the local miss probability at level i. Moreover, we introduce here the following notation: $\mu(i)$ indicates the miss rate at a hop (level) i while $\lambda(i)$ indicates the request rate at content level for the hop (level) i. This means that for a line $\lambda(i+1) = \mu(i)$ and for a binary tree $\lambda(i+1) = 2\mu(i)$. We set $\mu(0) = \lambda(1) = \lambda$.

2.3.4 Miss rate characterization

In order to derive the stationary miss probabilities at hop i, with $i > 1$ for the topology under study, we need the following auxiliary result which extends Lemma 2.3.3 to hops after the first.

Lemma 2.3.10. *Given a MMRP content request process with rate $\lambda(i)$, miss probability $p_k(i)$ and popularity distribution $q_k(i) = \prod_{j=1}^{i-1} p_k(j) q_k / \sum_{l=1}^{K} \prod_{j=1}^{l-1} p_l(j) q_l, \quad k = 1, ..., K$, (where $q_k = c/k^\alpha$, $\alpha > 1$) as input at the content store at i^{th} (level) hop and defined $S_i(0,t)$ the number of different packets requested in the open interval $(0,t]$ at the i^{th} node, as $K \to \infty$, $t \to \infty$, it holds*

$$1/g(i) \equiv \lim_{t\to\infty} \frac{I\!E[S_i(0,t)]^\alpha}{t} = \frac{\lambda(i)}{\mu(i-1)} \lambda c \sigma^\alpha m^{\alpha-1} \Gamma\left(1 - \frac{1}{\alpha}\right)^\alpha \quad = \frac{\lambda(i)}{\mu(i-1)} 1/g. \tag{2.12}$$

Proof. see Appendix A. □

Let us now state the main result on the miss probabilities at hop $i > 1$ characterization for topology (a) in Fig.2.1 in absence of aggregation.

Proposition 2.3.11. *Given a cascade of N caches as in Fig.2.1(a), a MMRP content request process with rate $\lambda(i)$, miss probability $p_k(i; x(1)..x(i)) \equiv p_k(i)$ and popularity distribution $q_k(i) = \prod_{j=1}^{i-1} p_k(j) q_k / \sum_{l=1}^{K} \prod_{j=1}^{l-1} p_l(j) q_l, \quad k = 1, ..., K$, (where $q_k = c/k^\alpha$, $\alpha > 1$) as input*

at the content store at i^{th} (level) hop, then $\forall\ 1 < i \leq N$ it holds

$$\log p_k(i) = \prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)} \right)^{\alpha} p_k(l) \log p_k(1) \tag{2.13}$$

Proof. see Appendix A. □

Similarly, one can study the binary tree topology in Fig.2.1(b). where the only difference with respect to the case of a cascade is the input rate at upper level caches, $\lambda(i+1) = 2\mu(i)$.

Corollary 2.3.12. *Given a binary tree with $2^N - 1$ caches (that is with N levels) as in Fig.2.1(b), a MMRP content request process with rate $\lambda(i)$, miss probability $p_k(i; x(1)..x(i)) \equiv p_k(i)$ and popularity distribution*
$q_k(i) = \prod_{j=1}^{i-1} p_k(j) q_k / \sum_{l=1}^{K} \prod_{j=1}^{l-1} p_k(j) q_k(l), \quad k = 1, ..., K$, as input at the content store at i^{th} (level) hop, then $\forall\ 1 < i \leq N$ it holds

$$\log p_k(i) = \prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)} \right)^{\alpha} p_k(l) \log p_k(1) \tag{2.14}$$

Proof. see Appendix A. □

Let us now move to the case with request aggregation.

2.3.5 Miss rate characterization with request aggregation

For the topology in Fig. 2.1(a) the request aggregation takes place at first cache only. In fact, the effective timescale of aggregation for requests of class k at node i in the linear topology is $\Delta_k(i) = \min(\Delta, \text{RVRTT}_k(i))$, where $\text{RVRTT}_k(i)$ is defined in (2.10). This implies that once requests are aggregated at the first cache, in a topology with no exogenous request arrival after the first hop, the request process is not filtered anymore. In the rest of the section for the ease of notation we will omit the i in $\Delta_k(i)$.

Proposition 2.3.13. *Given a cascade of N caches as in Fig.2.1(a), a MMRP content request arrival process (defined by the miss rate at level i) as described in in the previous section and a timescale aggregation for content request Δ, then it holds*

$$p_k^f(i) \equiv p_k^f(i; x(1)..x(i)) = p_k^f(1)^{\prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)} \right)^{\alpha} p_k(l)} = p_k(i)^{1 - p_{filt,k}(1)}. \tag{2.15}$$

Proof. see Appendix A. □

For the topology in Fig.2.1(b) the request aggregation can take place at several hops depending on traffic and cache parameters.

Proposition 2.3.14. *Given a binary tree with 2^N-1 caches (that is with N levels) as in Fig.2.1(b), a MMRP content request arrival process as described in Sec.2.3.4 and a timescale aggregation for content request Δ, then it holds*

$$p_k^f(i) \equiv p_k^f(i; x(1)..x(i)) = p_k^f(1)^{\prod_{l=1}^{i-1} p_k^f(l)(1-p_{filt,k}(l))}, \tag{2.16}$$

$$p_{filt,k}(i) = 1 - \frac{b_k(i)(1-b_k(i)^\sigma)}{(1-b_k(i))\sigma}, \quad k = 1, ..., N,$$
$$b_k(1) = e^{-\Delta_k \lambda_k / m}, \quad b_k(i) = e^{-\Delta_k 2\mu_k^f(i-1)/m}, \quad i > 1,$$

$$\mu_k^f(1) = \lambda_k p_k(1)(1 - p_{filt,k}(1)),$$
$$\mu_k^f(i) = 2\mu_k^f(i-1) p_k^f(i)(1 - p_{filt,k}(i)), \quad i > 1. \tag{2.17}$$

Proof. The proof is a simple extension of the one for the previous theorem (Th.2.3.13) proof. □

2.3.6 LRU simulation results

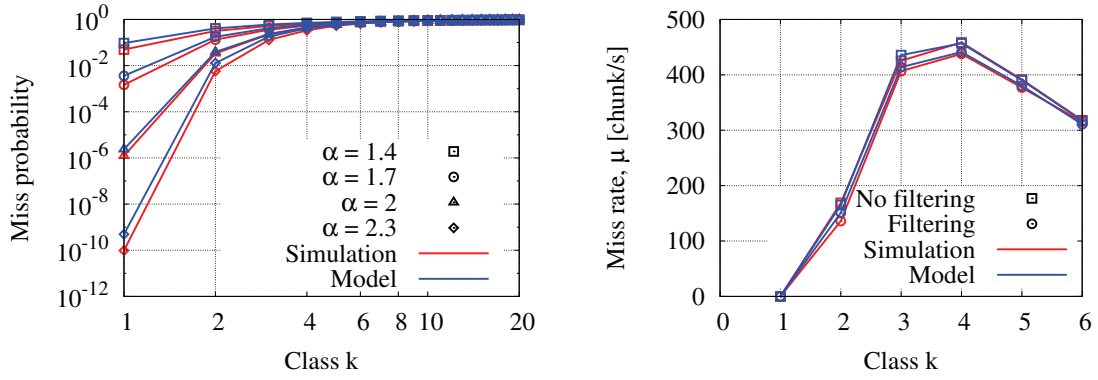
This section gathers numerical results obtained by means of packet-level simulations to corroborate theoretical results first in the case of a single cache and then in the case of a network of caches . To this purpose we developed CCNPL-sim, a packet level simulator whose details are specified in Chapter 9. Nodes' forwarding tables are computed according to a shortest path name based routing protocol and we assume that clients implement a simple transport protocol using a fixed window size for packet requests.

Through this section, unless otherwise specified, we consider a content items' catalog of $M = 20000$ content items, organized in $K = 400$ classes of decreasing popularity, each one with $m = 50$ items. Class popularity q_k is Zipf distributed and a given content item in class k is requested with probability q_k/m. We suppose content items are split in packets of $10kB$ each, and their size is geometrically distributed with average 690 packets, typical values for a UGC applications (see [35],[36]; [37]). Users generate content requests according to a Poisson process of intensity $\lambda = 40$ content items/sec, and the packet request window size is set to $W = 1$. We suppose a cache of size $x = 200000$ packets (2GBs) which implements the LRU replacement policy. Finally, network links have the same capacity $10Gbps$ and the same round trip delay equal

to $2ms$.

Notice that with our settings, $W = 1$, high bandwidth capacity equal to $10Gbps$, and propagation delay $1ms$, we can neglect the transmission delay so that $R(i) = PD_{up} + PD_{down} + t_{TX} \approx PD_{up} + PD_{down}$ where PD_{up}, PD_{down} are defined as constant up and down-link propagation delays and t_{TX} is the transmission delay.

Single LRU cache



(a) (b)

Figure 2.4: Single cache scenario. Asymptotic approximation of the miss probability as a function of popularity class (a) and miss rate with and without request filtering for $\alpha = 2$ (b) for LRU policy.

We analyze here some simulation results in order to validate the storage sharing model within a single cache.

In Fig. 2.4(a) we show the miss probability as a function of the popularity distribution for different values of the Zipf parameter α in absence of request aggregation. Fig. 2.4(b) reports the miss rate for $\alpha = 2$ with and without request filtering. Results are reported for the most popular classes (i.e. $k \leq 20$) for model and simulations, so confirming model accuracy in predicting miss probability/rate Eq.(2.2). The major discrepancy between model and simulations can be observed when the miss probability is very small, that is on the most popular classes for very skewed popularity. In such cases, cache misses are very rare events which are difficult to observe over a limited simulation time.

We notice that the miss probability is affected by content popularity for the 8 most popular classes; classes $k > 8$ count few requests for the considered α values and therefore result in miss probabilities close to 1. As expected, for most popular classes the miss probability increases as α

decreases. A smaller α parameter leads to a flatter popularity, which means an increasing number of different packets flowing trough the cache and eventually an higher miss probability. Notice also that the miss rate decreases when requests are aggregated, with a reduction of about 5% for classes 2 to 4.

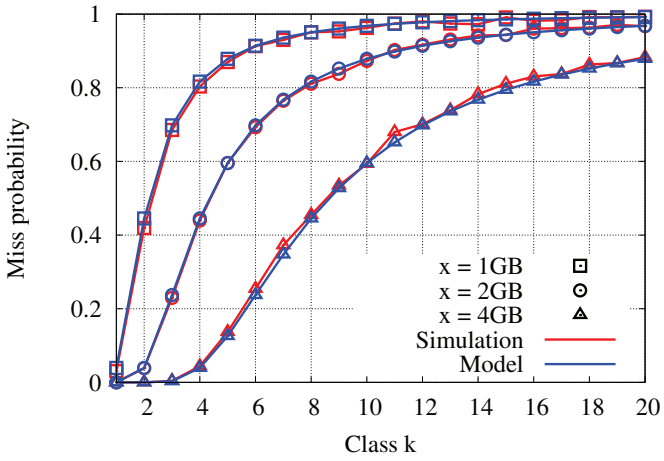


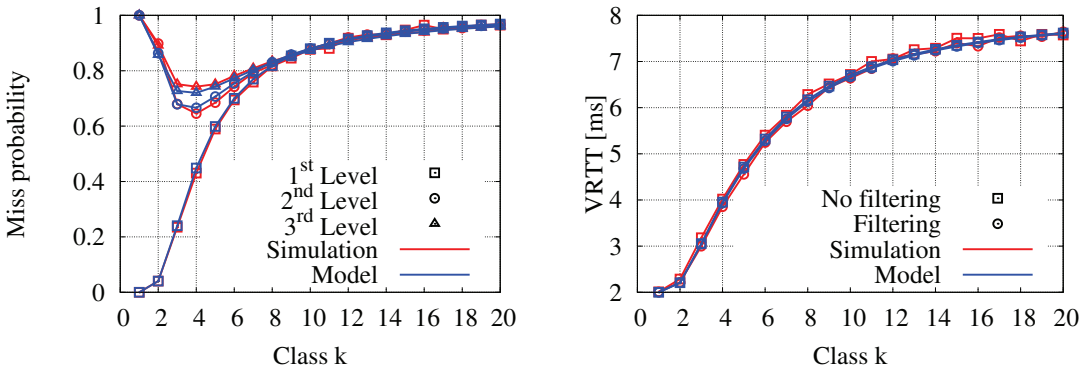
Figure 2.5: Single cache scenario. Asymptotic of the miss probability as a function of the popularity with different cache size, $\alpha = 2$ and LRU policy.

In Fig. 2.5 we illustrate the impact of the cache size on the miss probability for $\alpha = 2$. As expected, miss probability decreases as cache size increases because, for a given content size, the miss probability depends on the ratio between cache and content size.

Network of LRU caches

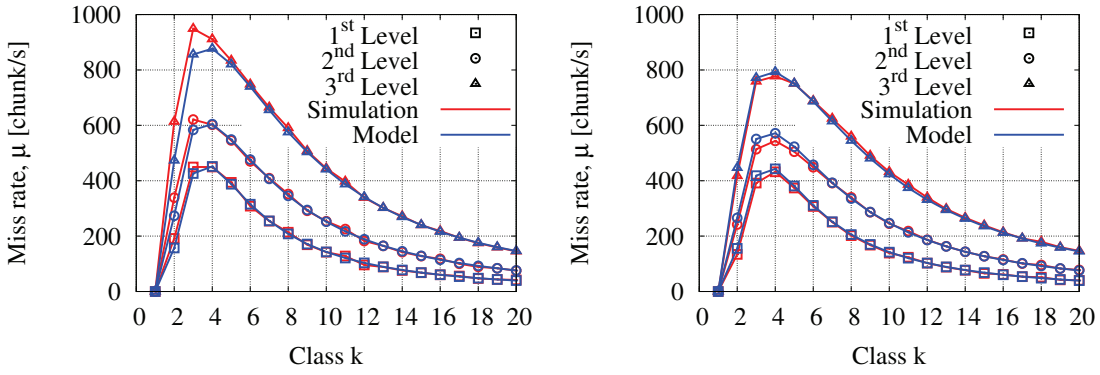
Let us now analyze the network case by means of packet-level simulations in order to assess model accuracy. Consider the topology reported in Fig.2.1(b), a *binary tree* with with $N = 4$ levels, where data are permanently stored at the root ($x(root) = \infty$) of the tree while leaf nodes are the entry points of user content requests. Notice that, this topology, is meant to represent a typical aggregation network collecting requests coming from different DSLAMs. The root of the tree serves as gateway for content items retrieved from the rest of the Internet. Content population characteristics are the same as in the single cache scenario explained above with Zipf parameter $\alpha = 2$.

Fig.2.6(a) compares the miss probabilities at different nodes (from the 1^{st} to the 3^{rd} level) without request aggregation. The comparison outlines the good match between model predictions Eq.(2.2) at level $i = 1$, Eq.(2.13) at level $i > 1$ and simulations. From Fig.2.6(a) it can be also



(a) (b)

Figure 2.6: Network of caches scenario. Asymptotic of the miss probability as a function of the popularity class with filtering disabled (a) and VRTT as a function of the popularity class experienced by end-users (b) in a binary tree topology with $\alpha = 2$ and LRU cache.



(a) (b)

Figure 2.7: Network of caches scenario. Miss rate as a function of the popularity class with filtering disabled (a) and enabled (b) in a binary tree topology with $\alpha = 2$ and LRU cache.

observed how content popularity changes along the path. Requests for content items of the most popular class are almost completely served by caches of first nodes. As a consequence, the miss probability for class $k = 1$ is nearly 1 at upper levels as its popularity becomes very low after the first cache. Content items of class 2 are mainly cached at first and second level, class $k = 4$ results to e the most popular one at level $i = 2, 3$, whereas less popular classes, represented in the queue of the curves, are very rarely cached as hardly requested.

Fig. 2.7(a),(b) reports simulation and model results of the miss rate as a function of content popularity with and without request aggregation. It is important to remark that the miss rate reduction with request aggregation, already observed in previous section for the single cache scenario, is more significant at higher levels of the network, with a maximum decrease of about 20% at level $i = 3$.

In Fig.2.6(b) we show the virtual round trip time, VRTT_k, as a function of content popularity with and without request aggregation in the binary tree topology for $\alpha = 2$. The virtual round trip time measures the average distance between the user and the content items as a function of the miss probabilities along the path. Therefore, it represents a suitable metric to evaluate data transfer performance, as it quantifies average packet delivery time.

As previously noticed, packets of the most popular content items are cached at first level nodes, so that $\text{VRTT}_1 \approx 2$ms, whereas the rarest items are not cached within the network, with a consequent round trip time of about 8ms (4 hops). The figure also highlights that the content aggregation has no or little impact on the stationary VRTT, even if it helps in strongly reducing the packet request traffic as also showed in Fig. 2.7(a),(b).

Convergence time

In this section we consider the convergence rate to the steady state of the simulations runs One of the most interesting metric to consider is $S_i(0,t)^\alpha/t$ that indicates the number of different packets per seconds requested in the open interval $(0,t)$ at level i.

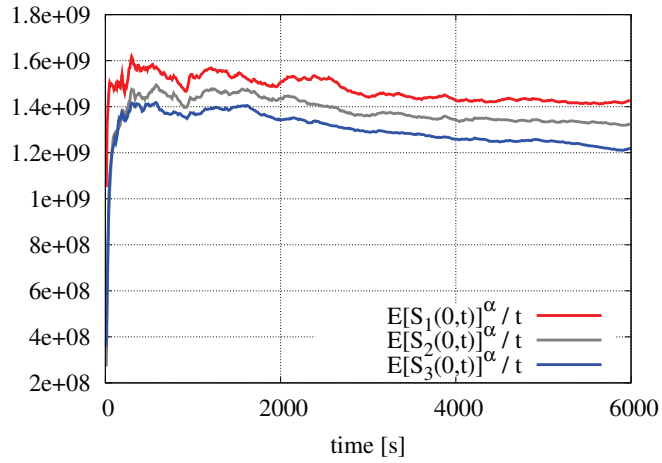


Figure 2.8: Network of caches scenario. Convergence evolution of $S_i(0,t)^\alpha/t$ in a binary tree topology with $\alpha = 2$ and LRU cache.

In Fig.2.8 we report $S_i(0,t)^\alpha/t$ as a function of time and of level i during a simulation run.

From the $S_i(0,t)^\alpha/t$ evolution, we notice that it maintains the same behavior for each level i: a fast increase in the first few seconds and then attains a stationary regime that lasts about 2000 seconds and finally a decreasing phase. This means that the most popular content items are requested during the first phase with little contributions from non popular classes. At a certain point more or less the same content items are periodically requested with little refresh because the number of different content items is finite during a simulation run. Such simulation limit, can be observed in the third regime (when $t > 2000$ seconds) in which $S_i(0,t)$ does not increase anymore as new content items are hardly requested. During this regime $S_i(0,t)^\alpha/t$ has an overall decreasing trend because $S_i(0,t)$ has no contribution from popular content items as they all have been requested at the end of the stationary phase.

2.4 Random replacement policy

Similarly to the model presented in the previous section for networks of LRU cache, in this section we address performance issues on networks of caches running the RND replacement policy.

In contrast to the previous section, in which we presented results for LRU caches, we here consider unit sized objects $\sigma = 1chunk$ and classes composed by one single object $m = 1$ so that the number of objects M of the catalog corresponds to the number of classes K.

2.4.1 Single cache model

We start from the single cache system with RND replacement policy, deriving analytic expressions of the miss probability together with asymptotes for large cache size.

Basic results

Consider a cache memory with finite size which is offered requests for objects. If the cache size is attained and a request for an object cannot be satisfied (corresponding to a *miss* event), the requested object is fetched from the repository server and cached locally at the expense of replacing some other object in the cache. The object replacement policy is assumed to follow the RND discipline, *i.e.*whenever a miss occurs, the object to be replaced is chosen at random, uniformly among the objects present in cache.

We first consider a discrete time system: at any time $t \in \mathbb{N}$, the t-th requested object requested from the cache is denoted by $C(t) \in \{1, 2, ..., M\}$, where M is the total number of objects which can be possibly requested (in the present analysis, the set of all possible documents is considered to be invariant in time). We assume that all M objects are ordered with decreasing popularity, the probability that object k is requested being q_k, $1 \leq k \leq M$. In the following, we consider

the *Independent Reference Model*, where variables $C(t)$, $t \in \mathbb{N}$, are mutually independent and identically distributed with common distribution defined by

$$\mathbb{P}(C = k) = q_k, \quad 1 \leq k \leq M.$$

Let $x(1) \equiv x \leq M$ be the cache capacity at the first node expressed in maximum number of objects stored and denote by $\mathcal{N}_x$ the set of all ordered subsets $\{j_1, ..., j_x\}$ with $1 \leq j_i \leq M$, $i \in \{1, ..., x\}$, and $j_i < j_\ell$ for $i < \ell$. Define the cache state at time $t \in \mathbb{N}$ by the vector $\mathbf{V}(t)$; $\mathbf{V}(t)$ may equal any configuration $\mathbf{v} = \{j_1, ..., j_x\} \in \mathcal{N}_x$. In the following, we let

$$G(x) = \sum_{\{j_1,...,j_x\} \in \mathcal{N}_x} q_{j_1} ... q_{j_x} \tag{2.18}$$

with $G(0) = 1$. Let $p(1; x)$ finally denote the stationary miss probability calculated at the first node $i = 1$ over all possibly requested objects with cache capacity $x(1) \equiv x$. As in the previous section, we will make an abuse of notation omitting $x(1)..x(i)$ dependence in the formula in order to simplify the notation i.e. miss probability at the first node will be $p(1) \equiv p(1; x)$.

It is shown [25] that the vector representing the cache configurations $(\mathbf{V}(t))_{t \in \mathbb{N}}$ define a reversible Markov process with stationary probability distribution given by

$$\mathbb{P}(\mathbf{V} = \mathbf{v}) = \frac{1}{G(x)} \prod_{j \in \mathbf{v}} q_j, \quad \mathbf{v} \in \mathcal{N}_x; \tag{2.19}$$

moreover ([25], Theorem 4), global miss probability equals

$$p(1) \equiv p(1; x) = \frac{\displaystyle\sum_{\{j_1,...,j_x\} \in \mathcal{N}_x} q_{j_1} ... q_{j_x} \sum_{k \notin \{j_1,...,j_x\}} q_k}{\displaystyle\sum_{\{j_1,...,j_x\} \in \mathcal{N}_x} q_{j_1} ... q_{j_x}}. \tag{2.20}$$

In [25], it is also shown that the cache configuration stationary probability distribution and miss rate probability are identical in the case of a FIFO cache, hence our results apply for the FIFO policy as well.

Expression (2.20) can be actually written in terms of normalizing constants $G(x)$ and $G(x+1)$ only; this will give formula (2.20) a compact form suitable for the derivation of both exact and asymptotic expressions for $p(1)$.

Lemma 2.4.1. *The miss probability $p(1)$ is given by*

$$p(1) \equiv p(1;x) = (x+1)\frac{G(x+1)}{G(x)} \tag{2.21}$$

with $G(x)$ defined in (2.18).

Proof. Any state $\mathbf{v} = \{j_1, ..., j_x\} \in \mathcal{N}_x$ corresponding to a unique sequence $1 \leq j_1 < j_2 < ... < j_x \leq M$ with $1 \leq j_k \leq M$, the denominator of (2.20) therefore equals $G(x)$. The numerator of (2.20) can in turn be expressed as

$$\begin{aligned}
&\sum_{1\leq j_1<...<j_x\leq M} q_{j_1}...q_{j_x} \sum_{k\notin\{j_1,...,j_x\}} q_k \\
&= \sum_{1\leq j_1<...<j_x\leq M} q_{j_1}...q_{j_x} \times \left(\sum_{1\leq k<j_1} q_k + ... + \sum_{j_{x-1}<k<j_x} q_k + \sum_{j_x<r\leq M} q_k \right) \\
&= (x+1) \sum_{1\leq k<j_1<...<j_x\leq M} q_k q_{j_1}...q_{j_x} \\
&= (x+1)G(x+1)
\end{aligned}$$

and expression (2.21) of $M(x)$ follows □

The latter results readily extend to the case when the total number M of objects is infinite, since the series $\Sigma_{j\geq 1} q_j$ is finite. The calculation of coefficients $G(x)$, $0 \leq x \leq M$, is now performed through their associated generating function F defined by

$$F(z) = \sum_{0\leq x\leq M} G(x) z^x, \quad z \in \mathbb{C},$$

for either finite or infinite population size M (as $p(1) \leq 1$ with cache $x(1) \equiv x$, Lemma 2.4.1 entails that $G(x+1)/G(x) \leq 1/(x+1)$ and the ratio test implies that the power series defining $F(z)$ has infinite convergence radius). We easily obtain the second preliminary result.

Lemma 2.4.2. *The generating function F is given by*

$$F(z) = \prod_{1\leq k\leq M} (1 + q_k z) \tag{2.22}$$

for all $z \in \mathbb{C}$.

Proof. Expanding the latter product and using definition (2.18) readily provide the result □

To further study the single cache properties, let $p_k(1)$ denote the per-object miss probability for a cache of size $x(1) \equiv x$, given that the requested object is precisely $k \in \{1, ..., N\}$. Defining

$$G_k(x) = \sum_{1 \leq j_1 < ... < j_x \leq M,\ k \notin \{j_1, ..., j_x\}} q_{j_1} ... q_{j_x} \tag{2.23}$$

with $G_k(0) = 1$, we then have $p_k = \mathbb{P}(k \notin \mathbf{V})$ so that (2.19) and (2.23) yield

$$p_k(1) \equiv p_k(1; x) = \frac{G_k(x)}{G(x)}. \tag{2.24}$$

Lemma 2.4.3. *The per-object miss probability $p_k(1; x) \equiv p_k(1)$ for given cache size $x(1) \equiv x$ and $k \in \{1, ..., M\}$ can be expressed as*

$$p_k(1) \equiv p_k(1; x) = 1 + \sum_{\ell=0}^{x-1} (-q_k)^{x-\ell} \frac{G(\ell)}{G(x)}. \tag{2.25}$$

The stationary probability $q_k(2)$ with $k \in \{1, ..., M\}$ and $x(1) \equiv x$, that a miss event occurs for object k is given by

$$q_k(2) = \frac{p_k(1)}{p(1)} q_k \tag{2.26}$$

where $p(1)$ is the miss probability averaged over the popularity classes.

Proof. By definition (2.23), the generating function $F_k(z)$ of coefficients $G_k(x)$, $0 \leq x \leq M$, is given by

$$F_k(z) = \frac{F(z)}{1 + q_k z}, \quad z \in \mathbb{C}. \tag{2.27}$$

Expanding the latter ratio as a powers series of z gives

$$G_k(x) = \sum_{\ell=0}^{x} (-q_k)^{x-\ell} G(\ell)$$

and provides (2.25) after using definition (2.24) for $p_k(1)$. Besides, letting $\mathcal{M}$ denote a miss event, Bayes formula entails

$$q_k(2) = \mathbb{P}(C = k \mid \mathcal{M}) = \mathbb{P}(C = k) \frac{\mathbb{P}(\mathcal{M} \mid C = k)}{\mathbb{P}(\mathcal{M})} = q_k \frac{p_k(1)}{p(1)}$$

hence relation (2.26) □

If the popularity distribution has unbounded support, then $\lim_{k \to +\infty} q_k = 0$ and formula (2.25) implies that the per-object probability p_k tends to 1 as $k \to +\infty$ for fixed x; (2.26) consequently

entails

$$q_k(2) \sim \frac{q_k}{p(1)}, \quad k \to +\infty. \tag{2.28}$$

For given x, asymptotic (2.28) shows that the tail of distribution $(q_k(2))_{k\in\mathbb{N}}$ at infinity is proportional to that of distribution $(q_k)_{k\in\mathbb{N}}$. The distribution $(q_k(2))_{k\in\mathbb{N}}$ describes the output process of the single cache generated by consecutive missed requests; it will serve as an essential ingredient to the further extension of the single cache model to network cache configurations considered in next sections.

First applications

Coefficients $G(x)$, $x \geq 0$, and associated miss probability $p(1)$ can be explicitly derived for some specific popularity distributions. In the following, the total population M of objects is always assumed to be infinite.

Corollary 2.4.4. *Assume a geometric popularity distribution $q_k = (1-j)j^{-1}$, $k \geq 1$, with given $j \in]0,1[$. For all $x(1) \equiv x \geq 0$, the miss rate equals*

$$p(1) \equiv p(1;x) = \frac{1-j}{1-j^{x+1}}(x+1)j^x. \tag{2.29}$$

Proof. Using Lemma 2.4.2, F is readily shown to verify the functional identity $F(z) = (1+(1-j)z)F(jz)$ for all $z \in \mathbb{C}$. Expanding each side of that identity in power series of z and identifying identical powers provides the value of the ratio $G(x+1)/G(x)$, hence result (2.29) by (2.21) $\square$

Let us now assume that the popularity distribution follows a Zipf distribution defined by

$$q_k = \frac{A}{k^\alpha}, \quad k \geq 1, \tag{2.30}$$

with exponent $\alpha > 1$ and normalization constant $A = 1/\zeta(\alpha)$, where ζ is the Riemann's Zeta function. We now show how explicit rational expressions for miss rate P can be obtained for some integer values of α.

Corollary 2.4.5. *Assume a Zipf popularity distribution with exponent α. For all $x(1) \equiv x \geq 0$,*

the miss probability equals

$$\begin{aligned} p(1) \equiv p(1;x) &= \frac{3}{2x+3} && \text{if } \alpha = 2, \\ &= \frac{45}{(4x+5)(4x+3)(2x+3)} && \text{if } \alpha = 4, \\ &= \frac{9!}{3!}\frac{(x+1)}{\prod_{4\le j\le 9}(6x+j)} && \text{if } \alpha = 6, \end{aligned}$$

Proof. When $\alpha = 2$, the normalization constant equals $A = 1/\zeta(2) = 6/\pi^2$. From the infinite product formula [38]

$$F(z) = \prod_{j\ge 1}\left(1 + \frac{u^2}{\pi^2 j^2}\right) = \frac{\sinh u}{u}$$

and expanding the left hand side into powers of $u^2 = Az\pi^2$ gives the expansion $F(z) = \Sigma_{x\ge 1}G(x)z^x$ where

$$G(x) = (\pi^2 A)^x/(2x+1)!, \quad x \ge 0.$$

Computing then ratio (2.21) with the above expression of $G(x)$ then provides $P = 3/(2x+3)$, as claimed. The cases when $\alpha = 4$ or $\alpha = 6$ follow a similar derivation pattern □

The formulas of Corollary 2.4.5 do not seem, however, to generalize for integer values $\alpha = 2p$ with $p \ge 4$; upper bounds can be envisaged and are the object of further study.

As also suggested by Corollary 2.4.5, the cache size corresponding to a target miss probability should be a decreasing function of α. This property is generalized in the next section where an asymptotic evaluation of $p(1)$ is provided for large cache size x and any Zipf popularity distribution with real exponent $\alpha > 1$.

Large cache approximation

The specific popularity distributions considered in Corollaries 2.4.4 and 2.4.5 show that $p(1)$ is of order xq_x for cache size x. In the present section, we derive general asymptotes for probabilities $p(1)$ and $p_k(1)$ for large cache size x and apply them to the Zipf popularity distribution.

We first start by formulating a general large deviations result (Theorem 2.4.6) for evaluating coefficients $G(x)$ for large x. To apply the latter theorem to the Zipf distribution (2.30), we then state two preliminary results (Lemmas 2.4.7 and 2.4.8) on the behavior of the corresponding generating function F at infinity. This finally enables us to claim our central result (Proposition 2.4.9) for the behavior of P for large x.

Theorem 2.4.6. *(i) Given the generating function F defined in (2.22), equation*

$$zF'(z) = xF(z) \tag{2.31}$$

has a unique real positive solution $z = \theta_x$ for any given $x \geq 0$;

(ii) Assume that there exists some constant $\sigma > 0$ such that the limit

$$\lim_{x \to +\infty} e^{v\sqrt{x}} \frac{F(\theta_x e^{-v/\sqrt{x}})}{F(\theta_x)} = e^{\sigma^2 v^2/2} \tag{2.32}$$

holds for any given $v \in \mathbb{C}$ with $\Re(v) = 0$ and that, given any $\delta > 0$, there exists $\eta \in]0, 1[$ and an integer x_δ such that

$$\sup_{\delta \leq |y| \leq \pi} \left| \frac{F(\theta_x e^{iy})}{F(\theta_x)} \right|^{1/x} \leq \eta \tag{2.33}$$

for $x \geq x_\delta$. We then have

$$G(x) \sim \frac{\exp(H_x)}{\sigma\sqrt{2\pi x}} \tag{2.34}$$

as x tends to infinity, with $H_x = \log F(\theta_x) - x \log \theta_x$.

Proof. See Appendix B □

Following Theorem 2.4.6, the asymptotic behavior of $p(1)$ can then be derived from (2.34) together with identity (2.21). This approach is now applied to the Zipf popularity distribution (2.30); in this aim, the behavior of the corresponding generating function F is first specified as follows.

Lemma 2.4.7. *For $\alpha > 1$ and large $z \in \mathbb{C} \setminus \mathbb{R}^-$, $\log F(z)$ expands as*

$$\log F(z) = \alpha(\nu_\alpha A z)^{1/\alpha} - \frac{1}{2}\log(Az) + V_\alpha + o(1) \tag{2.35}$$

with $A = 1/\zeta(\alpha)$ and V_α depending on α only and

$$\nu_\alpha = \left(\frac{\pi/\alpha}{\sin(\pi/\alpha)} \right)^\alpha, \tag{2.36}$$

Proof. See Appendix B □

Lemma 2.4.8. *For $\alpha > 1$ and large x, the unique real positive solution θ_x of equation (2.31) verifies*

$$\theta_x = \frac{x^\alpha}{A\nu_\alpha} + x^{\alpha-1} k_x \tag{2.37}$$

with $k_x = A_1 + O(x^{-1})$ *with* $A_1 = \alpha/2\nu_\alpha A$ *if* $\alpha \neq 2$ *and* $k_x = O(\log x)$ *if* $\alpha = 2$.

Proof. See Appendix B □

We can now state our central result.

Proposition 2.4.9. *For a Zipf popularity distribution with exponent* $\alpha > 1$*, the miss probability* $p(1)$ *for a cache of size* x *is asymptotic to*

$$p(1) \equiv p(1;x) \sim \nu_\alpha x q_x = \frac{A\nu_\alpha}{x^{\alpha-1}} \tag{2.38}$$

for large x*, with prefactor* ν_α *given in (2.36).*
Prefactor ν_α *verifies* $\lim_{\alpha\to+\infty} \nu_\alpha = 1$ *and* $\nu_\alpha \sim 1/(\alpha-1)$ *as* $\alpha \to 1$.

Proof. As verified in Appendix B, the conditions of Theorem 2.4.6 are satisfied for a Zipf distribution. Using asymptotes (2.35) and (2.37) of Lemmas 2.4.7 and 2.4.8 to explicit the argument H_x in (2.34) for large x, we then have

$$\begin{aligned} H_x &= \log F(\theta_x) - x\log\theta_x \\ &= \alpha(\nu_\alpha A\theta_x)^{1/\alpha} - \frac{1}{2}\log(A\theta_x) + V_\alpha + o(1) - x\log\left[\frac{x^\alpha}{A\nu_\alpha} + x^{\alpha-1}k_x\right] \end{aligned}$$

so that $H_{x+1} - H_x = -\alpha\log x + r + o(1)$ with constant $r = \log\nu_\alpha + \log A$. Coming back to definition (2.21) of $p(1)$, the latter estimates enable us to derive that

$$\begin{aligned} p(1) \equiv p(1;x) = (x+1)\frac{G(x+1)}{G(x)} &\sim x\exp[H_{x+1} - H_x] \\ &\sim x\frac{e^r}{x^\alpha} = \nu_\alpha\ x\ q_x \end{aligned} \tag{2.39}$$

as claimed □

Remark 2.4.1. *Proposition 2.4.9 provides asymptotic (2.38) for* $p(1)$ *under the weaker assumption that the popularity distribution* q_k*,* $k \geq 1$*, has a heavy tail of order* $k^{-\alpha}$ *for large* k *and some* $\alpha > 1$*, without being precisely Zipf as in (2.30). In fact, all necessary properties for deriving Lemmas 2.4.7 and 2.4.8 are based on that tail behavior only.*

To close this section, we now address the asymptotic behavior of p_k for a cache of size x defined in (2.24).

Proposition 2.4.10. *For any Zipf popularity distribution with exponent $\alpha > 1$ and given the object rank k, the per-object miss probability $p_k(1)$ is estimated by*

$$p_k(1) \equiv p_k(1;x) \sim \frac{\nu_\alpha k^\alpha}{x^\alpha + \nu_\alpha k^\alpha} \tag{2.40}$$

for large x, with prefactor ν_α defined in (2.36).

Proof. The generating function F_k of the sequence $G_k(x)$, $x \geq 0$, being given by (2.27), apply Theorem 2.4.6 to estimate coefficients $G_k(x)$ for large x. Concerning condition *(i)*, the solution $\eta = \eta_x$ to equation $\eta F_k'(\eta) = xF_k(\eta)$ reduces to equation (2.31) for $\theta = \theta_x$ where the term $q_k\theta/(1+q_k\theta)$ has been suppressed; but suppressing that term does not modify the estimate for θ_x with large x so that $\eta_x \sim \theta_x$. On the other hand, condition *(ii)* is readily verified by generating function F_k and we then obtain

$$G_k(x) \sim \frac{G(x)}{1+q_k\theta_x}. \tag{2.41}$$

By Lemma 2.4.8, we have $\theta_x \sim x^\alpha/A\nu_\alpha$ for large x; definition (2.24) of p_k and estimate (2.41) with $q_k = A/k^\alpha$ give

$$p_k(1) = \frac{G_k(x)}{G(x)} \sim \frac{1}{1+q_k\theta_x} \sim \frac{\nu_\alpha k^\alpha}{x^\alpha + \nu_\alpha k^\alpha} \tag{2.42}$$

and result (2.40) follows □

For any value $\alpha > 1$, (2.40) is consistent with the fact that p_k is an increasing function of object rank k and a decreasing function of cache size x.

2.4.2 Comparing RND to LRU

Let us now compare the latter results with the LRU replacement policy investigated in [18, 19]. Recall that, for a Zipf popularity distribution with exponent $\alpha > 1$, the miss probability for LRU of size $x(1) \equiv x$ is estimated by

$$p(1) \equiv p(1;x) \sim \xi_\alpha x q_x$$

for large x, with prefactor

$$\xi_\alpha = \frac{1}{\alpha}\left[\Gamma\left(1-\frac{1}{\alpha}\right)\right]^\alpha \tag{2.43}$$

where Γ is Gamma function ([19], Theorem 3). ξ_α is estimated by

$$\xi_\alpha \sim \frac{e^\gamma}{\alpha}, \quad \xi_\alpha \sim \frac{1}{\alpha-1}$$

as $\alpha \to +\infty$ and $\alpha \to 1$, respectively (γ is Euler's constant and $e^{\gamma} \approx 1,781...$). In view of Proposition 2.4.9, comparing asymptotes for coefficients ξ_α and ν_α shows that the difference $\nu_\alpha - \xi_\alpha$ tends to 1 as $\alpha \to +\infty$, thus illustrating that LRU discipline performs better than RND for large enough α (it can be formally shown that $\nu_\alpha > \xi_\alpha$ for all $\alpha > 1$). This difference diminishes, however, for smaller values of α since ν_α and ξ_α behave similarly as α is close to 1 (see Figure 2.9). Apart from that limited discrepancy, both disciplines provide essentially similar performance levels in terms of miss probabilities for large cache sizes.

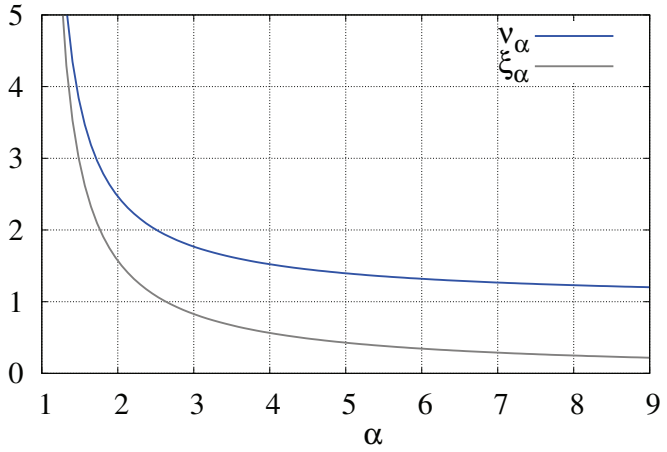


Figure 2.9: *Prefactors ν_α and ξ_α with $\alpha > 1$, for RND and LRU policies, respectively.*

In contrast to heavy-tailed popularity distributions, we can consider a light-tailed distribution where

$$q_k = A \exp(-Bk^{\beta}), \quad k \geq 1,$$

with positive parameters A, B, β. It is shown in this case [19] that the miss probability P for LRU cache of size x is asymptotic to

$$p(1) \equiv p(1; x) \sim \frac{e^{\gamma}}{\beta B} x^{1-\beta} q_x$$

for large x. For a geometric popularity distribution (with $\beta = 1$), the latter estimate shows that $p(1) = O(q_x)$; on the other hand, formula (2.29) of Corollary 2.4.4 shows that $p(1) = O(xq_x)$ for RND discipline. This illustrates the fact that RND and LRU replacements provide significantly different performance levels if the popularity distribution is highly concentrated on a relatively small number of objects.

2.4.3 RND Simulation results: single cache

We here present numerical and simulation results to validate the preceding estimates for a single RND policy cache. In the following, when considering a finite object population with total size M, the Zipf popularity distribution is normalized accordingly. We also mention that the content popularity distribution obviously refers to document classes instead of individual documents. For comparison purpose with the existing LRU analysis, we represent these classes by a single index, as if they were a single document. In the following, cache sizes must accordingly be scaled up to the typical class size.

Simulations are performed using the simulator described in chapter 9. In every simulation, performance measures are collected after the transient phase, once the system has reached the stationary state. Notice that, transient behavior is not considered and that that its duration obviously increases with the cache size.

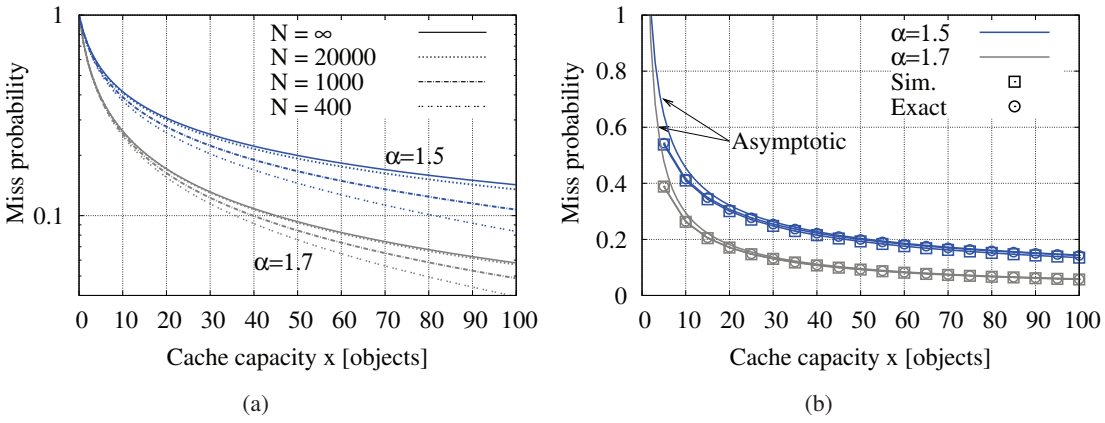


Figure 2.10: Single cache scenario. Exact (a) and asymptotic/exact comparison (b) of the global miss probability $p(1)$ as a function of the cache size, with $\alpha = 1.7$ for RND policy.

Besides, the most critical parameter in our simulation setting is the numerical value of α. As the Zipf distribution flattens when α get closer to 1, much longer simulation runs are necessary to have good estimates of the miss rate. Small enough values of α must, nevertheless, be considered as they are more realistic. Estimates of α are reported, in particular, in [36] for web sites providing access to video content like `www.metacafe.com` for which $\alpha = 1.43$, `www.dailymotion.com` and `www.veoh.com` for which $\alpha = 1.72$ and $\alpha = 1.76$, respectively. In the following, we hence fix $\alpha = 1.5$ or $\alpha = 1.7$ in our numerical experiments.

Fig. 2.10(a) first reports exact formula (2.20) for $p(1) \equiv p(1; x)$ with different cache x and for increasing values of total population M, where $p(1)^M$ measures the total miss probability for

a given cache of size x when the number of objects M is finite. As expected, the convergence speed of $p(1)^M$ to $p(1)^\infty$ as $M \to +\infty$ increases with α. In the case $\alpha = 1.5$ for instance, a population of $M = 20\ 000$ can be considered a good approximation for an infinite object population ($M = \infty$), whilst there is almost no difference between $p(1)^{20\,000}$ and $p(1)^\infty$ when $\alpha = 1.7$.

In Fig. 2.10(b), we compare exact formula (2.20) for P, asymptotic (2.38) and simulation results for the above scenario. Formula (2.20) for $N = \infty$ is computed with arbitrary precision and we used $M = 20\ 000$ for simulation as a good approximation for an infinite object population. Simulation and exact results are very close (especially for $\alpha = 1.7$) while asymptotic (2.38) gives a very good estimation of the miss probability as soon as $x \geq 20$.

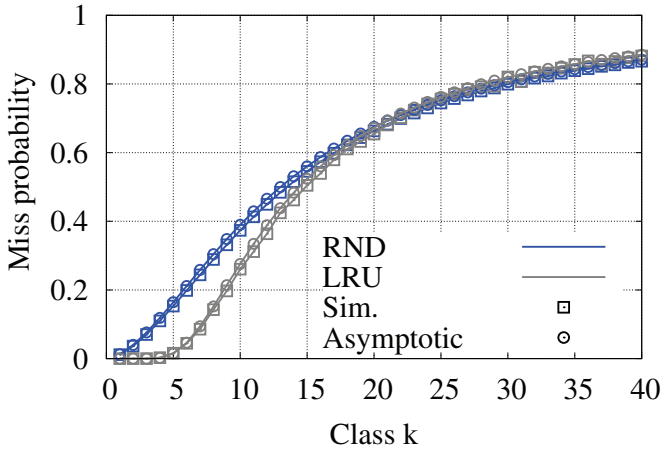


Figure 2.11: Single cache scenario. Asymptotic approximation of the miss probability $p_k(1)$ as a function of the class, with $x = 25$ objects, $\alpha = 1.7$ for RND and LRU policies.

Fig. 2.4.3 presents the miss probability $p_k(1)$ as a function of the object rank k for both RND and LRU policies with fixed $x = 25$ objects and $\alpha = 1.7$. Results are reported for the most popular classes and confirm the asymptotic accuracy of estimate (2.40) for RND and the corresponding one for LRU policy [20]. Beside the good approximation provided by the asymptotes, it is important to remark that RND and LRU performance are very close when object rank $k \geq 15$, while there is a slight difference for the most popular objects (say $k < 15$). Moreover, comparing the global miss probability $p(1)$ with a cache of $x = 25$ objects for RND and LRU (respectively equal to 0.147 and 0.108), we observe only 4% less of miss probability using LRU with respect to RND policy. This may suggest RND as a good candidate for caches working at very high speed, where LRU may become too expensive in terms of computation due to its relative complexity.

2.4.4 In-network cache model

In order to generalize the single-cache model, we will consider in this section networks of caches with various topologies.

Line topology

We first consider the system described in 2.3 for a linear topology Fig.2.1(a) in which a network of LRU caches use the In-Path Caching (IPC) schema. Notice that, also for the network of RND caches, we consider unit sized objects, one object per class and no request aggregation. Let now introduce some notation and properties for a tandem of caches.

Let $C_1(t) \in \{1, 2, ..., M\}$ denote the object requested at cache 1 at time t; we still assume that variables $C_1(t)$, $t \in \mathbb{N}$, describe an IRM process with distribution defined by $\mathbb{P}(C_1 = k) = q_k$, $1 \leq k \leq M$. Denoting by $\mathbf{V}_1(t)$ (resp. $\mathbf{V}_2(t)$) the state vector of cache 1 (resp. 2) at time t, the bivariate process $(\mathbf{V}_1(t), \mathbf{V}_2(t))_{t\in\mathbb{N}}$ is easily shown to define a Markov process that, however, is not reversible. It is therefore unlikely that an exact closed form for the stationary distribution of process $(\mathbf{V}_1, \mathbf{V}_2)$ can be derived to evaluate the miss probability for the two-cache system.

Alternatively, we here follow an approach based on the approximation of the request process to cache 2, similar to the one used in the previous section for the LRU cache modeling. Let t_n, $n \in \mathbb{N}$, denote the successive instants when a miss occurs at first cache 1, and $C_2(n)$ be the object corresponding to that miss event at time t_n. First note that the common distribution of variables $C_2(n)$ is the stationary distribution $(q_k(2))_{k\in\mathbb{N}}$ introduced in Lemma 2.4.3, equation (2.26), with cache size x replaced by $x(i = 1)$. In the following, we will further assume that the request process for cache 2 is an IRM, that is, all variables $C_2(n)$, $n \in \mathbb{N}$, are independent with common distribution:

$$\mathbb{P}(C_2 = k) = q_k(2), \quad k \in \mathbb{N}.$$

This simplifying assumption neglects any correlation structure for the output process of cache 1 (that is, the input to cache 2) produced by consecutive missed requests. Recall also that the tail of distribution $(q_k(2))_{k\in\mathbb{N}}$, defined by (2.28), is proportional to that of distribution $(q_k(1))_{k\in\mathbb{N}}$.

The latter 2-stage tandem model can be easily extended to a linear network consisting in a series of N caches ($N > 2$) where any request dismissed at caches $1, ..., i$, $i \geq 1$, is addressed to cache $(i + 1)$. As an immediate generalization of the IPC scheme, we assume that any requested document which experiences a miss at cache j, $1 \leq j \leq i$, and an object hit at cache $(i + 1)$ is copied backwards at all downstream caches $1, ..., i$. A request miss therefore corresponds to a miss event at each cache $1, 2, ..., j$. Furthermore, the assumption of IRM at caches 2, is generalized by

saying that any cache i considered in isolation behaves as a single cache with IRM input produced by consecutive missed requests at cache $(i-1)$. The size of cache i is denoted by $x(i)$.

In the following, the "global" miss probability $p_k^*(i; x(1)..x(i))$ (resp. "local" miss probability $p_k(i)$) for object's request of class k at cache i is the miss probability for object k over all caches $1, ..., i$ (resp. the miss probability for object k at cache i) so that

$$p_k^*(i; x(1)..x(i)) = \prod_{j=1}^{i} p_k(j; x(1)..x(j)) \tag{2.44}$$

(note that for a single cache, we have $p_k^*(1; x) = p_k(1; x)$). Notice that, as in the single RND cache modeling, we abusively write $p_k^*(i)$ (resp. $p_k(i)$) instead of $p_k^*(i; x(1)..x(i))$ (resp. instead of $p_k(i; x(1)..x(i))$), in order to simplify notation. Finally, if $q_k(i)$, $k \geq 1$, defines the distribution of the input process at cache i, the averaged local miss probability $p(i)$ at cache i is given by:

$$p(i) \equiv p(i; x(1)..x(i)) = \sum_{k \geq 1} p_k(i) q_k(i) \tag{2.45}$$

for any $i \in \{1, ..., N\}$.

Proposition 2.4.11. *For the N-caches tandem system with IPC scheme, suppose that the request process at cache 1 is IRM with Zipf popularity distribution with exponent $\alpha > 1$, and that IRM assumption holds for all caches $i \geq 2$. For any $i \in \{1, ..., N\}$ and large cache sizes $x(1), ..., x(i)$, the global miss probability $p_k^*(i)$ (resp. local miss probability $p_k(i)$) is given by*

$$\begin{aligned} p_k^*(i) \equiv p_k^*(i; x(1)..x(i)) &\sim \frac{\nu_\alpha k^\alpha}{\nu_\alpha k^\alpha + \sum_{j=1}^{i} x(j)^\alpha}, \\ p_k(i) \equiv p_k(i; x(1)..x(i)) &\sim \frac{\nu_\alpha k^\alpha + \sum_{j=1}^{i-1} x(j)^\alpha}{\nu_\alpha k^\alpha + \sum_{j=1}^{i} x(j)^\alpha}. \end{aligned} \tag{2.46}$$

Proof. Appendix B □

Proposition 2.4.11 shows how the N-stage tandem system with IPC scheme improves the performance in terms of miss probability by adding a term $x(j)^\alpha$ when the j-th cache is added to the path. From Propositions 2.4.11 and 2.4.9, it is readily derived that the average global miss

probability $p^*(i)$ for all objects requested along the cache network is

$$p^*(i) \equiv p^*(i; x(1)..x(i)) = \sum_{k \geq 1} p_k^*(i) q_k \sim \frac{A\nu_\alpha}{\left(\sum_{j=1}^{i} x(j)^\alpha \right)^{1-\frac{1}{\alpha}}} \tag{2.47}$$

for any $i \in \{1, ..., N\}$ and large cache sizes $x(1)$, ..., $x(i)$.

Tree topology

The previous linear network model can be easily extended to the homogeneous tree topology with Zipf distributed requests as in Fig. 2.1(b), used in the previous LRU modeling section (sec. 2.3).

We assume that all requests arrive at the leaves, following an IRM. Requests are served according to the IPC rule, *i.e.*, are forwarded upwards until the content is found, and the content is then copied in each cache between this location and the addressed leaf.

Corollary 2.4.12. *Consider a homogeneous tree with IPC scheme and suppose that assumption **(H)** holds for all its internal nodes. The results of Proposition 2.4.11 then extend to that tree with IRM request process at leaves and Zipf popularity distribution with exponent* $\alpha > 1$.

Proof. Only the order of requests in time matters since their precise timing is irrelevant; we can consequently assume that the requests arrive according to a Poisson process with intensity 1. Each leaf node, receive a Poisson request process of intensity q_k, with a Zipf popularity distribution $q_k = A/k^\alpha$, $r \geq 1$ Now, using IRM assumption mentioned for the linear topology and applying the previous results for a single cache to each leaf, we deduce that at any leaf, the miss sequence for object k is a Poisson process with intensity $q_k p_k(1)$. Merging these miss sequences from all children of a given second-level node, we deduce that the requests at this node follow a Poisson process and that the probability of request for an object k is $q_k p_k(1)/p(1) = q_k(2)$. This process has the same properties as the IRM process with distribution $(q_r(2))_{r \in \mathbb{N}}$ used in the proof of Proposition 2.4.11, which therefore applies. Repeating recursively this reasoning at each level, we conclude that Proposition 2.4.11 holds in this context □

Remark 2.4.2. *Corollary 2.4.12 is also valid for a homogeneous tree where different replacement policies are used at different levels i (e.g. Random at first level and LRU at second one).*

2.4.5 RND Simulation results: network of caches

We here report numerical and simulation results to show the accuracy of the approximations presented in previous Section.

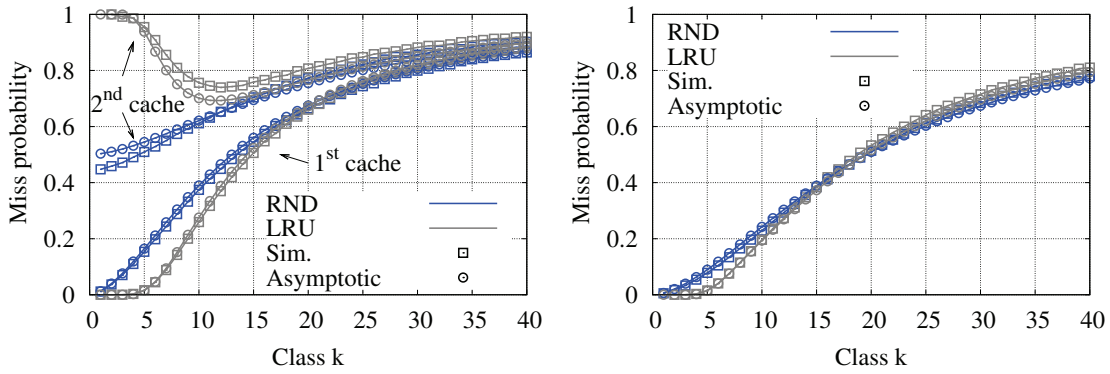


Figure 2.12: Network of caches scenario. Asymptotic approximation of the Miss probability for level 1,2 $p_k(1)$, $p_k(2)$ (a), and of the global miss probability $p_k^*(2)$ (b) as a function of the class, with $\alpha = 1.7$, $x(1) = x(2) = 25$ objects and RND or LRU caches.

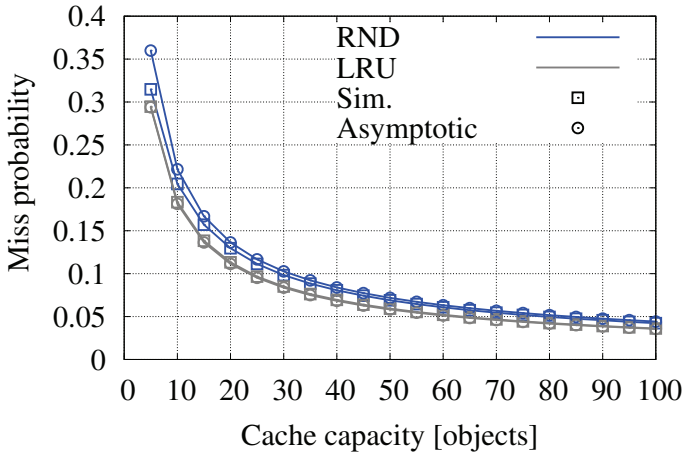


Figure 2.13: Network of caches scenario. Asymptotes of the global miss probability $p(2)$ as a function of the cache size, with $x(1) = x(2) \leq 100$, $\alpha = 1.7$ and RND or LRU caches.

Fig. 2.12(a) first reports estimate (2.46) of $p_k(1)$ and $p_k(2)$ for both RND and LRU (the approximation for the LRU tandem are taken from 2.3) with $x(1) = x(2) = 25$ objects and $\alpha = 1.7$. We focus on the second cache, as the performance of the first one has been largely analyzed in previous sections. We note a good agreement between the approximations and simulation results.

Furthermore, while less popular objects are affected in the same way when employing either RND or LRU (in our specific example, $k \geq 15$), a significantly different behavior is detectable

for popular objects ($k < 15$). Local miss probabilities $p_k(1)$ and $p_k(2)$ help understanding where an object has been cached, conditioned on its rank. The combination of LRU and IPC clearly tends to favor stationary configurations where popular objects are likely to be stored in the first cache. When using RND instead of LRU, however, the distribution of the content across the two caches is fairly different; in Fig. 2.12(a) for example, while the most popular objects are likely to be retrieved at the first cache when using either LRU or RND, only by using RND can such an object be also found in the second cache. It therefore appears that while both LRU and RND tend to store objects proportionally to their popularity, RND more evenly distributes objects across the whole path.

Fig. 2.12(b) reports the total miss probability at the second cache $p_k^*(2) = p_k(1)p_k(2)$, *i.e.*, the probability to query an object of rank r at the repository server. In this example, we see that objects with rank $k < 15$ are slightly more frequently requested at the server when using RND rather than LRU, but RND is more favorable than LRU for objects with higher rank $k \geq 15$. In average, the total miss probability at the second cache $p^*(2)$, reported in Fig. 2.4.5, is very similar either using RND or LRU, with a slight advantage to LRU. $p^*(2)$ indicates the amount of data that will be requested to the server.

2.5 Mixture of RND and LRU

So far, we have considered networks of caches where all caches use the RND ofthe LRU replacement policy. In practice, it is feasible to use different replacement algorithms in the same network. This section addresses the case of a tandem network, where one cache uses the RND replacement algorithm while the other uses the LRU algorithm. As in previous sections, these results also hold in the case of an homogeneous tree in which RND and LRU are used alternatively at different network level i.

2.5.1 Large cache size estimates

We first provide estimates for miss probabilities in the case of tandem network, when cache sizes $x(1)$ and $x(2)$ are large.

Proposition 2.5.1. *For the* 2*-caches tandem system with IPC scheme, suppose the request process at cache* 1 *is IRM with Zipf popularity distribution with shape parameter* $\alpha > 1$ *and that IRM assumption for the request process at cache* 2 *holds.*

I) *When cache* 1 *(resp. cache* 2*) uses the RND (resp. LRU) replacement policy, the local (resp.*

global) miss probability $p_k(2)$ (resp. $p_k^*(2)$) at cache 2 is given by:

$$p_k(2) \sim \exp\left(-\frac{\nu_\alpha x(2)^\alpha}{\alpha\xi_\alpha\left(\nu_\alpha k^\alpha + x(1)^\alpha\right)}\right)$$
$$p_k^*(2) \sim \frac{\nu_\alpha r^\alpha}{\nu_\alpha k^\alpha + x(1)^\alpha}\exp\left(-\frac{\nu_\alpha x(2)^\alpha}{\alpha\xi_\alpha\left(\nu_\alpha k^\alpha + x(1)^\alpha\right)}\right), \tag{2.48}$$

for large cache sizes $x(1)$, $x(2)$ and constants ν_α, ξ_α introduced in (2.36) and (2.43).

***II)** When cache 1 (resp. cache 2) uses the LRU (resp. RND) replacement policy, the local (resp.global) miss probability $p_k(2)$ (resp. $p_k^*(2)$) at cache 2 is given by:*

$$p_k(2) \sim \frac{\nu_\alpha k^\alpha}{\nu_\alpha k^\alpha + x(2)^\alpha \exp\left(-\frac{x(1)^\alpha}{\alpha\xi_\alpha k^\alpha}\right)}$$
$$p_k^*(2) \sim \frac{\nu_\alpha k^\alpha}{\nu_\alpha k^\alpha \exp\left(\frac{x(1)^\alpha}{\alpha\xi_\alpha k^\alpha}\right) + x(2)^\alpha}, \tag{2.49}$$

for large cache sizes $x(1)$, $x(2)$.

Proof. See Appendix B. □

2.5.2 Mixed RND-LRU Simulation results

We here report numerical and simulation results for mixed homogeneous tree topologies to show the accuracy of the approximations presented in previous section, in order to derive some more general considerations about the mixture of RND and LRU in a network of caches. According to remark 2.4.2, we simulate in fact tree topologies, with 2 leaves with cache size $x(1)$ and one root with cache size $x(2)$.

Fig. 2.14(a) reports $p_k(2)$ for RND-LRU and LRU-RND homogeneous tree networks with asymptotes (2.48) and (2.49), respectively. We first note that the latter provide estimates with reasonable accuracy. Besides, we observe that the behavior of $p_k(2)$ is in strict relation to the policy used for cache 1. If cache 1 is RND then, $p_k(2)$ has a behavior similar to that observed in RND caches tandem; similarly, if cache 1 is LRU, $p_k(2)$ behaves as in the case of LRU caches in tandem.

This phenomenon has a natural explanation. RND and LRU act in the same way on objects ranked in the tail of the Zipf popularity distribution. However, the two replacement policies manage popular objects in a rather different way, as we already observed in Section 2.4.5. The second level caches see a local popularity that is shaped, by the first level of caches, in the body of the

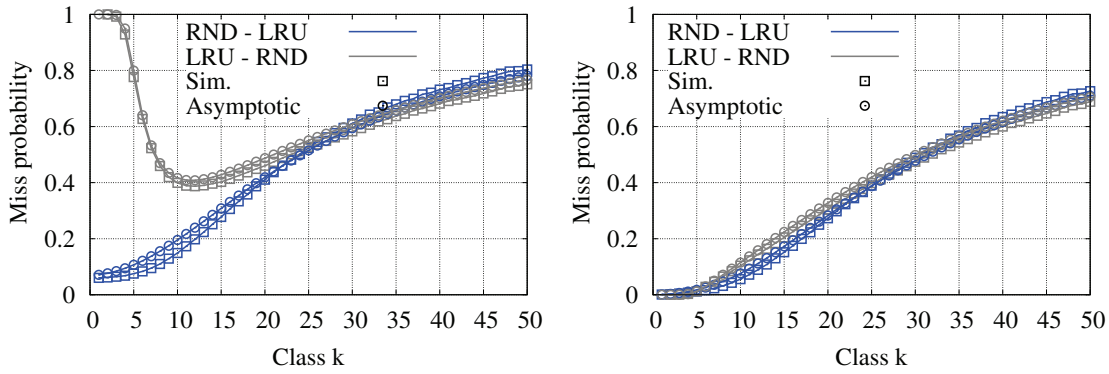


Figure 2.14: Mixed homogeneous tree scenario. Asymptotic approximation of the miss probability $p_k(2)$ (a) and global miss probability $p_k^*(2)$ (b) as a function of the popularity class for level 2 with $x(1) = 25objects, x(2) = 50objects$ and $\alpha = 1.7$.

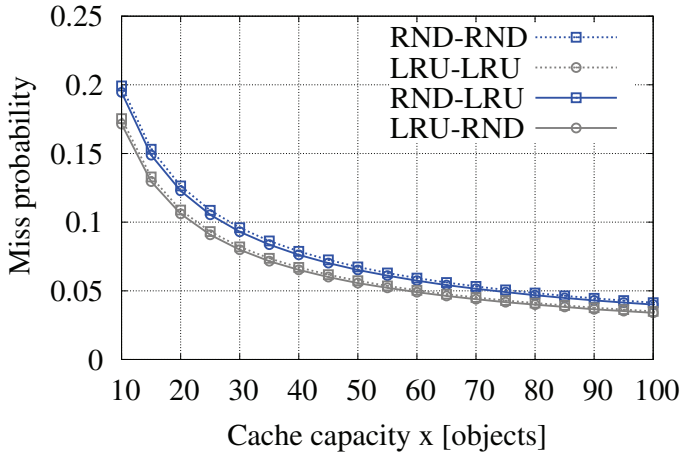


Figure 2.15: Mixed homogeneous tree scenario. Simulation results for the global miss probability $p^*(2)$ as a function of the cache size, with $x(1) = x(2) \leq 100$ and $\alpha = 1.7$.

probability distribution. The portion of the distribution that is affected by such shaping process is determined by the cache size $x(1)$ at first level. In the analysis reported in Figs. 2.14,2.15, the first level cache significantly determines the performance of the overall tandem system. In Fig. 2.14(b), we observe that the total miss probability $p_k^*(2)$ in the two mixed tandem caches is similar, while the distribution of the objects across the two nodes varies considerably.

Finally, in Fig. 2.5.2 we present simulation results of $p^*(2) = p^*(2; x(1), x(2))$ for all possible

configurations in an homogeneous tree network in which $x(1) = x(2) \leq 100$. Observe that LRU-RND tree cache network achieve slightly better performance than the LRU-LRU system at least with Zipf shape parameter $\alpha = 1.7$. This behavior would suggest to prefer LRU at the first cache because it performs better in terms of Miss probability and RND at the second one in order to save significant processing time.

2.6 Conclusion

In this chapter, we studied the LRU and the RND replacement policies in a network of caches, in order to capture storage sharing performance in the CCN architecture. Assuming that the content popularity is represented by a Zipf law with parameter $\alpha > 1$, that content requests follows the IRM, and neglecting the bandwidth sharing, we provided analytical models for the performance evaluation of the two replacement policies, explicitly characterizing steady state dynamics. Closed form expressions for the stationary miss probability are provided and compared against packet level simulations, showing the dependence from key system parameters such as content popularity and cache size. Moreover, for the LRU policy, we also modeled the request aggregation feature that allows to aggregate identical packet requests, slightly modifying the miss rate process form a cache. Finally, we compared LRU and RND cache performance and also considered the case of mixed policies (RND at one network level and LRU at the other one or vice-versa).

Our results suggest that the RND and LRU performance is reasonably close, especially for cache employed deep in the network (i.e. close to the repository). As a consequence, RND is a good candidate for high-speed caching when the complexity of the replacement policy becomes critical. In the presence of a hierarchy of caches, caches close to receivers (i.e. access networks) typically serve a relatively small number of requests per second which can be easily sustained by a cache running LRU. For this reason, the LRU policy should consequently be implemented at the bottom level since it provides the best performance. Meanwhile, higher-level caches see many aggregated requests and should therefore use the RND policy which yields similar performance (as to second-level cache) while being less computationally expansive.

Chapter 3

Bandwidth sharing modeling

In order to completely characterize CCN and more generally network of caches' performance, in this chapter, we will consider the interaction between storage and bandwidth sharing.

In CCN, a receiver can download data from multiple nodes and share bandwidth with other concurrent transfers. User-perceived performance, critically depends on the way bandwidth is shared between parallel downloads. The notion of fairness has been deeply studied in the literature. Nowadays, a generally accepted fairness objective is max-min. Max-min fair allocation consists in equally allocate the available bandwidth by infinitesimal increments until one flow is satisfied. Amongst the remainder of the flows, the available bandwidth is again equally divided until one flow is satisfied and so on, until the bandwidth is completely assigned or the flows are completely satisfied. Max-min has been extensively studied in the literature and it can be realized either trough flow control at the receiver or by fair-queuing scheduling in network links (cfr.[39],[40], [41]).

3.1 Model description

In the previous chapter, we computed the probability to find a given packet in the cache of a given node i through the miss probabilities $p_k(i)$, $i = 1, ..., N$ for RND and LRU network of caches under certain request properties. In this section, we will only consider LRU even if, with proper extensions, we can also apply the RND performance analysis to the bandwidth sharing model. Once founded in a cache in the network, data packets are sent back to the end-user following the reverse path of packet requests and stored using the In-Path Caching principle. This implies that a request satisfied by node i triggers a corresponding data transmission to the users over the route that we denote with i, from node i to the end-user (blue arrows in Fig.2.1). We make an abuse of notation by tagging every route with the same index i used for nodes, by meaning that route i is the set of links $l \ni i$ from the user to the i-th node along the path that goes up to the repository.

Conversely $i \ni l$ indicates all routes traversing through link l.

Every route i is characterized by a number of parallel transfers, N_i, sharing the same route and therefore the limited bandwidth. N_i is a random process that varies according to the demand λ and the service rate each transfer in progress gets. N_i is a birth and death Markov process, with birth rate $\mu(i-1) - \mu(i)$, with $\mu(0) \equiv \lambda$, and death rate $n_i\gamma_i/(\sigma P)$ when n_i parallel transfers are in progress. Recall that $\mu(i-1) - \mu(i)$, in case of network of LRU caches that we are considering, denotes the rate of the Poisson process of content requests satisfied by node i, under the assumption of a MMRP miss process of intensity $\mu(i)$ at node i. The death rate, $n_i\gamma_i/(\sigma P)$, is determined by the max-min fair shared bandwidth γ_i in bps associated to each of the n_i content transfers in parallel over route i. Indeed, in this section, we assume that bandwidth is fairly shared among each transfer in the max-min sense. Let us recall the definition of the max-min fairnes. At a given link l, for the bottlenecked flows the max-min fair share γ_l is equal to the fair share rate f_l defined by

$$f_l = \frac{C_l - \sum_{j \in N_l^{nb}} \gamma_{j,l}}{N_l - N_l^{nb}}, \tag{3.1}$$

where N_l is the number of flows traversing the link l and $\sum_{j \in N_l^{nb}} \gamma_{j,l}$, N_l^{nb} are respectively the sum of rates and the number of flows whose demand is lower than the fair rate (not bottlenecked flows at link l).

In our case, considering the capacity constraint,

$$\sum_{i \ni l} n_i\gamma_i \leq C_l, \qquad \text{for all links } l, \tag{3.2}$$

there exists at least one bottleneck link $l \ni i$ for every route i serving such transfers at the maximum rate, i.e. $\exists l \ni i$ such that

$$\sum_{i \ni l} n_i\gamma_i = C_l \qquad \text{and} \qquad \gamma_i = \bigvee_{i' \ni l} \gamma_{i'} \tag{3.3}$$

where $\bigvee$ denotes the maximum operator. Such conditions uniquely define the max-min share γ_i related to route i. Notice that differently from traditional networks, in CCN a single content download can encounter more than one bottleneck in a single path. This is due to the presence of in-network caches, that can directly reply to interests without using upstream bandwidth resources.

3.2 Average content delivery time

Let us now apply bandwidth and LRU storage models to determine average content delivery time. As illustrated in Fig.2.1, down-link $i > 1$ is shared among all routes j, with $j > i$. By definition, the traffic load at link i is:

$$\rho_i \equiv \frac{r_i}{C_i}, \tag{3.4}$$

where r_i is the average data rate that traverses the link i. At packet level, the data rate is given by the expression $\sigma \sum_{j=i}^{N}(\mu(j-1) - \mu(j))$, which accounts for all data packets flowing from upper caches (up to the repository) down to the receiver. The sum representing the data rate is a telescoping series that simplifies to $\sigma(\mu(i-1) - \mu(N))$ where $\mu(N) = 0$ (as the node N represents the content repository). If we substitute r_i in the traffic load (ρ_i) formula at link i we obtain,

$$\rho_i \equiv \frac{\sigma P}{C_i} \sum_{j=i}^{N}(\mu(j-1) - \mu(j)) = \frac{\sigma P}{C_i}\mu(i-1), \tag{3.5}$$

Clearly, ρ_i must satisfy the stability condition $\rho_i < 1 \; \forall i = 1, \ldots, N$ to keep $N_j, \forall j \ni i$ finite. With no loss of generality, we assume that all the significant source of delay is generated at the bottleneck link encountered along the path and consider any constant propagation delay negligible either in the upstream and downstream. Therefore, the average delay between user and node i R_i,

$$R_i = PD_{up} + PD_{down} + P/\gamma_i \approx P/\gamma_i, \tag{3.6}$$

where P/γ_i is the transmission time (t_{TX}) of a single packet for the route i. plus the transmission delay of data packets along route i.

Proposition 3.2.1. *If $\rho_i < 1$, $\forall i \geq 1$, the average delivery time for content items in popularity class k, $\forall k = 1, ..., K$ is upper bounded by*

$$\mathbb{E}[T_k] \leq \sum_{i=1}^{N} \bigvee_{j \ni i} \frac{\sigma P}{C_j(1-\rho_j)}(1 - p_k(i)) \prod_{j=1}^{i-1} p_k(j), \tag{3.7}$$

Proof. Without bandwidth limitation, $\mathbf{E}[T_k] = \sigma \text{VRTT}_k / W$ where VRTT is the virtual round trip time defined as

$$\text{VRTT}_k = \sum_{i=1}^{N} R_i(1 - p_k(i)) \prod_{j=1}^{i-1} p_k(j). \tag{3.8}$$

In such case, the choice of a window of interests $W > 1$ makes a fundamental difference, as the system is not limited in bandwidth and $E[T_k]$ reduced of a factor W. In the case of a bandwidth-limited downstream path, the max-min fair share is determined by the number of transfers in progress (which does not vary with W) and by the link capacities so that Eqs.(3.2-3.3) are satisfied. Under these conditions, allowing for a higher W does not enhance content delivery performance as the receiver is already exploiting all the available bandwidth. As a consequence, $\mathbb{E}[T_k] = \sigma\text{VRTT}_k$ as if $W = 1$ (in the fluid approximation), with $R_i \approx P/\gamma_i$ as above explained. γ_i can be easily computed using the processor sharing model for parallel downloads assuming equal share of available bandwidth (see [39]). The average per data packet delay is then given by P/γ_i with

$$\gamma_i \geq \bigwedge_{j \ni i} C_j(1 - \rho_j) \tag{3.9}$$

where $\bigwedge$ denotes the minimum operator. Such allocation verifies conditions (3.2),(3.3) when all flow are bottlenecked on the same link, otherwise (3.9) is a lower bound. The statement follows by invoking Eqs.(3.6) and (3.9) in Eq.(2.1).

□

3.3 Simulation results

To support analytical findings and to assess model accuracy, we used the event driven simulator described in Chapter 9, using forwarding queues with Deficit Round Robin scheduler [42] in order to impose max-min fairness among parallel transfers.

Let us first consider the linear topology in Fig.2.1(a) with $N = 3$, where content requests are forwarded through intermediate LRU caches before reaching the content repository. By varying link capacities, four configurations can be distinguished, depending on the bottleneck link which determines max-min fair shares for routes $1, 2, 3$:

Case I) $C_1(1 - \rho_1) < C_2(1 - \rho_2), C_1(1 - \rho_1) < C_3(1 - \rho_3)$.
In this case, eq. (3.7) reduces to

$$\mathbb{E}[T_k] = \frac{\sigma P}{C_1(1 - \rho_1)},$$

In Case I the three available routes are bottlenecked at the first link and hence the delivery time is constant and does not depend on the popularity.

Case II) $C_2(1-\rho_2) < C_1(1-\rho_1), C_2(1-\rho_2) < C_3(1-\rho_3)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)) + \frac{\sigma P}{C_2(1-\rho_2)}p_k(1).$$

In Case II, there exist two different bottleneck in the network. In fact, route 1 is bottlenecked at the first link as $C_2(1-\rho_2) < C_1(1-\rho_1)$. On the other side, route $2,3$ are bottlenecked at the second link as $C_2(1-\rho_2) < C_3(1-\rho_3)$. Notice that here, most popular content downloads that are more likely retrieved from route 1 will experience lower delivery time.

Case III) $C_3(1-\rho_3) < C_2(1-\rho_2) < C_1(1-\rho_1)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)) \quad + \frac{\sigma P}{C_2(1-\rho_2)}p_k(1)(1-p_k(2)) + \frac{\sigma P}{C_3(1-\rho_3)}p_k(1)p_k(2)$$

In case III the routes $1,2,3$ are bottlenecked at three different links as $C_3(1-\rho_3) < C_2(1-\rho_2) < C_1(1-\rho_1)$ and delivery time performance are strictly copupled with the content location (that depends on its popularity).

Case IV) $C_3(1-\rho_3) < C_1(1-\rho_1) < C_2(1-\rho_2)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)p_k(2)) + \frac{\sigma P}{C_3(1-\rho_3)}p_k(1)p_k(2).$$

In case IV route $1,2$ are bottlenecked at the first link as $C_1(1-\rho_1) < C_2(1-\rho_2)$ and content download performance of route 2 are limited by link 1.On the other side, route 3 is bottlenecked at the third link as $C_3(1-\rho_3) < C_1(1-\rho_1)$.

Fig. 3.1 and 3.2 reports analytical and simulated average content delivery time as a function of the popularity distribution and in a subset of configurations. We consider M = 20000 files, equally partitioned into K = 2000 classes, of average size σ = 100 packets, of size P = 10kBytes. Content popularity distribution is assumed to be Zipf(α) with α = 2. Network links have a propagation delay equal to 10 μs, while nodes are equipped with an LRU cache of $x(i)$ = 5000 packets for $i = 1,2$ (90% of the most popular content items in the catalog). Finally, content request rate is $\lambda = 1$ content item/s and $W = 2$.

Case I is illustrated in Fig. 3.1 where we set $(C_1, C_2, C_3) = (10, 20, 30)$Mbps. As expected, the content delivery time is the same for all popularity classes, being link 1 the bottleneck for all

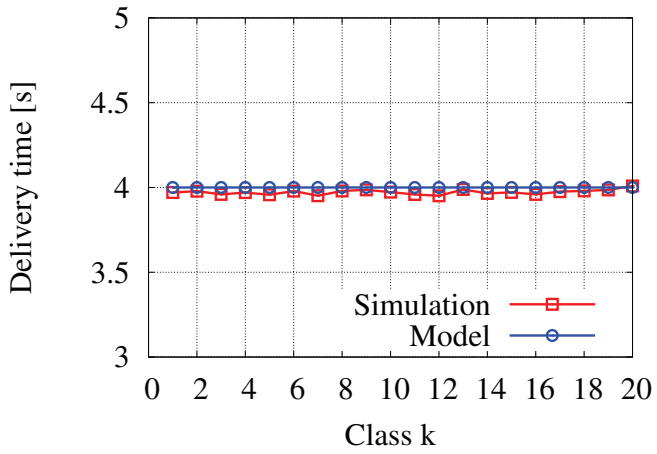


Figure 3.1: Linear topology scenario. Average content delivery time as a function of the popularity class k with network in case I.

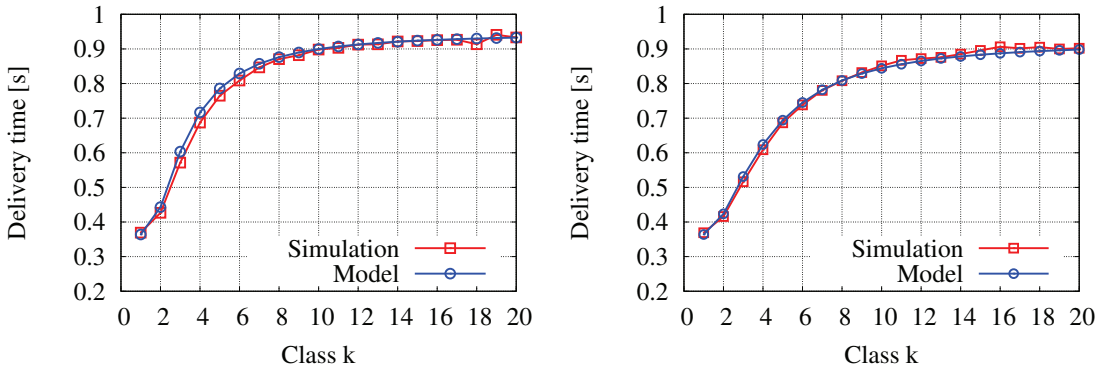


Figure 3.2: Linear topology scenario. Average content delivery time as a function of the popularity class k with network in cases II and III from left to right.

routes, and a good match between model and simulation results can be observed.

Fig. 3.2 shows $\mathbb{E}[T_k]$ in the linear topology in configuration (ii) with $(C_1, C_2, C_3) = (30, 10, 20)$ Mbps, and (iii) with $(C_1, C_2, C_3) = (30, 20, 10)$ Mbps. In case II and III we remark a significant reduction of the average content delivery time w.r.t. to case I, which benefits in a larger proportion to more popular classes, whose requests are more likely to be satisfied by the first cache.

Let us now analyze the binary tree topology described in Fig. 2.1(b), in a symmetric and asymmetric setting depending on link capacities across the network and on content request rates. In the

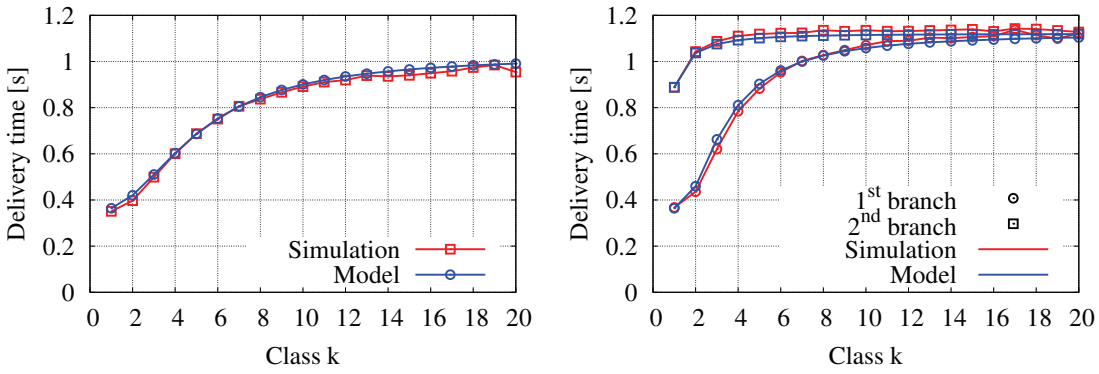


Figure 3.3: Symmetric (left) and asymmetric (right) binary tree topologies. Average content delivery time as a function of the popularity class.

symmetric case, we set $\lambda = 1$ content item/s, $(C_1, C_2, C_3) = (30, 20, 10)$ Mbps respectively for the first, second and third level links, while $x(1) = x(2) = 5000$ packets. In the asymmetric case, instead, we differentiate between first and second branch of the tree as denoted in Fig. 2.1(b). Notice that, we indicate with $C_{i,j}$ and $x(i,j)$ respectively the link capacity and the cache size at level i of the branch j and with λ_j the total request rate from a leaf node of the branch j. For the first branch, $\lambda_1 = 1$ content item/s, $(C_{1,1}, C_{2,1}) = (30, 20)$ Mbps, whereas for the second one, $\lambda_2 = 2$ content items/s, $(C_{1,2}, C_{2,2}) = (30, 30)$ Mbps. The third level links, shared by the first and second branch have capacities $C_3 = 20$ Mbps. Moreover, cache sizes are: $(x(1,1), x(1,2), x(2)) = (5000, 1000, 1000)$ packets. With such parameters, first and second branch are in configurations (iii) and (iv), respectively, being both characterized by a bottleneck in the same shared link 3. Compared to the symmetric tree, the second branch is under-provisioned in terms of storage capacity. This results in higher traffic load upstream and consequent larger delivery time for the second branch. The first branch performance is, however, almost unchanged. If we compare the two tree setups in Fig.3.3 we see that performance for user's belonging to the first branch are insensitive to upstream overload due to misconfiguration of the storage in the second branch.

3.4 Conclusion

In this chapter, we provided an analytical framework for the evaluation of average content delivery performance under statistical bandwidth and storage (from Chapter 2) sharing. With the proposed model we captured the bandwidth/storage tradeoff under fairly general assumption on total de-

mand, topology, content popularity and limited network resources. Moreover, as for the storage sharing model, in order to validate the analytical results, we also compared our analytical results against packet-level simulations.

Using the proposed analytical framework, we explicitly wrote bounds for the content delivery time as a function of the content popularity class for the analyzed networks (line and binary tree topology with 3 links). In this scenario, we identified four possible configurations based on the bottlenecks position. Except one case in which the bottleneck was placed close to the users, as expected, our results highlighted that the content delivery time highly depends on the popularity class (more popular content objects can be found close to the user) of the requested data.

Chapter 4

Conclusion of PART I

The CCN paradigm (more generally ICN) radically changes transport networks: we move from the original TCP/IP model of a flow of bytes from a source to a destination to receiver-driven retrieval of named data pieces triggered by end-user requests. Ephemeral data carried by agnostic IP packets are replaced with self-identified and self-certified data chunks stored by forwarding network nodes and accessible by users' requests. Distributed in-network caching is an attractive property of ICN architectures as it avoids inefficient retransmissions of the same data from the source multiple times and naturally favors mulch-point to mulch-point communication. In this context, the analysis of the user performance in terms of content delivery, constitutes a fundamental building block for the design of a receiver-driven transport control protocol aimed at avoiding network congestion while realizing specific fairness criteria. Independently on the specific ICN architecture considered, the definition and evaluation of transport and caching protocols is of the most importance to prove the viability of the entire concept.

In the first part of this dissertation, we provided an analytical framework for the evaluation of average content delivery performance under statistical bandwidth and storage sharing. In particular, in Chapter 2, assuming particular traffic characteristics (Zipf content popularity with $\alpha > 1$ and IRM requests), we analyzed the LRU and RND cache performance in terms of miss probability. Exploiting this results, we then introduced the bandwidth sharing model (Chapter 3) in the analytical framework, providing formula for the average content delivery performance evaluation. Our analysis, though not tailored to a specific transport protocol, made the assumption of optimal receiver-driven flow control yielding full and fair resource utilization.

In the context of the CCN proposal, preliminary results on transport protocol appeared in [43]. However, a comprehensive CCN transport protocol design still lacks and will be presented in part II. To this purpose, the model presented in this chapter constitutes a reference for the evaluation of the newly designed protocols in terms of efficiency and fairness.

Some points remain open in the CCN performance analysis. For the storage sharing performance analysis, different content popularity profiles can be considered and some hypotesis on the one used in this thesis relaxed (e.g. zipf with $\alpha > 1$). Jelenković et al. in [44] and Roberts et al. in [32] considered different popularity profiles (e.g. weibull, zipf with $\alpha \leq 1$, combination of multiple zipf with different shape parameters) for the LRU replacement policy supposing caching is performed at file-level rather than packet-level (as in CCN). In the same direction, the RND modeling framework we proposed need to be extended to the case of packet-level caches to better represent the CCN scenario. Finally, an interesting point is represented by the comparison of the obtained models aginst real traffic traces in more complex and realistic network topologies.

Part II

Transport protocol and Forwarding Mechanisms for CCN

Introduction

In the previous part of the dissertation, we analyzed the user performance (i.e. content delivery time) through storage and bandwidth sharing modeling in the CCN architecture. We introduced closed formulas to predict the optimal and efficient resource sharing equilibrium. Using the mathematical tools introduced in the first part as a benchmark, the second part of the dissertation contributes to the design and the analysis of transport and forwarding mechanisms suitable for CCN.

Due to the connectionless nature of the CCN communication model (one-to-many communication), a receiver- driven congestion control protocol is the most natural choice. In Chapter 5 we introduce a receiver-driven congestion control protocol, namely Interest Control Protocol (ICP) that adapt the maximum number of interest the receiver is allowed to send in parallel (interest window), following the Additive Increase Multiplicative Decrease (AIMD) rule commonly adopted in TCP. We then analyze the performance of ICP and prove that it reaches the same efficient and fair equilibrium described by the formula presented the first part of the dissertation.

Despite its one-to-many communication model, the CCN architecture has an interesting feature that can be exploited in a congestion control protocol. In CCN, requests (interests) are routed-by-name and data follow the reverse path from the repository (or a cache) to the receiver. The congestion control protocol can exploit this feature adding an Hop By Hop mechanism, that we introduce in Chapter 6, regulating the interest emission rate at a given interface in order to prevent congestion. Moreover, as for ICP, we prove that ICP coupled with the HBH mechanism converges to the efficient and fair equilibrium described in the first part of the dissertation.

The design of the receiver-driven congestion control protocol without the management of the different paths (joint or disjoint) to possible data locations (as ICP), results inefficient when paths are significantly different in terms of capacity and propagation delay. To cope with this limitation, we modify our first congestion control protocol design to use a Remote Adaptive Active Queue Management (RAAQM) mechanism in order to efficiently control multiple paths in parallel. We then study the new congestion control protocol and prove its stability. Furthermore, we introduce a per interface and per prefix interest forwarding mechanism comparing its performance to the optimal forwarding strategy.

Chapter 5

Interest Control Protocol - ICP

In this chapter, we present the design of a CCN transport protocol which aims at attaining the efficient and fair regime studied in Part I. Let us first of all review some related works on receiver driven transport protocols.

5.1 Related Works

Currently deployed transport protocol as TCP variants like Reno, Cubic or Compound use a sender-based controller to detect congestion and to regulate packet sending rate via a window-based mechanism. However in CCN, the control is completely delegated to the receiver that expresses per packet requests (interests). In the literature of transport and congestion control protocols, the idea of delegating some or all control functions to the receiver, as in CCN, is not new and various proposals can be found. A partial delegation of transmission rate control to the receiver is advocated in WTCP [45], TFRC [46] and TCP Real [47] proposals.

In particular, WTCP (Wireless TCP), introduced by Sinha et al., is a rate-based transport protocol monitoring the inter-packet delay at the receiver to adjust the congestion window at the sender, so partially delegating the transmission rate control to the receiver. WTCP objective, is to improve the poor TCP performance over wireless links by avoiding the mis-interpretation of transmission losses as congestion indications. The same functionality appears in TFRC, and TCP Real proposals. More precisely, TFRC (TCP Friendly Rate Control) by Floyd et al. , makes the receiver calculate the congestion control information, i.e. the loss event rate, by claiming that this modification would result in a simpler management of multicast communication. Similarly, TCPReal by Tsaoussidis et al., incorporates a receiver-driven control component by adding a new feature to the receiver, namely the possibility to adjust the window prior to congestion, through ACK piggybacking. The objective of TCPReal, is twofold: eliminate inaccuracy in congestion

information due to ACK losses and enhance responsiveness of congestion control.

A complete receiver-centric transport protocol is proposed in WebTP [48], RCP [49] and its extension R2CP [50]. Targeted to web content transport, WebTP by Gupta et al. , moves transport initiation, flow control and congestion control functionalities at the receiver. Moreover, it also specifies a naming scheme allowing receivers to request data in a semantic fashion similarly to CCN. RCP (Reception control protocol), proposed by Hsieh et al. , is a TCP clone delegating all the control functionalities to the receiver with the aim of exploiting channel condition awareness available at mobile hosts to take power-saving decisions in an efficient way.An extension of this protocol, R2CP by Chien et al. , allows the receiver to use k different active interfaces by controlling k independent connections.

In [51], authors acknowledge the various advantages of a receiver-driven solution and tackle the issue of a misbehaving receiver, focusing on the RCP transport protocol. They outline the vulnerabilities due to shift of key control functions at the receiver in terms of denial of service attacks, resource stealing and protocol parameters malicious tuning. Moreover, an end-point solution requiring a sender-side verification is proposed and analytically evaluated. In a different spirit, the transport protocol developed in [52], Dynamic Video Rate Control (DVRC) builds on top of UDP and realizes a hybrid AIAMD (Additive Increase Additive/Multiplicative Decrease) receiver-centric rate control with the additive decrease component that smooths the protocol response to non congestion losses.

The problem of the definition of a CCN transport protocol is faced in [43] by Ott et al. , in which a preliminary design of a pull-based and completely data-oriented mechanism, named ConTug, is provided. In ConTug, the receiver estimates the number of sources available in the network for a specific content and controls an equal number of 'conceptual congestion windows'(CCWND) kept at the receiver in a TCP-like fashion, with the difference that CCWNDs are kept at the receiver.

5.2 ICP Design

Protocol Objectives

A robust receiver-driven transport protocol is responsible for realizing efficient data delivery by controlling the receiver request rate and adapting it according to available network resources. In CCN, a transport session is realized by Interest queries progressively sent out by the receiver, routed by name towards a content store. In the ICP design process, protocol objectives are:

- **Reliability**: the first goal of ICP is to guarantee reliable data transfers by re-expressing Interests in case of Data packets losses. ICP schedules Interest retransmissions at the expi-

ration of a timer τ, maintained at the receiver. In this way, the transport protocol does not rely on losses as congestion notifications, but on delay measurements and timer expiration.

- **Efficiency**: the second objective of ICP is to minimize the completion time of data transfers. To this aim, ICP utilizes an AIMD (Additive Increase Multiplicative Decrease) mechanism for adapting the Interest window size as a mean to attain the maximum available rate allowed on the network path. The choice of such window-based controller is motivated by the large body of work in TCP literature, proving that even in the receiver-driven case the optimal controller is AIMD. Indeed, the window based approach has been adopted in many packet data networks (TCP/IP), and we believe it can be successfully used in CCN.

- **Fairness**: the third goal of ICP is to achieve fair bandwidth allocation among flows (namely, content retrievals), sharing the same route and, thus, the same limited down-link bandwidth. Let us specify the details of the ICP protocol design, driven by the objective to attain a globally efficient and fair equilibrium.

Protocol description

In CCN, data packets are identified by a unique name (the content name plus a segment identifier) and are requested via Interests in the order decided by the application. In our design, the receiver window, W, defines the maximum number of Interests a receiver is allowed to send in parallel. Furthermore, W, increase and decrease following an AIMD mechanism that is better detailed in the next sections.

$\mathcal{P}$	set of paths between user and repository(ies)
$\mathcal{R}^{\phi}$	set of routes of path ϕ (i.e. hops of a given path)
$R(r_i^{\phi})$	Round trip delay of route r_i^{ϕ} (user to i^{th} hop on path ϕ)
W	Interest window
η,β	Window increase and decrease factors
τ	Interest retransmission timer
$W_k(t)$	Interest window of class k at time t
$X_k(t)$	Interest rate of class k at time t
$Q(i,t)$	Output queue occupancy at hop i at time t (on a path)

Table 5.1: Notation

Interest Window Increase

W is increased by η/W at Data packet reception. This operation is implemented using integer arithmetic so that W is incremented by η (set to 1 as default value) when a complete window of interest is acknowledged by data reception. Fig.5.1(a) shows window's evolution at the beginning of a flow.

Interest Window Decrease

Each interest is associated with a timer τ and a congestion detection coincides with a timer expiration. In this case, the protocol reacts by multiplying W by a decrease factor $\beta < 1$. All the timer expiration that occurs after the first one (that signal congestion) and before τs are ignored in order to avoid multiple successive W decreases (i.e. Fig.5.1(b)). Note that re-expression of an interest, after a time-out, is necessary to recover losses, even though data may just be delayed at the bottleneck. At the same time, retransmitted Interests sent just after a time-out can be filtered, if the PIT timer at the first node is not yet expired (see, for example, Interests 7 and 8 in Fig.5.1(b)).

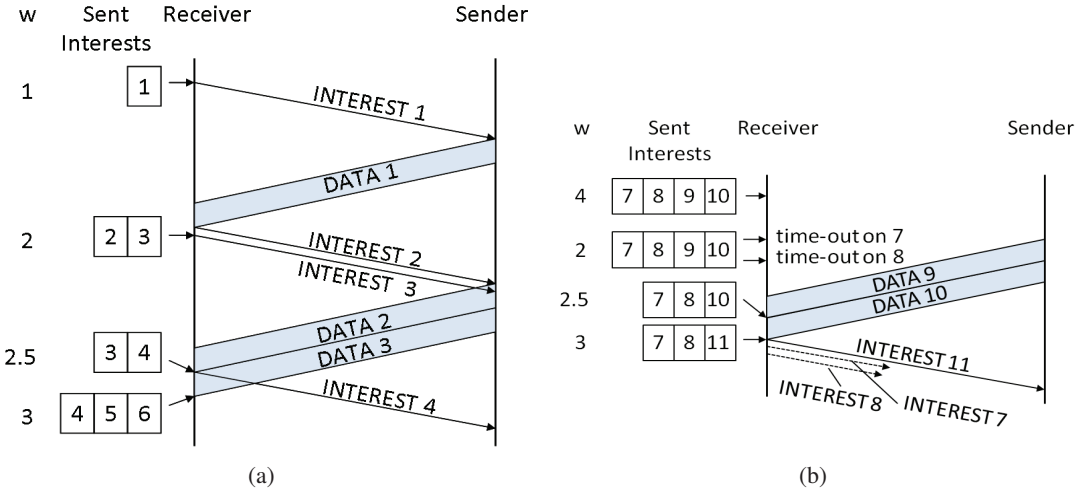


Figure 5.1: ICP window evolution: increase (left), decrease (right).

Timers

Properly setting the interest timer value τ and the PIT timer (which allows to remove a pending interest entry at a node) is crucial in CCN. About τ, note that in the rest of the chapter we consider both constant and variable timer values. In general, τ must be trivially set larger than the minimum network delay, otherwise every sent interest would trigger a timer expiration. In addition, a minimum τ value is necessary to guarantee high utilization of the available bandwidth, as precised by our single and multiple bottleneck analysis in the next sections. A trade-off between a large τ value for efficiency reasons and a small τ value for faster window adaptation, is given by a variable timer based on the estimation of the minimum round trip delay, as common in TCP literature (see e.g. [53]). Therefore, in the case of variable τ, ICP maintains round trip delay estimates at every data packet reception, by updating $\mathrm{RTT}_{\min}$ and $\mathrm{RTT}_{\max}$ averaged over a history of samples (30 Data packets in our implementation), excluding retransmitted packets. Each flow adapts its timer

τ according to

$$\tau = \text{RTT}_{min} + (\text{RTT}_{max} - \text{RTT}_{min})\delta \tag{5.1}$$

with $\delta < 1$ (i.e. $\delta = 0.8$ as default in our implementation). Notice that, in our design, a complete history of RTT samples is necessary before triggering a window decrease.

The PIT timer is fundamental in CCN to limit the number of pending Interests, hence the table size. The larger this value the higher the number of filtered Interests, not forwarded upstream towards content repository. On the other hand, a small PIT timer implies a large number of unnecessary retransmissions, since delayed packets arriving after PIT time-outs are discarded. The optimal PIT timer is not studied here and throughout this chapter, we simply set the PIT timer never smaller than τ.

5.3 Preliminary analysis of a single bottleneck

Consider the case of a single bottleneck link of finite down-link capacity C pkt/s relying a user to a content repository. We assume a constant round-trip propagation delay R and a constant interest retransmission timer $\tau > R$.

We focus on a single content retrieval before extending the analysis to the case of n parallel content transfers. Interests are sent using the protocol described in the previous section, with no maximum window limitations and fixed timer τ. Data packets are queued at repository output interface, as a result of down-link bandwidth limitations. The queuing delay is $Q(t)/C$, where $Q(t)$ is defined as the instantaneous queue occupancy in number of packets in the repository output queue at time t.

We analyze the system evolution via a deterministic fluid model of the instantaneous rate at the receiver $X(t)$ and the instantaneous queue occupancy $Q(t)$ at the repository, both modeled as continuous time variables (see [54]). System dynamics are described by the following ODEs:

$$\begin{aligned} \dot{X}(t) &= \frac{\eta}{(R + Q(t)/C)^2} - \beta X(t) \mathbb{1}_{\{R+Q(t)/C=\tau\}}, \\ \dot{Q}(t) &= X(t) - C\mathbb{1}_{\{Q(t)>0\}} \end{aligned} \tag{5.2}$$

The interest rate is assumed to be linearly proportional to the interest window $W(t)$, that is $X(t) = W(t)/(R+Q(t)/C)$ and hence it linearly grows proportionally to the inverse of the square of the round trip time. A timer expiration triggers a window (hence, a rate) drop proportional to

the decrease factor $\beta < 1$.

We remark that the model defines a time-out at the instant when the round trip time is equal to τ to avoid unrealistic repeated window decreases. This models the fact that the protocol triggers no more than one window decrease over a time window of τ, after a timer expiration. Note that this fluid representation via ODEs has been extensively adopted in TCP modeling literature.

Proposition 5.3.1. *The steady state solution of Eqs.(5.2) is periodical with limit cycle of length* $\widetilde{T}$, $0 < \widetilde{T} < \frac{2C\beta\tau^2}{(2-\beta)\eta}$. *Given* $\tau > \tau_{min}$, *with* $\tau_{min} = R + \frac{2\eta}{C}(\frac{2-\beta}{\beta})^2$, *the average stationary values of* $X(t)$ *and* $Q(t)$ *over* $\widetilde{T}$ *satisfy the following conditions*

$$\begin{aligned} \widetilde{X} &= \frac{1}{\widetilde{T}} \int_0^{\widetilde{T}} X(t)dt = C, \\ \widetilde{Q} &= \frac{1}{\widetilde{T}} \int_0^{\widetilde{T}} Q(t)dt \in [Q_L, C(\tau - R)] \end{aligned} \tag{5.3}$$

where $Q_L = C(\tau - R) - \frac{1}{3\eta}(\frac{C(\tau-R)\beta}{2-\beta})^2$

Proof. Consider the $X(t)$ evolution between two points t_n and t_{n+1}, where $Q(t_n)/C = \tau - R$, $n \geq 1$ and t_n, t_{n+1} represent two congestion events. Given the value of $X(t)$ at congestion points, that is $X(t_n^-)$ before the multiplicative decrease and $X(t_n^+) = (1-\beta)X(t_n^-)$ immediately after, we define an upper and lower bound for $X(t)$, with $t \in (t_n, t_{n+1})$,

$$X(t_n^+) + \frac{\eta}{\tau^2}(t - t_n) \leq X(t) \leq X(t_{n+1}^-) - \frac{\eta}{\tau^2}(t - t_{n+1})$$

The bounds define two parallel linear trajectories with same slope η/τ^2, whose values in t_n are $X(t_n^-)$ and $X(t_n^+) = (1-\beta)X(t_n^-)$ (respectively for the upper and lower bound) and with discontinuous decrements of amplitude $X(t_n^+) = (1-\beta)X(t_n^-)$.
Given $X(0)$ at $t = 0$, one can write

$$\begin{aligned} X(t_n^-) &= C[2\sum_{i=0}^{n}(\beta-1)^i + \zeta(1-\beta)^n] \underset{n\to\infty}{\longrightarrow} \frac{2C}{2-\beta}, \\ X(t_n^+) &\underset{n\to\infty}{\longrightarrow} \frac{2C(1-\beta)}{2-\beta}, \end{aligned}$$

with $\zeta = \sqrt{1 - 2\frac{X(0)-\tau\eta}{C} + \frac{X(0)^2 - 2\eta Q(0)}{C^2}} - 1$.

The time average of the periodic limit cycle is then computed as

$$\widetilde{X} = \frac{1}{\widetilde{T}} \int_0^{\widetilde{T}} X(t)dt = \lim_n \frac{X(t_n^-) + X(t_n^+)}{2} = \frac{1}{2}[\frac{2C}{2-\beta} + \frac{2C(1-\beta)}{2-\beta}] = C.$$

An upper bound for the queue occupancy is given by $C(\tau - R)$ reached in the instants of timer expiration before window decreases. A lower bound for the average queue occupancy in steady state can be obtained by using the same bounds on $X(t)$. Hence,

$$\widetilde{Q} = \frac{1}{T} \int_0^T Q(t)du > C(\tau - R) \left(1 - \frac{C\beta^2\tau}{3\eta(2-\beta)^2}\right).$$

To compute the lower bound for $Q(t)$, we impose $Q(t) > 0$ in steady state, guaranteed by $\tau > R + \frac{2\eta}{C}(\frac{2-\beta}{\beta})^2$. Otherwise the full resource utilization is not attained. Observe that this means that the queue must be able to accommodate at least the bandwidth delay product CR plus $2(2/\beta - 1)^2$ packets. □

From this analysis we can draw the following conclusions:

- The value of τ must be chosen larger than the network delay R, otherwise every interest request expires and the window decreases to the minimum value.

- At the same time, τ should be chosen large enough to accommodate more than the bandwidth delay product in order to fully utilize the link capacity as reported in the proof of Prop.5.3.1. Here we focus on a large buffer regime with no packet losses. In case of small buffers, the queue would empty with consequent loss of utilization.

Prop.5.3.1 can be generalized to the case of multiple flows sharing the same link.

Proposition 5.3.2. *Given a variable number of flows sharing the same link, generated according to a Poisson process $\mathcal{N}$ of rate λ, each one composed of σ packets on average, the expected flow rate in steady state is $\mathbb{E}[\widetilde{X}] = C(1-\rho)$, where $\rho = \lambda\sigma/C$ represents the charge on the link.*

Proof. First observe that in case of a constant number of flows, n, sharing the same link, Prop.5.3.1 can be easily extended to show that $\widetilde{X} = C/n$ for each of them. Now, under the assumption of Poisson generated flows of size σ packets on average, the number of ongoing flows on the link varies over time as the number of customers in a M/G/1 processor sharing queue, that is geometrically distributed with parameter $\rho = \lambda\sigma/C$. Hence,

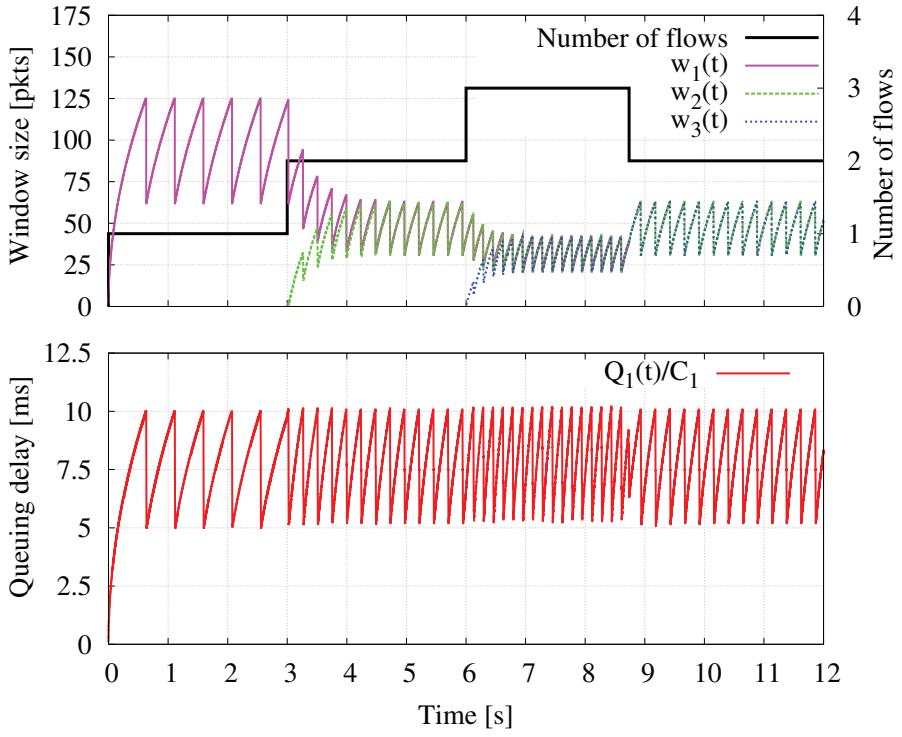


Figure 5.2: Interest window and queuing delay evolution for three data transfers, with R=20 μs, C=100 Mbps, $\tau = 10$ ms.

$$\mathbb{E}[(C/n)\mathbb{1}_{\{n>0\}}|\mathcal{N}=n] = \frac{C(1-\rho)}{\rho}\rho = C(1-\rho).$$

Note that the equivalence with the M/G/1 processor sharing queue holds in practice for long lived data transfers. □

Using the CCN packet level simulator (see Chapter 9), in which we also implemented the ICP protocol, we run simulations in order to support analytical findings and assess model accuracy. In Fig.5.2 we show the simulated interest window size evolution of three content retrievals starting at different instants, with the first transfer ending at $t = 9$ s. The queuing delay is also reported. Let us first observe the periodical behavior of window size and queue occupancy, whose limit cycle satisfies analytical prediction. Moreover, as expected, the frequency of queue occupancy oscillations is proportional to the number of parallel transfers n, since the queue input rate linearly

increases with slope $n\eta$.

5.4 Multi-bottleneck analysis

Let us extend this analysis to the network case where multiple bottlenecks can exist and the distance between the user and the closest content replica is impacted by the caches along the route. For the multiple bottleneck analysis, we use the performance measure $\mathbb{E}[T_k]$ (obtained in Chapter 3) as an optimal reference curve for the designed protocol. As observed in Part I, each path is associated a metric that we denote as Virtual Round Trip Time that accounts for the average distance (in time) at which a Data packet is retrieved. More precisely, we report here the VRTT definition adding the dependence on path ϕ :

Definition 5.4.1. *$VRTT_k$ represents the average time interval between the dispatch of an Interest and the Data packet reception for packets of content items in class k.*

$VRTT_k^\phi$ is defined for a given path ϕ from the user to a repository as a weighted sum of the round trip delays $R(r_i^\phi)$ associated to the sub-paths r_i^ϕ, i.e. to the route between the user and the i^{th} hop, with $i = 1...N$, where the weights correspond to the stationary hit probabilities $(1 - p_k(r_i^\phi))$ to find a packet of class k at hop i on the considered path, given that a miss was originated by all previous hops. For $k = 1, \ldots, K$,

$$VRTT_k^\phi = \sum_{i:r_i^\phi \in \mathcal{R}^\phi} R(r_i^\phi)(1 - p_k(r_i^\phi)) \prod_{j=1}^{i-1} p_k(r_j^\phi) \tag{5.4}$$

To ease the notation in the case of a single path, we will replace r_i^ϕ by i, with $i = 1, ..., N$ whenever a single path is considered.

The virtual round trip time weights the delivery time associated to each route by the probability to retrieve a data packet from the given route. As a result of the interaction among flows, and of the transport protocol reaction triggered by time-outs, route delivery time varies over time on a short timescale. On the contrary, the probability to retrieve a Data packet from a given route, changes on a longer timescale. In fact, it depends on cache dynamics at different network levels, less sensitive to traffic variations. Such timescale separation is confirmed by our simulations and motivated by the stationariness of the global request process which impacts the hit/miss probabilities at different caches. Therefore, we can refer to the storage sharing analysis as a *macroscopic* model of storage sharing of CCN network dynamics, which gives us the hit/miss probabilities based on a stationary request process. On top of it, we provide in this section a *microscopic* study of the packet-level dynamics of content retrieval (flows), subject to the AIMD interest rate control

described in Sec.5.2. The analysis proves that ICP realizes the fair bandwidth sharing equilibrium described in Chapter 3.

Let us state the main results about rate and queue occupancies evolution and exemplify it in a three links scenario as in Fig.5.4 (right branch).

Main results

Let us focus on the case of a single path between the user and the content repository. Therefore, we omit the path index ϕ. The number of ongoing flows of class k over route r_i (route between the receiver and the node i) is denoted by $n_{i,k}$, where the total number of flows sharing that route is $n_i \equiv \sum_{k=1}^{K} n_{i,k}$. We generalize the single bottleneck analysis to the case of a path composed of N links (with link capacities C_1,.., C_N) as in Fig.5.4, fed by content requests at the leaf node. We assume a propagation delay negligible with respect to the queuing delay at each link. The instantaneous rate $X_k(t)$ is modeled by a weighted sum of the rates achieved on different routes, $X(i,t)$, corresponding to the retrieval of packets from different hops. We write the following set of equations:

$$
\begin{aligned}
X_k(t) &= \sum_{i=1}^{N} X(i,t)(1-p_k(i)) \prod_{j=1}^{i-1} p_k(j) \\
\dot{X}(i,t) &= \frac{\eta}{\text{VRTT}(i,t)^2} - \beta X(i,t) \mathbb{1}_{\{\text{VRTT}(i,t)=\tau\}}, \\
\dot{Q}_i(t) &= n_i X(i,t) - C_i \mathbb{1}_{\{Q_i(t)>0\}} + \sum_{l=i+1}^{N} \Big(\prod_{j=i+1}^{l} \mathbb{1}_{\{Q_j(t)=0\}} n_j X(l,t) + \mathbb{1}_{\{Q_l(t)>0\}} C_l \Big)
\end{aligned}
\tag{5.5}
$$

where $\text{VRTT}(i,t) = \sum_{j=1}^{i} \frac{Q_j(t)}{C_j}, \quad i = 1, ..., N$ denotes the round trip time given by the sum of queuing delays associated to route r_i. Note that the ODEs for $X(i,t)$ are the same as in the single link case, where $X_k(t) = X(1,t)$.

Observation 5.4.2. *It is worth noticing that content popularity affects the offered traffic on each link, as a consequence of request rate and hit/miss probabilities at different hops. This leads to a different rate $X_k(t)$. However, the rate each flow gets on a given route, $X(i)$, is the same for all popularity classes given that the delay associated to a route r_i, VRTT(i,t), is the same for all $k = 1, ...K$. Therefore, flows fairly compete on a given route for sharing bandwidth.*

Proposition 5.4.3. *The steady state solution of Eqs.(5.5) is periodical with limit cycle of length $\widetilde{T}$*

and it satisfies

$$\widetilde{X}_k = \sum_{i=1}^{N}(1-p_k(i))\prod_{j=1}^{i-1}p_k(j)\widetilde{X}(i),$$
$$\widetilde{X}(i) = \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} X(i,t)dt = \gamma(i),$$
$$\widetilde{VRTT(i)} \equiv \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} VRTT(i,t)dt = \frac{1}{\gamma(i)}, \tag{5.6}$$

where $\gamma(i)$ is the max-min fair rate of route i defined in Chapter 3.

The proof is a straightforward generalization of the following proof for the three links scenario.

Example: a three links scenario

Consider the case of a single path composed by three hops relying the user to the repository.

Corollary 5.4.4. *Given that the number of ongoing flows on route i is n_i, $i = 1, 2, 3$, it holds (5.6) where $\gamma(i)$ vary according to C_i/n_i values. Four cases can be distinguished (as in Chapter 3):*

- ***Case I):*** $C_1/n_1 < C_2/n_2, C_3/n_3$
  $\gamma(1) = \gamma(2) = \gamma(3) = C_1/(n_1 + n_2 + n_3)$.

- ***Case II):*** $C_2/n_2 < C_1/n_1, C_3/n_3$
  $\gamma(1) = (C_1 - C_2)/n_1$,
  $\gamma(2) = \gamma(3) = C_2/(n_2 + n_3)$.

- ***Case III):*** $C_3/n_3 < C_2/n_2 < C_1/n_1$
  $\gamma(i) = (C_i - C_{i+1})/n_i$, $i = 1, 2$,
  $\gamma(3) = C_3/n_3$.

- ***Case IV):*** $C_3/n_3 < C_1/n_1 < C_2/n_2$
  $\gamma(1) = \gamma(2) = (C_1 - C_3)/(n_1 + n_2)$,
  $\gamma(3) = C_3/n_3$.

Proof. Above equations (5.5) specialize into:

$$\begin{aligned}
\widetilde{X}_k =& \sum_{i=1}^{N}(1-p_k(i))\prod_{j=1}^{i-1}p_k(j)\widetilde{X}(i), \qquad i=1,2,3,\\
\dot{X}(i,t) =& \frac{\eta}{\text{VRTT}(i,t)^2} - \beta X(i,t)\mathbb{1}_{\{\text{VRTT}(i,t)=\tau\}}\\
\dot{Q}_1(t) =& (n_2X(2,t) + \mathbb{1}_{\{Q_3(t)=0\}}n_3X(3,t) +\\
&+ \mathbb{1}_{\{Q_3(t)>0\}}C_3)\mathbb{1}_{\{Q_2(t)=0\}} + C_2\mathbb{1}_{\{Q_2(t)>0\}} - C_1\mathbb{1}_{\{Q_1(t)>0\}}\\
\dot{Q}_2(t) =& n_2X(2,t) + \mathbb{1}_{\{Q_3=0\}}n_3X(3,t) + \mathbb{1}_{\{Q_3>0\}}C_3 - C_2\mathbb{1}_{\{Q_2>0\}}\\
\dot{Q}_3(t) =& n_3X(3,t) - C_3\mathbb{1}_{\{Q_3>0\}}
\end{aligned} \tag{5.7}$$

Note that we considered the homogeneous case of all flows starting from the same initial condition. However, the existence and uniqueness of a stable equilibrium point can be easily proved and it guarantees that this simplifying assumption has no impact on the attained stationary regime. As previously observed, we can distinguish four cases according to C_i/n_i values.

Case I): $C_1/n_1 < C_2/n_2, C_3/n_3$

By construction, in this configuration the first queue to build up is $Q_1(t)$ when the global input rate $n_1X(1) + n_2X(2) + n_3X(3)$ reaches C_1. This can be seen by simply rewriting the ODEs in the case of empty queues. After this point on, the rate of all flows keeps linearly growing on each route until the queuing delay equals the timer, i.e. $Q_1(t) = C_1\tau$, whereas $Q_2(t)$ and $Q_3(t)$ remain empty. Q_1 attains the limit cycle of Prop.5.3.1 never emptying out as $\tau > \tau_{min}$ and $0 < Q_1 \leq C_1\tau$. The system of ODEs reduces to

$$\begin{aligned}
\dot{X}(t) =& \frac{\eta}{(Q_1(t)/C_1)^2} - \beta X(t)\mathbb{1}_{\{Q_1(t)/C_1=\tau\}},\\
\dot{Q}_1(t)dt =& \sum_{i=1}^{3} n_iX(t) - C_1\mathbb{1}_{\{Q_1>0\}}
\end{aligned}$$

This system is solved as in Prop.5.3.1 and leads to $\widetilde{X}(i) = C_1/n$ with $n = \sum_{i\leq 3} n_i$.

Case II): $C_2/n_2 < C_1/n_1, C_3/n_3$

By construction, in this configuration the first queue to build up is $Q_2(t)$, when its input rate $n_2X(2) + n_3X(3)$ reaches C_2. From this point on, $Q_3(t)$ remains empty as the route r_3 is bottlenecked at link 2, whereas $Q_1(t)$ starts filling in as long as $n_1X(1,t) + C_2 = C_1$. The system of

ODEs for $i = 1, 2, 3$ reduces to:

$$\dot{X}(i,t) = \frac{\eta}{\sum_{j=1}^{i}\left(\frac{Q_j(t)}{C_j}\right)^2} - \beta X(i,t)\mathbb{1}_{\{\sum_{j=1}^{i}\frac{\dot{Q}_j(t)}{C_j}=\tau\}},$$
$$\dot{Q}_1(t) = n_1 X(1,t) + C_2 - C_1\mathbb{1}_{\{Q_1(t)>0\}},$$
$$\dot{Q}_2(t) = (n_2+n_3)X(2,t) - C_2.$$

Remark that the two ODEs in $X(1,t)$ and $Q_1(t)$ can be separately solved as in in Prop. 5.3.1, as long as $Q_2(t) > 0$. Such condition, coupled with $Q_1(t)/C_1 + Q_2(t)/C_2 \leq \tau$, $Q_1(t)/C_1 \leq \tau$ implies that the system is always fully utilized on route r_1 and r_2. In this regime the steady state solution is

$$\widetilde{X}(3) = \widetilde{X}(2) = C_2/\sum_{i=1}^{2} n_i,$$
$$\widetilde{X}(1) = (C_1 - C_2)/n_1 \geq C_1/\sum_{i=1}^{3} n_i.$$

Case III): $C_3/n_3 < C_2/n_2 < C_1/n_1$
In this configuration r_1 has a bottlenecked on link 1, r_2 on link 2 and r_3 on link 3. Following the same reasoning use for the previous cases, Q_1 always remains non empty. So results of Prop.5.3.1 applies to this case $0 < Q_{\min} \leq Q_1 \leq C_1\tau$. As in the previous case Q_2, Q_3 can be empty for a time instant without changing the bottleneck position and keeping the system at full utilization as the buffering along the routes r_1 is always positive. The system of ODEs reduces to:

$$\dot{Q}_1(t) = n_1 X(1,t) + C_2\mathbb{1}_{\{Q_2(t)>0\}} - C_1,$$
$$\dot{Q}_2(t) = n_2 X(2,t) + C_3 - C_2, \frac{dQ_3(t)}{dt} = n_3 X(3,t) - C_3.$$

By solving the ODEs by couple starting from $(X(1), Q_1)$ to $(X(3), Q_3)$ as in previous cases, it follows that in this case:

$$\widetilde{X}(1) = (C_1 - C_2)/n_1 \geq C_1/\sum_{i=1}^{3} n_i,$$
$$\widetilde{X}(2) = (C_2 - C_3)/n_2 \geq C_2/\sum_{i=1}^{2} n_i,$$
$$\widetilde{X}(3) = C_3/n_3.$$

Case IV): $C_3/n_3 < C_1/n_1 < C_2/n_2$

$$\dot{Q}_1(t) = (n_1 + n_2)X(1,t) + C_3 - C_1\mathbb{1}_{\{Q_1(t)>0\}},$$
$$\dot{Q}_3(t) = n_3X(3,t) - C_3\mathbb{1}_{\{Q_3(t)>0\}}, X(1,t) = X(2,t)$$

It results:

$$\widetilde{X}(1) = \widetilde{X}(2) = (C_1 - C_3)/(n_1 + n_2) \geq C_1/\sum_{i=1}^{3} n_i,$$
$$\widetilde{X}(3) = C_3/n_3.$$

□

In Fig.5.3 we report queue and window size time evolutions drawn from a simulation run with two data transfers in case II. The number of flows in progress vary from 4 to 6 as indicated by the right y axis. We plot $W_1(t)$, the window size of a flow retrieving a popular content (downloading chunks more likely from the first hop cache) and of $W_2(t)$ associated to a non-popular content downloading (retreiveng chunks more likely from the second hop cache).

The queuing delays, $Q_1(t)/C_1$ and $Q_2(t)/C_2$, respectively for link 1 and 2, show that queuing oscillates among the two bottlenecks a consequence of the start of flow 1, while the sum is bounded by τ, set to 4 ms, reflecting previous analysis.

In all cases, as long as the bottleneck link queue does not empty (which is guaranteed by $\tau > \tau_{min}$), the time average of the allocated rate corresponds to max-min fair share. If we de-condition on the number of concurrent flows (n_1, n_2, n_3), we find a lower bound for the throughput, or conversely an upper bound for the transfer time as computed in Chapter 3. The accuracy of max-min fair allocation is confirmed by the simulation results gathered in the next section.

5.5 Design guidelines and protocol assessment

In previous section we have shown that, under proper parameter setting, the equilibrium achieved by the system, in terms of average rates and queue occupancy, is max-min fair and efficient. In practice, the choice of the parameters can be critical and can potentially lead the system to an inefficient regime where bandwidth is under-utilized, as already observed in Sec.5.3. The present section provides guidelines for tuning protocol parameters, i.e. η, β and τ, by illustrating their

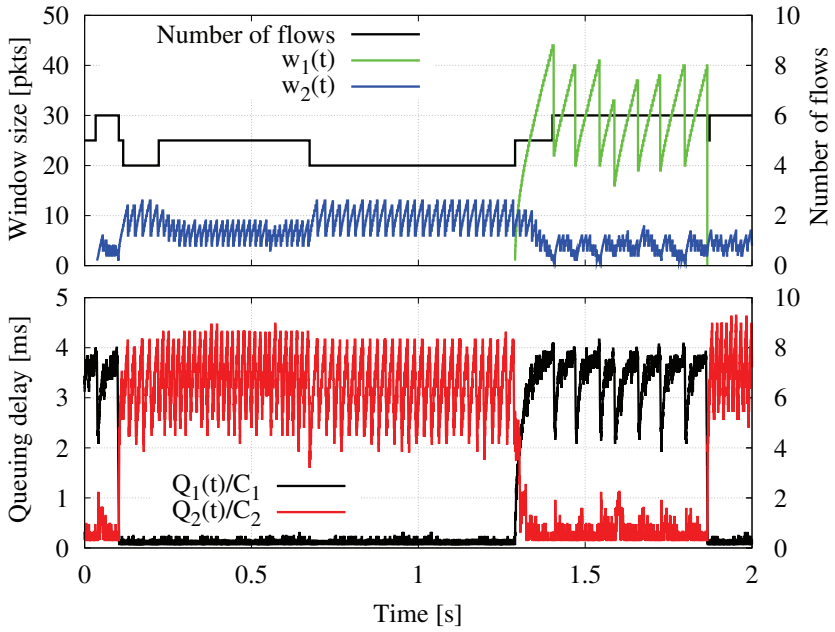


Figure 5.3: Window and queues evolutions in a two bottleneck scenario with C1=100Mbps C2=50Mbps and $\lambda = 1.5$

impact on system performance. The value of τ, also allows to estimate the minimum amount of buffering needed to attain the fully efficient regime.

Parameters setting

The *additive increase parameter* η, has a default value of 1 which corresponds to an interest window increase of one packet per round-trip time (as in TCP Reno). By increasing the window size more aggressively ($\eta > 1$), a flow can achieve higher throughput, if bandwidth resources are not fully utilized, as observed in [51]. In case of fair and efficient bandwidth sharing as the one described in Chapter 3, we observe that the average rate is not impacted by η, and its value only determines the length of the limit cycle.

Similar considerations hold for the *multiplicative decrease parameter* β, which has a default value of 0.5 (as in TCP Reno) such that the Interest window is halved upon the expiration of the timer. As for the η parameter, a smooth decrease, guaranteed by $\beta \geq 0.5$, results in higher flow throughput only if bandwidth is not fully utilized. Indeed, in case of efficient bandwidth sharing, the β value does not affect the average rate at the receiver, but rather the length of the limit cycle and the amplitude of the queue/rate oscillations.

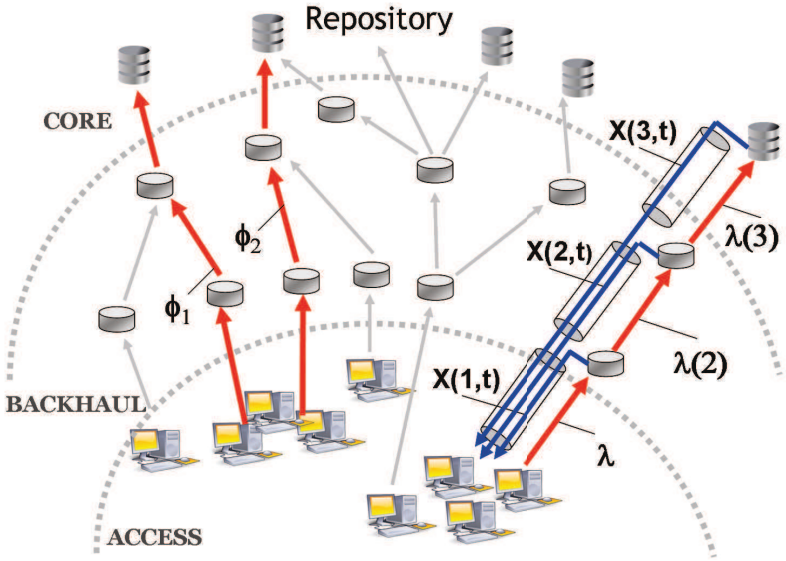


Figure 5.4: Topology

The most critical protocol parameter is the *interest retransmission timer* τ, that must be chosen not smaller than the minimum network delay to have non zero throughput, and, at the same time, large enough to queue at least the bandwidth delay product along the path. In this way, bottleneck bandwidth can be fully exploited. Moreover, an arbitrarily large τ would let ICP converge very slowly, which motivates our choice to set τ equal to the estimated minimum network delay (Eq.5.1).

Simulation results

To assess model accuracy and test protocol performance, in this section, we compare analytical and simulation results (obtained using CCNPL-Sim, see Chapter 9).

Constant τ value

We consider a multi-bottleneck case, namely a three links linear network (as the right branch in Fig.5.4), with negligible propagation delays, $\sigma = 5000$ packets (5MB, packet size is 1kB) and equal cache sizes of 50000 packets (50MB), implementing the previously studied LRU replacement policy (see Sec. 2.3). Content items are requested according to a Poisson process of intensity $\lambda = 1$ content/s and a given item in class k is requested with probability $\frac{1}{m}\frac{c}{k^{\alpha}}$, where $m = M/K = 5$ is the number of content items in the same popularity class, and $\alpha = 1.7$ is the Zipf's shape parameter. Fig.5.5 shows the average delivery time (top) and virtual round trip time,

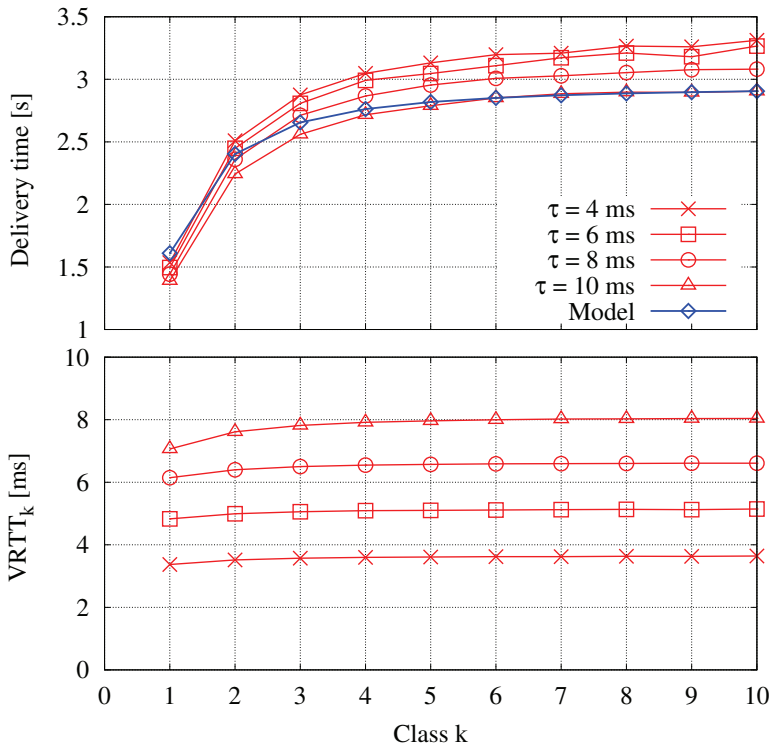


Figure 5.5: Delivery time and VRTT$_k$ in case II with C_1=C_3=100Mbps, C_2=40Mbps with fixed τ

VRTT$_k$ (bottom), experienced for content retrievals in the ten most popular classes, $k = 1, \ldots, 10$, for different values of $\tau \in [4 \text{ ms}, 10 \text{ ms}]$. Link capacity values are respectively C_1=C_3=100Mbps, C_2=40Mbps. As predicted by the model, VRTT$_k$ is shown to be proportional to τ in Fig.5.5 (bottom). About the delivery time, simulation results are compared to the optimal reference curve that indicates efficient and fair allocation of resources.

As a remark, the delivery time appears to be negatively affected by a too small τ value, especially for classes $k > 1$, since longer routes r_2, r_3 (of two/three hops) are not able to fully utilize bottleneck capacity C_2. Conversely, most popular data transfers ($k = 1$) get slightly better performance than what predicted by global optimum, as the τ value is well set for the shorter route r_1 they mainly utilize. We argue that each data transfer would benefit from adapting τ to the measured delay of the used routes and move to the case of variable τ.

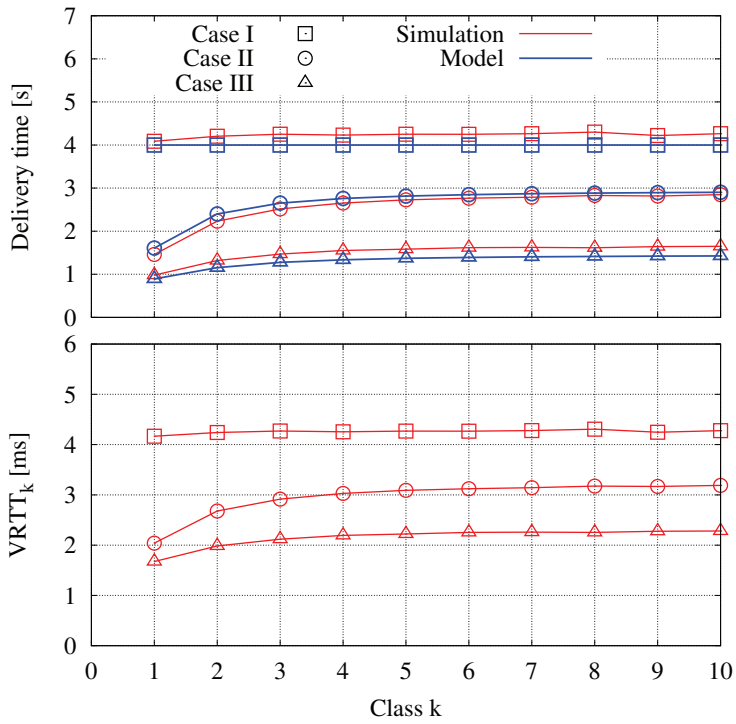


Figure 5.6: case I,II,III with adaptive τ case I: $C_1 = 50$Mbps, $C_2 = C_3 = 100$Mbps; case II: $C_1 = C_3 = 100$Mbps, $C_2 = 40$Mbps; case III: $C_1 = C_2 = 100$ Mbps, $C_3 = 50$Mbps

Variable τ value

In Fig.5.6 we present a set of simulations with the same parameters setting used in the previous section (varying the links Capacity) where every flow estimates the value of τ implementing the adaptive algorithm described in Sec.5.2. In particular, we analyze three of the cases defined in Sec.5.4: case I: $C_1 = 50$Mbps, $C_2 = C_3 = 100$Mbps; case II: $C_1 = C_3 = 100$Mbps, $C_2 = 40$Mbps; case III: $C_1 = C_2 = 100$ Mbps, $C_3 = 50$Mbps. Globally, the queuing delay converges to a much smaller value than that obtained using a constant τ, as a consequence of the fact that τ_k, for flows of class k, converges to VRTT$_k$ (that is function of the popularity class and, hence, of the data distribution along the path).

Multi-homed scenario

A CCN receiver does not care about the location of the requested Data packet. In CCN, routes are built following the reverse path of the Interests, routed by the name to the closest hitting

content store (or cache). Every receiver can easily exploit multiple interfaces by load balancing Interests on the different interfaces. Let us consider the network topology in Fig.5.4 with multiple repositories, where one group of users is multi-homed, making use of two disjoint paths, ϕ_1 and ϕ_2, in configuration II and III respectively (according to the classification in Sec.5.4) with $C_1^{\phi_1} = C_2^{\phi_1} = 100Mbps$, $C_3^{\phi_1} = 60Mbps$ and $C_1^{\phi_2} = C_3^{\phi_2} = 100Mbps$, $C_2^{\phi_2} = 60Mbps$.

Multi-homed users share bandwidth resources with other *single path* users exploiting ϕ_1 or ϕ_2 only. Performance results are reported in Fig.5.7 (top). A good agreement can be observed between simulation results and analytical predictions both in the single path and in the multi-path case. Moreover, as expected, VRTT_k and delivery time experienced by *multi-homed* users, for all k, are significantly reduced w.r.t. those perceived by ϕ_1/ϕ_2 *single path* users as a consequence of the higher throughput achieved by using both paths in parallel.

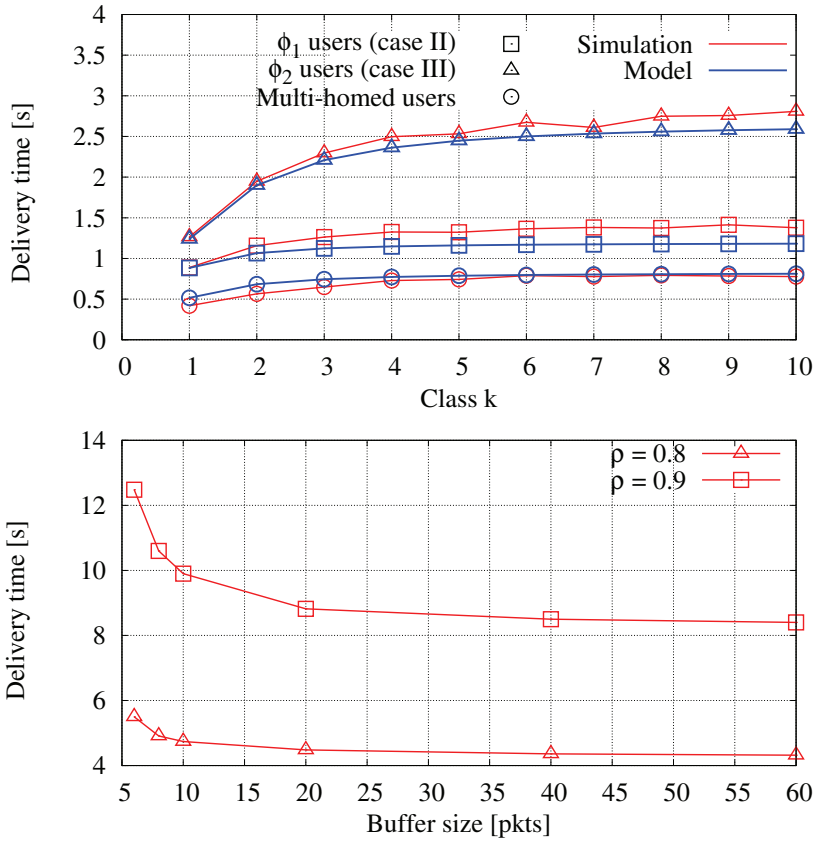


Figure 5.7: Multi-Homed network

Buffer Sizing

Throughout this chapter we have considered a network with enough buffer capacity to queue what requested by the τ value, in order to prevent queue from emptying. System dynamics in a small buffer regime have not been analyzed due to lack of space. For completeness, we report in Fig.5.7 (bottom) the average delivery time measured in a single bottleneck scenario at two loads, 0.8, 0.9, under different buffer sizes. The loss of performance due to little buffering can be significant. However, we believe that there is no clear advantage in saving output buffer memory in CCN, where technological limitations are mainly faced in the content store [55].

5.6 Conclusion

In this chapter we presented ICP, a transport protocol for content-centric networks driven by a window-based AIMD controller of the interest rate expressed at the receiver. ICP is conceived to control Data packet retrieval from the different routes (subpaths). In fact in CCN, even in presence of a single path between a user and a repository, Interest packets are forwarded up to the first hitting cache and Data packets may therefore be downloaded from there. Communication is, hence, intrinsically multi-point.

An analytical characterization of stationary rate and queue dynamics is provided, allowing to understand the impact of CCN transport and caching dynamics on user performance. The key takeaway of our analysis is that the present ICP design realizes the optimal statistical bandwidth sharing we previously studied in Part I guaranteeing full resource utilization and fair bandwidth allocation. Finally, protocol performance is evaluated by packet-level simulations corroborating analytical results on a hierarchical network topology.

Chapter 6

Hop-By-Hop congestion control

In CCN interest and data follow the same path in opposite directions. Exploiting this path simmetry, it is possible to implement an Hop-By-Hop interest shaper regulating interest emission at each network node with the objective of achieving faster congestion detection/reaction. The idea of an hop-by-hop flow controller, its not new and we start the chapter with a review of the related works.

6.1 Related Works

Hop-by-hop flow control mechanisms have been extensively studied in the past, as an alternative or in conjunction with end-to-end congestion control, to overcome the difficulties of the sender-driven rate- based flow control in handling bursty traffic. More precisely, Kung et al. in [56] and [57] observe that, in presence of bursty traffic, rate adaptation performed at the sender can be imprecise and slow to converge to the optimal value. Authors propose a credit-based flow control at each node, regulating data packet forwarding according to a credit balance. Each time a data packet is received, the credit balance is incremented by one unit, while a data packet forwarding corresponds to a decrement of one unit. The results present the performance improvement w.r.t end-to-end flow control only. In [58], Mishra et al. develop a hop-by-hop rate-base congestion control mechanism targeted to bursty traffic, as an alternative to its end-to-end version. However, the objective is different: intermediate nodes forward back to the sender information about congestion status and, ultimately, the sender adapts its rate, based on the received feedbacks.

In the context of CCN, or more generally of a pull-based network, such kind of mechanisms are particularly appropriate when applied to the requests' flows, as argued in [59] by Fdida et al. and in [60] by Roberts et al.

6.2 HBH Interest Shaping design

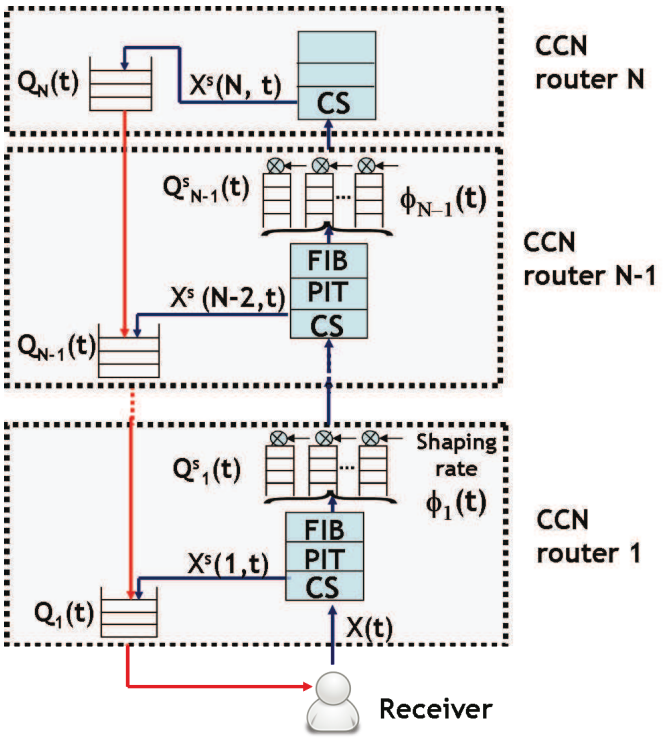


Figure 6.1: N hops network topology.

This section introduces an Hop-By-Hop Interest shaping mechanism, which realizes per-flow Interest rate control at the output interfaces of every CCN router (as illustrated in Fig.6.1). The rationale behind an additional Interest control realized at every network node is twofold:

- anticipate congestion detection and,
- trigger rate reduction via Interest shaping, before the receiver can detect the congestion via a timer expiration.

In every output interface, we maintain one virtual queue per flow, identified by the name of the content (i.e. the flow identifier is a hash of the name, excluding the packet identifier). Each virtual queue is associated a credit counter initialized to a maximum value B (in bytes), indicating the number of Data bytes the flow is granted to transmit, with no additional delay. The counter is incremented at the estimated fair rate of the corresponding down-link and decremented by forwarded Interests. The evolution of the virtual queue and the counter is driven by the algorithm reported in Algo.1. As a note, a flow refers to a single content retrieval, and is defined as bottlenecked

when at least one Interest packet is queued at the output buffer or the credit counter is null. Notice that keeping one virtual queue per flow implies to maintain a flow counter as long as there are outstanding Interests in the PIT.

Algorithm 1 Virtual queue f management

At interest enqueue (interest, f)
$bottlenecked[f] = ((length(f) > 0)OR(counters[f] <= 0))$
if $(bottlenecked[f])$ **then**
    $shaping_queue_tail(interest, f)$
else
    $link.enqueue(interest, t = 0)$
    $update_rate(rate, interest.data_size)$
end if
$counter[f] = max(0, counter[f] - interest.data_size)$

At interest dequeue()
    $interest = shaping_queue_head(current_q)$
    $reduced_rate = downlink_rate - rate$
    $link.enqueue(interest, t = interest.data_size/reduced_rate)$
    $current_q = next_q$

Different implementations of the flow table are possible, e.g. a hash table per content name storing a pointer to an Interest shaper for each output interface. This implementation avoids to have additional name lookups and can be directly included in the PIT. Indeed, the HBH mechanism exploits the PIT information about ongoing flows, by introducing a credit counter per active flow. In this way, the scalability with the number of ongoing flows is preserved. When an Interest arrives at the output interface, the algorithm checks if it belongs to a bottlenecked flow or not. If the flow is non- bottlenecked, the Interest is directly sent upstream and the counter decremented by the number of bytes of the corresponding Data packet. For a bottlenecked flow, the Interest is queued in a drop tail FIFO of size Q_{max} served at the shaping rate. The shaping rate, $\phi_i(t)$, of node i at time t, is set to the estimated fair rate, as we detail in the next section, in Eq.6.2, i.e. the link capacity reduced by the rate of non-bottlenecked flows, equally divided among bottlenecked flows. The latter is estimated by the number of non empty waiting queues, while the rate of non-bottlenecked flows is approximated by the total rate of non shaped flows, estimated using a time window weighted moving average of size ΔT, where we count every non shaped Interest by the size of the corresponding Data packet in bytes. Such size value is deductible from the Interest by using the segmentation information in the name (e.g. in the CCNx prototype [61]), however, a standard component or field would be preferable. The shaping mechanism can be implemented by using virtual times (based on the fair rate) or by round robin based implementation. The latter is

detailed in the pseudo-code of Algo. 1 .

6.3 HBH-ICP analysis

In this section, we show that the presented HBH mechanism coupled with ICP (see Chapter 5) is stable and converges to the same equilibrium of ICP (without shapers) in steady state on average and in absence of packet losses. Such result is important to state that proactive shaping of Interests does not introduce additional delay and does not prevent, in absence of losses, the convergence to the efficient and fair average throughput equilibrium guaranteed by ICP. Let us consider a single flow traversing the network in Fig.6.1 composed by N hops between the user and the repository. Each intermediate node implements the Interest shaping mechanism described in Sec.6.2.

We describe the instantaneous time evolution of the rate $X(t)$, of a flow controlled by ICP at the receiver, of the output queues occupancy, $Q_i(t)$ and of the shaping queues occupancy, $Q_i^s(t)$. As in the previous chapter (for the ICP analysis), we express the flow rate $X(t)$ as the average of the rates associated to route i (i.e. sub-path between the user and the i^{th} cache), weighted by the stationary miss probabilities p_i

$$X(t) = \sum_{i=1}^{N} X^s(i,t) \prod_{j=1}^{i-1} p_j(1-p_i).$$

in which $X(i,t)$ is replaced by its "shaped" version, as a result of the Interest shaping performed at node i (see Fig.6.1) $X^s(i,t)$. Notice that the receiver does not implement Interest shaping, $X^s(1,t) \equiv X(1,t)$. The temporal evolution of $X(t)$ driven by ICP (with constant τ) and that of the output queues $Q_i(t)$ is a slight modification of Eq.5.5 in Chapter 5, where we replaced $X(i,t)$ with $X^s(i,t)$.

$$\begin{aligned}
\dot{X}^s(i,t) &= \frac{\eta}{\text{VRTT}(i,t)^2} - \beta X^s(i,t) \mathbb{1}_{\{\text{VRTT}(i,t)=\tau\}}, \\
\dot{Q}_i(t) &= X^s(i,t) - C_i \mathbb{1}_{\{Q_i(t)>0\}} \\
&+ \sum_{l=i+1}^{N} \Big(\prod_{j=i+1}^{l} \mathbb{1}_{\{Q_j(t)=0\}} X^s(l,t) + \mathbb{1}_{\{Q_l(t)>0\}} C_l \Big)
\end{aligned} \tag{6.1}$$

where $\text{VRTT}(i,t) = \sum_{j=1}^{i} \Big(Q_j(t)/C_j + Q_j^s(t)/\phi_j'(t) \Big)$, $\quad i = 1, ..., N$ denotes the round trip time given by the aggregated queuing delay associated to route i, i.e. all links from the receiver to

the i^{th} node. The Interest shaping rate at node i is the max-min fair share $\phi_i(t)$

$$\phi_i(t) = (C_{i+1} - \sum_{j>i, j\in\mathcal{R}_i^{nb}} X^s(j,t))/(N - \|\mathcal{R}_i^{nb}\|), \tag{6.2}$$

defined as the link capacity C_{i+1} minus the rate of non bottlenecked routes at link i, i.e. in $\mathcal{R}_i^{nb} = \{\cup_j, j \neq \underset{i'\ni i}{\text{argmin}}\gamma_{i'}\}$ divided by the number of bottlenecked routes. The evolution of shaping queues is described by

$$\dot{Q}_i^s(t) = \sum_{j>i, j\notin\mathcal{R}_i^{nb}} X^s(j,t) - \phi_i(t)\mathbb{1}_{\{Q_i^s(t)>0\}} \tag{6.3}$$

where $i = 1, \ldots, N-1$. Indeed, Eq.(6.3) describes the shaping performed at node i on the Interest rate of routes bottlenecked link in i, paced to the fair rate $\phi_i(t)$.

Proposition 6.3.1. *In the regime described by Eqs.(6.1)-(6.2)-(6.3), HBH-ICP is stable and attains the max-min fair and efficient equilibrium of ICP described in the previous chapter, i.e. the steady state solution is periodical with average value*

$$\widetilde{X} = \sum_{i=1}^{N}(1-p_i)\prod_{j=1}^{i-1} p_j \widetilde{X}^s(i), \qquad \widetilde{X}^s(i) = \gamma(i) \tag{6.4}$$

where $\gamma(i)$ is the max-min fair rate of route i defined in Chapter 3.

Proof. The proof consists in showing the equivalence between Eqs.(6.1)-(6.2)-(6.3) and the ODEs describing ICP (see Chapter 5), when $Q_i^s(t)$ replace $Q_i(t)$ in the ICP system, $\forall t \geq t^* > 0$, where t^* accounts for the time needed for the shaping queues to fill in, dependently on their initial condition. The proof follows two steps:

1. Under HBH-ICP, $Q_i(t) \leq Q_i(0), \forall t > 0, i = 2, \ldots, N$;

2. $\forall\, t > t^*$, $Q_1(t) = Q_1^{ICP}(t)$ and $Q_i^s(t) \equiv Q_{i+1}^{ICP}(t)$.

1. Let us focus on the output queues $Q_i(t)$ under HBH-ICP. Due to the Interest shaping performed at node N, the input rate of queue Q_N at time t is $X_N^s(t)$ and by definition of the shaping rate $\phi_i(t)$, $X_N^s(t) \leq C_N$. Thus, $\dot{Q}_N < 0$ and $Q_N(t) \leq Q_N(0), \forall t > 0$. Similarly for queue $Q_{N-1}(t)$,

$$X_N^s(t)\mathbb{1}_{\{Q_N(t)=0\}} + C_N\mathbb{1}_{\{Q_N(t)>0\}} + X_{N-1}^s(t) \leq C_N + (C_{N-1} - C_N) = C_{N-1}$$

For $Q_i(t)$, $i > 2$ in general

$$\sum_{l=i+1}^{N} \Big(\prod_{j=i+1}^{l} \mathbb{1}_{\{Q_j(t)=0\}} X^s(j,t) + \mathbb{1}_{\{Q_l(t)>0\}} C_l \Big) + X_i^s(t) \leq C_i + (C_{i+1} - C_i) = C_i$$

Notice that the inequality does not hold for $Q_1(t)$, since $X(1,t)$ is not shaped.

2. Let us focus on a three links scenario as done in the previous chapter for the ICP multi-bottleneck analysis. As for ICP, the generalization to the case of $N > 3$ is straightforward, but lengthy (we need to distinguish more cases to explicitly compute the max-min rate). Under HBH-ICP, the queues evolution satisfies the following ODEs according to the configurations listed below.

Case I): $C_1 < C_2, C_3$ - In this case all routes are bottlenecked at link 1, therefore there is no shaping at hop 2 and 3 and

$$\dot{Q}_1(t) = X(t) - C_1 \mathbb{1}_{\{Q_1(t)>0\}}, \quad Q_1^s(t) = Q_2^s(t) = 0$$

like for ICP only. This leads to $\widetilde{X}(i) = C_1$, $i = 1, 2, 3$, if we consider a single flow in isolation.

Case II): $C_2 < C_1, C_3$ - In this case route 2 and 3 are bottlenecked at link 2,thus the shaping rate for the two at the first hop is $\phi_1(t) = C_2$.

$$\begin{aligned} \dot{Q}_1(t)dt &= X(1,t) + C_2 - C_1 \mathbb{1}_{\{Q_1(t)>0\}}, \\ \dot{Q}_1^s(t) &= X(2,t) + X(3,t) - C_2 \mathbb{1}_{\{Q_1^s(t)>0\}}, \quad Q_2^s(t) = 0. \end{aligned}$$

It suffices to compare the ODE of $Q_1^s(t)$ and that of $Q_2(t)$ in ICP, to conclude about their equivalence and, hence, about the equivalence of their solution.

Case III): $C_3 < C_2 < C_1$ - In this configuration, the ODEs of HBH-ICP reduces to:

$$\begin{aligned} \dot{Q}_1(t) &= X(1,t) + C_2 - C_1 \mathbb{1}_{\{Q_1^s(t)>0\}}, \\ \dot{Q}_1^s(t) &= X(2,t) - (C_3 - C_2) \mathbb{1}_{\{Q_1^s(t)>0\}}, \\ \dot{Q}_2^s(t) &= X(3,t) - C_3 \mathbb{1}_{\{Q_2^s(t)>0\}}. \end{aligned}$$

where we used $\phi_1(t) = C_2 - C_3$ and $\phi_2(t) = C_3$ as route 3 is bottlenecked at link 3. This implies $Q_1^s(t) = Q_2^{ICP}(t)$, $Q_2^s(t) = Q_3^{ICP}(t)$.

Case IV): $C_3 < C_1 < C_2$ - Similarly to previous cases, we can specialize Eqs.(6.1)-(6.2)-(6.3),

$$\dot{Q}_1(t) = X(1,t) + X(2,t) + C_3 - C_1 \mathbb{1}_{\{Q_1(t)>0\}},$$
$$\dot{Q}_2^s(t) = X(3,t) - C_3 \mathbb{1}_{\{Q_2^s(t)>0\}}.$$

This implies $Q_1^s(t) = Q_2^{ICP}(t)$, $Q_2^s(t) = Q_3^{ICP}(t)$. □

6.4 Evaluation

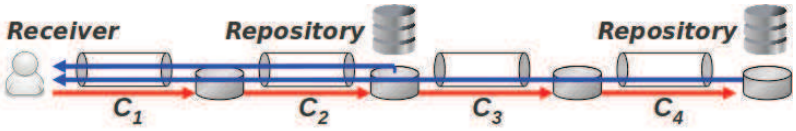


Figure 6.2: Network topology 1: nodes 1,2; network topology 2: nodes 1,..,4.

To assess the Hop By Hop interest shaper performance, we use the CCNPL-Sim described in Chapter 9 implementing the Interest shaping mechanism described in Sec. 6.2. In this section, we gather a selected set of simulation results illustrating HBH-ICP functioning, properties and benefits.

Scenario 1.*Hop-by-Hop Interest shaping is not enough.*
In this scenario we show i) the need for hop-by-hop Interest shaping to be coupled with a receiver-driven Interest control adapting Interest window over time (as ICP),ii) the loss rate reduction realized by our Interest shaping.

To this aim, we consider a two hops network as in Fig.6.2 and focus on a single data retrieval for a content item of 5MB. We set link capacities to (C_1, C_2)=$(100, 40)$ Mbps, propagation delay $1ms$, 10kB data buffers and Interest shaper with burst size B =2kB, ΔT = 100ms, and infinite queuing capacity (large enough to have no Interest discard). We compare two receivers: one implementing ICP and the other using a constant Interest window W and a constant Interest re-expression timer set to 10ms. Data packet size is 1kB, while Interest packet size is 25B. The results of the comparison are reported in Tab.6.1. Using ICP at the receiver the delivery time is optimal, equal to 1s and, given the absence of losses, the same average behavior is observed with and without Interest shaping, as proved in Sec.6.3.

W	Delivery Time[s]		Throughput[Mbps]		Losses[%]	
	w HbH	w/o	w	w/o	w	w/o
2	2.42	2.42	16.30	16.30	0	0
10	1.00	1.00	39.70	39.60	0	0
15	1.00	2.08	39.60	19.20	0	11.20
20	1.00	1.90	39.60	20.90	0	15.30
ICP	1.00	1.00	39.80	39.80	0	0

Table 6.1: Scenario 1. ICP - fixed window comparison.

On the contrary, in presence of a constant Interest window, the delivery performance depends on the W value. A small constant window value W causes inefficient content retrieval due to resources under-utilization (i.e., for $W = 2$, the delivery time is 2.42s). With $W = 10$ the link is efficiently utilized and the delivery time is 1.0s, while increasing W to $15 - 20$ induces Data packet losses and leads to higher delivery time, when no hop-by-hop mechanism is used.

In this case, Interest shaping allows to reduce loss rates by queuing Interest packets before the bottleneck. However, we conclude that the shaping mechanism itself is not sufficient to guarantee optimal delivery performance, as the optimal value of W is not known a-priori and should vary according to available network resources.

Scenario 2.*The benefits of HBH-ICP over ICP*

We consider a second simulation scenario, in the same network set-up described above. By varying the number of flows over time as shown in Fig.6.3 and 6.4, we observe:

- how HBH-ICP queues Interests before the bottleneck link;
- how it reacts to a congestion phenomenon arising at a given time instant.

A single ICP flow starts at time zero, while a second one begins at time 0.5s. In the same simulation run, a greedy CBR flow arrives at t=1.0s, with demand rate corresponding to the link capacity $C_2 = 40$Mbps, sharing the downlink capacity of the bottlenecked link with the two ICP flows.

Before the arrival of the CBR flow, no losses are observed and, as expected from the analytical predictions in Sec.6.3, ICP and HBH-ICP reach the same stationary regime, given by the max-min rate allocation. This can be observed by comparing the time evolution of $Q_2(t)$ and $Q_1^s(t)$, respectively for ICP and HBH-ICP, in Fig.6.3 and 6.4. Under ICP, the output FIFO queue, $Q_2(t)$ is displayed, while under HBH-ICP we report the two per-flow queues in the up-link, $Q_1^s(t)$, since $Q_2(t) = 0$. The total queuing is the same in the two cases, though in HBH-ICP the buffering is

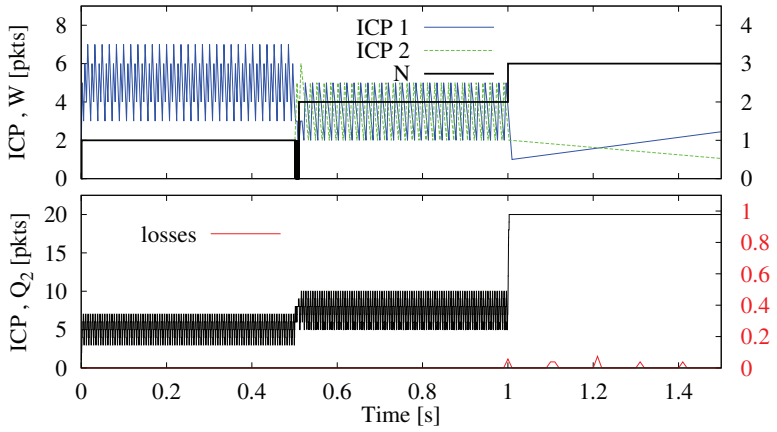


Figure 6.3: Scenario 2. ICP without HBH interest shaping.

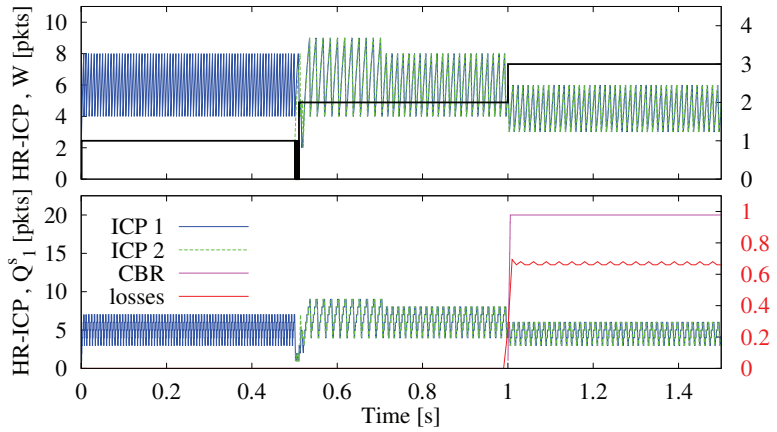


Figure 6.4: Scenario 2. ICP with HBH interest shaping.

distributed in two per-flow buffers scheduling Interest transmissions at the fair rate.

At time t=1s, when the CBR flow starts, the two ICP flows observe losses and their windows drastically reduce in absence of Interest shaping. Conversely, with our Interest shaping mechanism, the CBR flow is immediately shaped and gets a penalty of 66% of Interest packets loss rate, achieving a download rate equal to its fair share. In conclusion, HBH-ICP is shown to protect conformant ICP traffic flows from greedy misbehaving receivers sending Interests at a rate higher than their fair rate. Indeed, HBH-ICP prevents all ICP Data packet losses at the bottleneck, by early reacting to the congestion caused by the greedy CBR flow, dropping interest packets instead.

Scenario 3. *HBH-ICP in a multi-bottleneck scenario.*

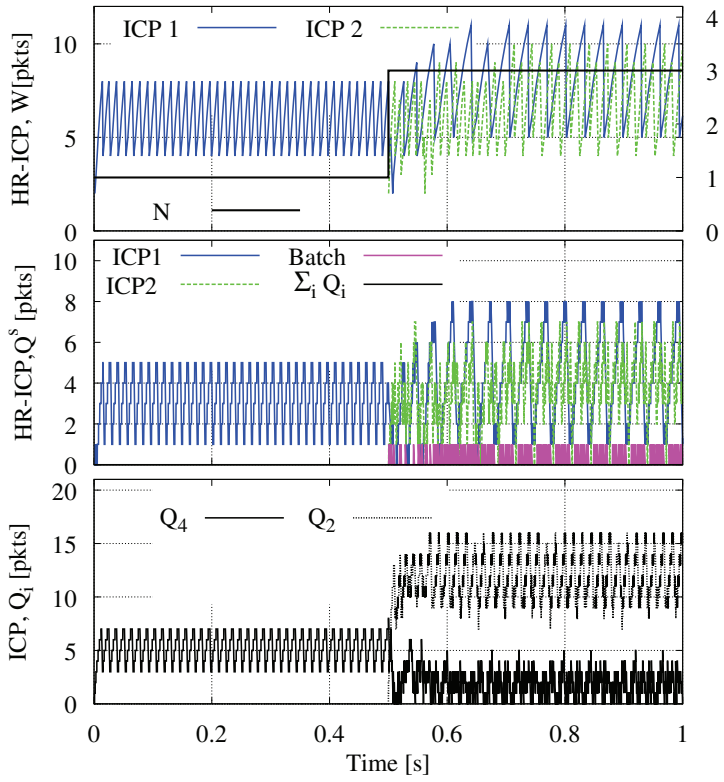


Figure 6.5: Scenario 3.

In the third simulation set, we consider four links including an additional bandwidth bottleneck as in Fig.6.2. The other parameters remains equal to the previous simulation scenario while added links are of capacity (C_3,C_4)=(20, 100) Mbps. Two ICP flows retrieve Data from the second and fourth nodes, respectively. The first ICP flow, starting at time 0, is bottlenecked at link 4, whereas the second, starting at t=0.5s, is bottlenecked at link 2. As a first observation, HBH-ICP creates Interest queuing for each flow before its bottleneck, by leaving empty the shaper queues at other nodes and all Data output queues. Therefore, in Fig.6.5 we plot $Q_3^s(t)$ for ICP 1 and $Q_1^s(t)$ for ICP 2 under HBH-ICP, which correspond to $Q_4(t)$ and $Q_2(t)$ under ICP only. The evolution of the windows is almost the same with and without Interest shaping and coincides with that reported in Fig.6.5. After the first 0.5s, we introduce a new flow, sending four Interests in

batch every 10ms, which corresponds to an average Data delivery of 5Mbps, with a peak rate of 100Mbps. The Interest window of the two ICP flows is slightly reduced according to the fair share reduction induced by the new flow, but their rate is almost not affected. In the HBH-ICP system, with hop-by-Hop Interest shaping, the new flow gets priority along the request path and does not experience queuing in the Data path. Instead, in absence of Interest shaping, the new flow experiences a significant queuing along the Data path given by $Q_2 + Q_4$. Such example shows a desirable property of our shaping mechanism, i.e. prioritization of real-time and delay sensitive traffic, whose rate is lower than the fair rate.

6.5 Conclusion

In this chapter, we introduced an Interest control mechanism for CCN, HBH (coupled with ICP), jointly realized by the end-user and within the network by in-path routers. We believe that Interest control is an essential building block of the CCN architecture, whose definition needs to exploit the specific features of Content-Centric communication. We have shown that the receiver plays an important role in the rate control loop to guarantee full bandwidth/ storage resources utilization and flow fairness. Hop-by-hop Interest shaping enhances rate and congestion control performance and it is particularly suited to CCN for various reasons.

Interest not Data control. Controlling Interests instead of Data packets, gives the opportunity to prevent congestion onset by delaying Interest forwarding. Interest packets are smaller in size than Data packets, hence they require smaller buffer capacity and with reasonable buffer dimensioning Interest losses may be avoided, except in case of greedy misbehaving receivers.

Early congestion detection. It allows to realize early congestion detection by locally monitoring at each node the per-flow Interest rate in uplink and the corresponding Data fair rate on the downlink. A discrepancy between an Interest rate larger than its associated Data rate signals the beginning of a congestion phenomenon, before the detection of packet losses at the receiver.

Protection from misbehaving receivers. By shaping Interest in a hop-by-hop fashion, the greedy behavior of a non-conformant receiver can be fast detected and controlled in order to protect concurrent flows. HBH-ICP and the solution proposed in [59] react to misbehaving receivers by queuing Interests up to a certain threshold, before discarding them. In [60], instead, all Interests exceeding in rate the corresponding Data fair rate are directly discarded. Compared to alternative solutions for CCN Interest control [59], [60] our proposal brings additional benefits due to: (i) the coupling with a rate/fairness optimal receiver control, (ii) Interest shaping at output interfaces, (iii) max-min rate by only controlling Interests.

Scalability/Feasibility. HBH-ICP exploits the PIT information about ongoing flows, by just introducing a credit counter per active flow. Thus, scalability is bounded by the PIT size, dif-

ferently from the solution presented in [59] where the estimation of the response delay $A(t)$ is required per each flow based on an history of samples.

Delay-sensitive flows protection. HBH-ICP brings a considerable reduction of the Data queuing at the bottleneck, by queuing Interests at output buffers (ideally, under the fluid model assumption the queue totally empties). As a consequence, delay-sensitive flows like streaming or voice flows, whose average rate is lower than the fair rate see no Interest queuing and low/zero Data queuing delay due to the Interest shaping of bottlenecked flows. This is a distinctive of our Interest shaping mechanism w.r.t. the solution proposed in [59], where Interests are FIFO queued and every ongoing flow experiences the overall Data queuing delay.

No Interest losses. We have observed that, under a reasonable dimensioning of the output buffer, no Interest is dropped with HBH-ICP, except for misbehaving flows. Conversely, in [60], overload control is managed on a per-flow basis through Interest discard. The incoming Interest rate, at every CCN router, is limited to the fair rate realized by the corresponding Data packets at the bottleneck, under the assumption of a per-flow scheduler implemented at each network node. Interests are discarded as long as the Interest rate exceeds the bottleneck fair rate and before any router processing (CS/PIT/FIB). We argue that this kind of Interest drop should be avoided, as it prevents the removal of the associated PIT entry at the current node (PIT not yet accessed) and at previous nodes. If the PIT entries corresponding to the discarded Interest in previous nodes are not removed, subsequent re-expressions of the Interest are filtered, as the first ones were ongoing until their expiration. In[60] authors propose an explicit signaling of the Interest packet loss back to the user, though it would require message prioritization and it would rise security concerns. Similarly, in [59], where a receiver-driven control of the Interest lacks, authors claim that some back-pressure mechanism is needed to avoid Interest losses, leaving its definition for future work.

Additional traffic control opportunities. By implementing Interest shaping at the output interfaces, HBH-ICP opens additional traffic control opportunities. As a consequence of an Interest drop at a given node, our mechanism permits to trigger the removal of the corresponding entry in the local PIT or the forwarding of the Interest of another feasible interface, when available in the FIB. In the latter case, a *congestion-aware load balancing of the Interests* may be envisaged. Notice that Interest redirection is not possible when the Interest is discarded after the forwarding, like in [60]. Also, a *measurement-based admission control* can be implemented by selectively refusing Interests of new flows, while not penalizing already ongoing flows. A natural extension of this work is the definition of traffic control mechanisms for the management of a multi-path communication, either at the receiver and hop-by-hop as coupled with an Interest forwarding policy.

Chapter 7

Multipath transport protocol and request forwarding

In this section, we introduce and analyze a receiver-driven congestion control mechanism for CCN called Remote Adaptive Active Queue Management (RAAQM), extending our initial ICP design. Furthermore, we develop a distributed algorithm for dynamic request forwarding at CCN nodes.

7.1 Related Works

There is a large body of work on joint multipath routing and congestion control with the twofold objective to study the stability of multipath controllers as in [62], [63] and to propose transport protocol implementations based on TCP (mTCP [64], MPTCP [65], EWTCP [66], etc).

Like most of multipath transport proposals, [64] introduced by Zhang et al., defines uncoupled congestion control on each path implemented through separate connections (subflows) independently managed. Each subflow maintains a congestion window as in TCP and perform its own path congestion control. In fact, it is shown that one global congestion window for the entire flow may lead the throughput of the whole flow being lower than that of a single-path TCP flow on a single path. However the main drawback of uncoupled multipath congestion control consists in unefficient/unfair control of shared bottlenecks. To alleviate congestion over paths shared by multiple subflows, mTCP tries to suppress some of them.

Some path correlation is introduced in EWTCP [66] by Eggert et al. in order to overcome the unfairness and inefficient control of shared bottlenecks with no need for explicit detection of shared bottlenecks. EWTCP proposes a weighted version of multipath TCP, where each subflow increases its congestion window proportionally to a weight, dynamically updated according to the resource utilization of all subflows.

A theoretical analysis of the properties of the optimal multipath controller is provided in [62] and [63]. Specifically, in [62] Towsley et al. study a class of algorithms implemented at the source to stably and optimally split a flow between multiple source-destination pairs while accounting for path correlation. Such multi-path routing/ congestion control algorithms, accounting for are also proved to outperform a single-path congestion control scheme.

The same problem of dynamic route selection on a collection of available paths is tackled in [63] by Kelly et al, where authors claim that it should be performed at transport layer jointly with congestion control. Load balancing decisions taken by the source are based on path measurements of the round trip delay and integrated in a Scalable TCP like controller. As authors underline, in presence of heterogeneous paths, RTT estimation must be accomplished independently on each path in order to obtain good measurements. The main difference w.r.t. our solution is that route selection is performed by the source only, whilst we propose a distributed request forwarding algorithm implemented at each CCN node.

Based on the analysis in [62],[63], Multipath TCP [65] as proposed by the IETF working group MPTC and introduced by Wischik et al., improves previous designs in terms of efficient network usage utilization and of RTT unfairness compensation. The work also address protocol issues related to the problem of opening and maintaining multiple coupled connections and provides a currently standardized protocol implementation.

7.2 Congestion control mechanism

The initial transport protocol we proposed, ICP, is based on an Additive Increase Multiplicative Decrease (AIMD) Interest window controller and relies on round trip delay measurements that are compared to an adaptive threshold to detect congestion. From TCP literature (see e.g.[67]), it is well known that an AIMD closed-loop control is stable and efficient under proper parameters tuning. However, a simple adaptive threshold is not easy to set when RTT estimation varies significantly. In ICP, a window decrease is deterministically applied when a round trip delay measurement exceeds the threshold τ (see Chapter 5), regularly adapted through monitoring of the average round trip delay.

The simulations we carried out outline two limitations of such approach:

1. deterministic window decreases are sensitive to delay estimation errors (during a congestion phenomenon such problem is worsened by the absence of delay samples);

2. the difficulty to adapt the threshold τ in presence of multipath communication and heterogeneous path delays.

For these reasons, we introduce a probabilistic window decrease mechanism based on Remote Adaptive Active Queue Management (RAAQM), that we further prove to be stable and efficient. The motivation behind the introduction of a RAAQM-based window decrease is to anticipate window decrease as long as a variation of the monitored round trip delay is sensed, and to realize a smoother decrease of the congestion window by tuning RAAQM parameters. The next section focuses on the core of the proposed congestion control scheme.

AIMD Interest Control with RAAQM

Let us describe the proposed congestion control mechanism. Data are requested via Interests (one Interest per Data packet) in the order decided by the application, according to a window-based *AIMD* mechanism. The congestion window, W, is kept at the receiver and defines the maximum number of outstanding Interests the receiver is allowed to send as in ICP. The interest window W is increased by η/W upon each Data packet reception. This corresponds to an increment of η ($\eta = 1$ by default) each time a complete window of Interests is acknowledged by Data reception.

Remote Adaptive Active Queue Management

When an Interest is sent out, the receiver measures the instantaneous round trip delay as the time elapsed until the reception of the corresponding Data packet. The receiver estimates minimum and maximum round trip delay ($R_{\min}$ and $R_{\max}$), maintaining an history of samples (a sliding window of 30 by default in our implementation). A complete window of samples is required before triggering a window decrease and lost Interest/Data packets are not considered in the delay estimation. The measured instantaneous round trip delay $R(t)$ and the estimates of $R_{\min}$ and $R_{\max}$ concur to determine over time the probability p of a window decrease. The probability p, is assumed to be a monotonically increasing function of $R(t)$ (as in Fig.7.1), going from a minimum value $p_{\min}$ (10^{-5} in our settings) up to a maximum value $p_{\max} \leq 1$ when above $R_{\max}$. The evolution of p is represented by the following equation:

$$p(t) = p_{\min} + \Delta p_{\max} \frac{R(t) - R_{\min}(t)}{R_{\max}(t) - R_{\min}(t)} \tag{7.1}$$

with $\Delta p_{\max} = p_{\max} - p_{\min}$. $R_{\min}(t), R_{\max}(t)$ indicate estimates at time t. For the ease of notation we will omit the time indication and use $\Delta R_{\max} = R_{\max} - R_{\min}$. Whenever $R_{\min} = R_{\max}$ in a window of samples, we take a decrease probability equal to $p_{\min}$.

In this way, we realize a *Remote Adaptive Active Queue Management*, building upon RAQM [68] previously introduced by Ardelean et al. , that allows to control the Interest window at the

receiver based on the bottleneck queuing delay, as inferred by the measured round trip delay over time.In case of window decrease, W is multiplied by a decrease factor $\beta < 1$.
In analogy with the fluid representation of ICP dynamics in Chapter 5, the modified window update can be described by the following ODEs, where $W(t)$ represents the continuous version of the congestion window value:

$$\dot{W}(t) = \frac{\eta}{R(t)} - \beta W(t) \frac{p(t - R(t))W(t - R(t))}{R(t - R(t))}, \tag{7.2}$$

If the additive increase term remains unchanged ($\frac{\eta}{R(t)}$), the multiplicative decrease $\beta W(t)$ occurs with dropping rate $\frac{p(t-R(t))W(t-R(t))}{R(t-R(t))}$. The rationale behind is that each incoming Data packet sent in the previous round trip time triggers a window drop with probability $p(t - R(t))$, computed based on the round trip delay estimates of the previous round trip time. A similar fluid representation of window dynamics under RED-like AQM can be found in [67].

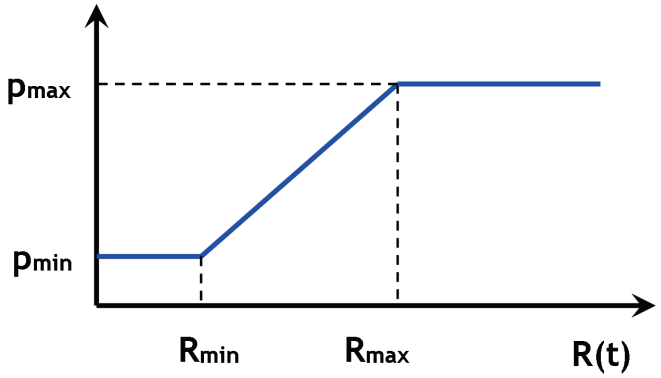


Figure 7.1: Window decrease probability.

Additional Features

Slow Start

To accelerate the transient phase of Interest Window growth in Congestion Avoidance phase, we introduce a slow start mechanism like in TCP where the window follows a multiplicative increase of 1 per data packet received. The slow start phase ends when $R_{max} - R_{min}$ is above a given threshold (as a default a transmission delay equal to 5 packets in queue), and is repeated as long as the monitored R_{min} and R_{max} coincide in a window of samples.

Interest Retransmission

Interest re-expression after a time-out is necessary to recover losses, even though Data may just be delayed at the bottleneck. At the same time, retransmitted Interests sent just after a time-out can be filtered, if the PIT timer is bigger than the Interest timer at the receiver. The PIT timer is set to limit the number of pending Interests. The larger this value the higher the number of Interests not forwarded upstream towards content repository. On the other hand, a small PIT timer implies a large number of unnecessary retransmissions, since delayed packets arriving after PIT time-outs are discarded. In our settings, we consider that the retransmission timer at the receiver is equal to the PIT timer, that we assume as a default set to 1 s.

Single Path Analysis

$R(t)$ dynamic reflects the evolution of the bottleneck queue remotely controlled by the RAAQM algorithm. To describe queue and window/rate dynamics, let us focus on a single path, relying a user to a content repository through a sequence of hops and characterized by a capacity C (the bottleneck link capacity). The presence of caches at intermediate hops would make the communication naturally point-to-multipoint. We will consider such case in the next sections, while here we neglect intermediate caches and focus on the evolution of N flows (content retrievals) over a single path. Each flow is associated to a congestion window $W_i(t)$, $i = 1, \ldots, N$ and a corresponding rate $X_i(t) = W_i(t)/R(t)$, assumed to be proportional to the congestion window like for TCP by virtue of Little's law. System dynamics can be described through the following fluid ODEs:

$$\dot{X}_i(t) = \frac{\eta}{R(t)^2} - \beta X_i(t) X_i(t - R(t)) p(t - R(t)), \quad \forall i \tag{7.3}$$

$$\dot{Q}(t) = \sum_{i=1}^{N} X_i(t) - C, \quad R(t) = R_{\min} + Q(t)/C, \tag{7.4}$$

$$p(t) = p_{\min} + \Delta p_{\max} \frac{Q(t)/C}{\Delta R_{\max}}. \tag{7.5}$$

Note that $R_{\min}$ and $R_{\max}$ are assumed to be constant and equal to the minimum and maximum delay estimates over a history of round trip delay estimates. As in [67] we approximate $R(t)$ with a constant value, $\bar{R}$, equal to the average value. The resulting rate evolution is,

$$\dot{X}_i(t) = \frac{\eta}{\bar{R}^2} - \beta X_i(t) X_i(t - \bar{R}) p(t - \bar{R}) \tag{7.6}$$

The equilibrium point of Eqs.(7.4)-(7.5)-(7.6) is

$$\widetilde{X} = C/N, \quad \widetilde{p} = \frac{\eta N^2}{\beta \bar{R}^2 C^2}, \quad \frac{\widetilde{Q}}{C} = \frac{(\widetilde{p} - p_{\min})\Delta R_{\max}}{\Delta p_{\max}} \tag{7.7}$$

Single path stability

The stability analysis of AIMD congestion control with AQM has been previously studied in the literature in [67] by Hollot et al. Similar arguments allow us to prove the stability of Eqs.(7.4)-(7.5)-(7.6), in absence and presence of feedback delays.

Proposition 7.2.1. *Given the system of ODEs in Eqs.(7.4)-(7.5)-(7.6) in absence of delays, i.e. where the rate evolution is given by*

$$\dot{X}_i(t) = \frac{\eta}{\bar{R}^2} - \beta X_i^2(t) p(t) \quad \forall i, \tag{7.8}$$

Eq.(7.7) is a global asymptotically stable equilibrium $\forall \beta > 0$, $\Delta p_{\max} > 0$. *In presence of delays, where the rate evolution is described by Eq.(7.6), Eq.(7.7) is a locally asymptotically stable equilibrium. A region of attraction for the equilibrium is represented by:*

$$(X_1(t), ..., X_N(t), p(t)) : X_i < \widetilde{X}_i \left(1 + \frac{\kappa \bar{R}\sqrt{\zeta}}{p_{min}}\right), \forall i$$

Proof. The proof is reported in App.C. □

Multipath Extension

The designed congestion control mechanism enables a remote control of one bottleneck queue along the path. In presence of multiple caches over a single path and multiple paths (not necessarily disjoint), the application of RAAQM needs careful analysis.

Let us denote with *route* any unique sequence of nodes linking a user to a potential Data source (hitting cache or repository). We consider that each CCN node takes forwarding decisions by selecting an interface on a per-packet basis, according to a per interface and per prefix metric with no coordination with other nodes. Independently from the specific forwarding policy implemented by CCN nodes, let us focus on congestion control performed at the receiver when multiple routes exist.

A straightforward extension of single path congestion control to the multipath case would be to keep a unique Interest window W, to gather round trip delay measurements coming from all

different nodes and to exploit all RTT samples to compute a unique window decrease probability p. In this setting, the probabilistic decrease profile would refer to a global queuing delay averaged over possibly heterogeneous queuing delay samples related to different routes, while the estimated minimum and maximum round trip delay, $R_{\min}, R_{\max}$ would represent respectively the minimum and maximum delay routes. As confirmed by our simulations, controlling multiple bottlenecks through a unique decrease probability p results in inefficient resource control (cfr. Sec.7.5).

An alternative solution is proposed in literature for Multipath TCP [65] by Wischik et al. MPTCP controls different paths through separate congestion windows, as they are independent connections, whose evolutions are coupled by a global congestion window term. However, such approach seems hardly applicable in CCN where the number of routes is not known a priori and can vary according to the popularity of the requested item, also unknown to the receiver.

Therefore, we opt for a unique congestion window of Interests, kept at the receiver, with separate per-route RAAQM mechanisms. This add some requirements to our congestion protocol design:

- *Route labeling*: each Data packet must carry a route label composed by the sequence of traversed node identifiers, starting with the Data source. The node identifier can be obtained in different ways such as MAC addresses or a CCN name space dedicated to node for network management functionalities);

- *Route delay monitoring*: the receiver must keep separate estimates of the minimum and maximum round trip delay observed on a given route by distinguishing the samples according to their route label. Delay estimates are kept only for those routes associated to a minimum number of samples (e.g. a window of 30) has been collected;

- *Route probabilistic decrease*. Based on the route delay estimates, a per-route window decrease probability p_r, is computed.

If we denote by $\mathcal{R}$ the set of monitored routes, the fluid equation of the window evolution gives:

$$\dot{W}(t) = \frac{\eta}{R(t)} - \beta W(t) \sum_{r \in \mathcal{R}} \frac{p_r(t-R(t)) s_r W(t-R(t))}{R(t-R(t))}, \tag{7.9}$$

where $s_r W(t)$ denotes the fraction of the current window $W(t)$ of Interests routed along route r resulting from the forwarding decisions taken in a distributed way by CCN nodes. Note that the split ratios s_r of the flow over different routes do not need to be known at the receiver, who simply applies the probabilistic decrease with probability p_r at reception of a packet identified by the label r. Further, it is worth observing that the number of monitored routes whose RTT estimations are

kept at the receiver, is limited in practice as indicated by simulation results in Sec.7.5. In fact, more popular items are replicated close to the user, while less popular one are cached far from the user in a few caches/repositories, as they are rarely requested.

7.3 Dynamic Request forwarding

Optimal load balancing of requests would require a global knowledge of the network (all users demand, bandwidth/storage resources, cache hit probabilities) available at each router. In practice, this is not feasible because of the large size of the network (prefixes, routes, users) and of dynamic nature of such information that would require regular updates. Congestion control and request forwarding decisions can only be based on the local (partial) knowledge of the network available at each node via distributed algorithms. In Sec.7.2, we introduced the distributed congestion control implemented on the user-side. In this section, we focus on the distributed algorithm for output interface selection to integrate in the CCN router forwarding engine.

According to the CCN forwarding model, a request (Interest) is forwarded upstream to a next hop when not satisfied by the local cache nor by a pending request present in the PIT. Such operation is performed by using the FIB information for the specific name prefix (if routed) about the list of matching output interfaces. For interface selection purposes, we propose to associate to each output interface a metric reflecting:

1) content proximity, i.e. the overall hit probability at different hops along the paths towards the repository(ies) for a given name prefix and the associated response rime;

2) congestion status, i.e. the quality of the residual network path from that output interface and the repository(ies).

We claim that an indirect measure available at each node of content proximity and congestion status of the residual path towards the repository is the average *number of Pending Interests* (PI) per interface and per name prefix. Such metric can be directly derived from the information stored in the PIT and it can be stored with low overhead within the corresponding FIB entry. Besides the different popularity of items with the same name prefix, the number of PI per prefix and per interface provides an average measure of the residual distance to closest content replicas. Keeping finer information would be unfeasible in terms of additional state to maintain in the FIBs and not necessarily beneficial. Intuitively, the number of PI can be seen as the occupancy of a queue with input rate equal to the Interest rate (per-prefix, per-interface, in upstream) and output rate equal to the corresponding Data rate (downstream). The larger the average queue occupancy, the higher is the indication of congestion and/or of low availability of the content on the path(s)

Algorithm 2 Interest forwarding algorithm

```
1: At Data Packet Reception (name,face_in)
2: if (!PIT_miss) then
3:     Forward_Data(face_out);
4:     FIB_PI_Decrease(face_in, prefix);
5:     FIB_Weight_Update(face_in, prefix);
6: else Drop_Data;
7: end if
8:
9: At Interest Packet Reception(name, face_in)
10: if (CACHE_miss && PIT_miss) then
11:     (prefix,weight_list)=FIB_Lookup(name);
12:     f*=RND_Weight(weight_list);
13:     FWD_I(f*);
14:     PIT_I_Update(name,f*);
15:     FIB_PI_Incr(f*, prefix);
16:     FIB_Weight_Update(f*, prefix);
17: else CCN_standard_processing;
18: end if
19:
20: At PIT_Timeout(name, prefix, face)
21:   FIB_PI_Decr(face, prefix);
22:   FIB_Weight_Update(face, prefix);
23:
24: procedure FIB_WEIGHT_UPDATE(face, prefix)
25:  avg_PI(face, prefix) ← α · avg_PI + (1 − α) PI(face, prefix);
26:  w(face, prefix) ← 1/avg_PI(face, prefix);
27: end procedure
28:
29: procedure FIB_PI_INCR(face, prefix)
30:     PI(face, prefix) + +;
31: end procedure
32:
33: procedure FIB_PI_DECR(face, prefix)
34:     PI(face, prefix) − −;
35: end procedure
```

behind that interface. Therefore, the forwarding decision can be taken based on a weight inversely proportional to such metric.

Let us detail the steps of the algorithm, as described in Algorithm 2. At each Data/Interest packet reception, the number of PI associated to a given FIB entry (thus to a name prefix) and to a particular output interface is updated (FIB_PI_Incr/FIB_PI_Decr). Based on the instantaneous value of PI, a moving average (with parameter 0.1 to weight the new sample) is regularly updated and used to recompute interfaces' weights (FIB_Weight_Update). A random weighted algorithm is used to select the output interface for each incoming Interest (RND_Weight). At the beginning each interface has the same weight equal to one and the forwarding process is completely random. Weights are normalized w.r.t. the sum of all the weights of the interfaces available for a given prefix.

7.4 Optimization framework

Optimal usage of network bandwidth and user throughput maximization depend on request rate control and on the policy adopted for route selection and request forwarding.

In the following, we formulate the corresponding joint multipath rate control and route selection problem (RCRS) under the following assumptions:

- finite link bandwidth: max-min rate fairness criterion between flows over shared links;
- in-network caching: a flow can be terminated at the repository or at an intermediate network cache;
- name-based RIB with name prefix entries already computed per name prefix and stored in the FIB.

Network flow model

The network is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with finite link capacities $C_{ij} > 0$, where C_{ij} denotes the capacity of arc $(i, j) \in \mathcal{A}$. Data flow from the repository(ies) (denoted $\mathcal{S} \subset \mathcal{V}$) to the users (denoted $\mathcal{U} \subset \mathcal{V}$), hence modeling the flows of data (most of the time $|\mathcal{S}| << |\mathcal{U}|$). The variable x_{ij}^k denotes the data rate on link (i, j) (on the reverse path of the Interest). We denote $\Gamma^+(i) = \{j \in \mathcal{V} : (i, j) \in \mathcal{A}\}$ and $\Gamma^-(i) = \{\ell \in \mathcal{V} : (\ell, i) \in \mathcal{A}\}$ respectively the set of egress and ingress nodes for node i. The optimal flow rate allocation through the network provides the useful information for Interest forwarding on the upstream path (from the users to the repositories).

A user $u \in \mathcal{U}$ sends out Interests for retrieving a content item k, $k \in \mathcal{K}$. The item k is identified by a name prefix f, $f \in \mathcal{P}$. The index k hence also identifies the flows (commodities)

of Data. Each network node $i \in \mathcal{V}$ is characterized by a finite size cache and average stationary hit probabilities $p_k(i)$, $\forall k \in \mathcal{K}$, function of content popularity, content size and cache size. We define $h_k(i) \in \{0, 1\}$ the Bernoulli variable equal to 1 with probability $p_k(i)$ in case of hit for a packet of content item i, and 0 otherwise. Like in Sec.3 we impose max-min fairness criterion for bandwidth sharing between flows over shared links.

Optimization Model

In the following, we formulate the RCRS problem as a multi-commodity flow problem with the objective to maximize overall flow rates:

$$\begin{cases} \max z, & (7.10) \\ \sum_{\ell \in \Gamma^-(i)} x^k_{\ell i} = z \qquad \forall i \in \mathcal{U}, k \in \mathcal{K} & (7.11) \\ \sum_{k \in \mathcal{K}} x^k_{ij} \leq C_{ij} \qquad \forall (i,j) \in \mathcal{A} & (7.12) \\ \sum_{\ell \in \Gamma^-(i)} x^k_{\ell i} = \sum_{j \in \Gamma^+(i)} x^k_{ij}(1 - h_k(i)), \quad \forall k, i & (7.13) \\ x^k_{ij} \leq s^f_{ji} \sum_{\ell \in \Gamma^-(j)} x^k_{\ell j} \qquad \forall (i,j) \in \mathcal{A}, f \in \mathcal{F}, k \in \mathcal{K}^f & (7.14) \\ s^f_{ji} \in [0,1], x^k_{ij} \geq 0, & (7.15) \end{cases}$$

This problem can be viewed as a *maximum concurrent flow problem*, with the objective to maximize the total Data rate received by a user i from all interfaces in $\Gamma^-(i)$ (7.11), while satisfying the capacity constraints (7.12). However, there is no flow conservation at the nodes because of the caches that may intercept the flow entering a node i resulting in a cache hit (7.13). The variable s^f_{ji} denotes the split ratio applied on Interest packets at node j towards all nodes $i \in \Gamma^-(j)$ for flows of class k having prefix f. Splitting decisions are common to flows identified by the same name prefix (7.14).

The optimal value of this (RCRS) problem defines the first level of max-min rate, where the same rate is allocated to all flows. To obtain the full max-min rate allocation, one has to solve a series of (RCRS) problems where the rates of some flows are progressively fixed to their threshold value. More precisely, at a given step r, flows for which the common rate z can still be increased (denoted by $\mathcal{K}^r_{free}$) are considered as variables in a single problem where the other flows (denoted by $\mathcal{K}^r_{fixed}$) are fixed to their maximum allowable rate (computed in a previous step). The process stops when no more flow rate can be increased. The successive problems (RCRS^r) are obtained by replacing the constraints (7.11) by:

$$\begin{cases} \sum_{\ell \in \Gamma^-(i)} x_{\ell i}^k = z \qquad \forall i \in \mathcal{U}, k \in \mathcal{K}_{free}^r & (7.16) \\ \sum_{\ell \in \Gamma^-(i)} x_{\ell i}^k = z_{fixed}^k \qquad \forall i \in \mathcal{U}, k \in \mathcal{K}_{fixed}^r & (7.17) \end{cases}$$

Unfortunately, the problem (RCRS) (and (RCRS^r)) is non-linear, because of the product of variables in constraints (7.14) and hence quite difficult to solve. To evaluate its optimal value z^*, we propose to consider two problems providing respectively an upper- bound and a lower-bound.

Computing an upper-bound of z^*

To obtain an upper-bound on z^*, we simply consider the relaxed problem (RCRS$_{up}$) where all constraints (7.14) are removed from (RCRS). The resulting problem is a linear programming problem that can easily be solved, but where each flow can be routed differently. The complete max-min rate allocation is computed as described before by considering a sequence of (RCRS$_{up}^r$) problems, where rate values are progressively fixed to their upper-bound.

Computing a lower-bound of z^*

We compute a lower-bound on z^* by choosing a particular routing pattern and setting the splitting variables accordingly. The chosen routing pattern is the one resulting from (RCRS$_{up}$) when adding up all flow values corresponding to a same prefix:

$$\tilde{s}_{ji}^f = \sum_{k \in \mathcal{K}^f} \tilde{x}_{ij}^k / \sum_{k \in \mathcal{K}^f} \sum_{\ell \in \Gamma^-(j)} \tilde{x}_{\ell j}^k \tag{7.18}$$

where $\tilde{x}_{ij}^k$ are the optimal values returned by (RCRS$_{up}$). The problem (RCRS$_{down}$) is then formed by setting the splitting variables to the constant values $\tilde{s}_{ji}^f$ in (RCRS). The complete max-min rate allocation is computed as before. The optimization problem formulation holds for a general cache replacement policy and under general content size and content popularity assumptions. In our setting, we consider LRU (Least Recently Used) caches, σ-sized items (in number of packets) and Zipf content popularity, i.e.the probability for a user to request content item k is $q_k = c/k^\alpha$, with k referring to a popularity class of m equally popular items, $k = 1, \ldots, K$. Under such assumptions, and $\alpha > 1$, we use the formula proposed in Chapter 2.3 in order to compute the average per-class stationary hit probability that we use here to derive $p_k(i)$, $\forall k \in \mathcal{K}$, $i \in \mathcal{V}$. The solution is averaged over a series of runs of the optimization problem, over realizations of $h^k(i) \in 0, 1$.

Parameters		Infinite Buffer		$B = 100$ pkts	
$\Delta p_{\max}$	β	$\widetilde{Q}$	$\widetilde{W}$	ploss (%)	$\widetilde{W}$
0.5	0.5	21.67	2.36	0	2.36
0.2	0.5	36.93	4.07	0	4.07
0.2	0.2	60.73	6.45	0	6.45
0.1	0.5	54.74	6.11	0	6.11
0.1	0.2	86.97	9.34	0.09	9.32
0.1	0.05	177.84	18.37	2.28	13.34
0.1	0.02	267.90	27.55	4.97	16.96
0.05	0.02	∞	∞	8.05	21.76

Table 7.1: Sensitivity Analysis.

7.5 Performance Evaluation

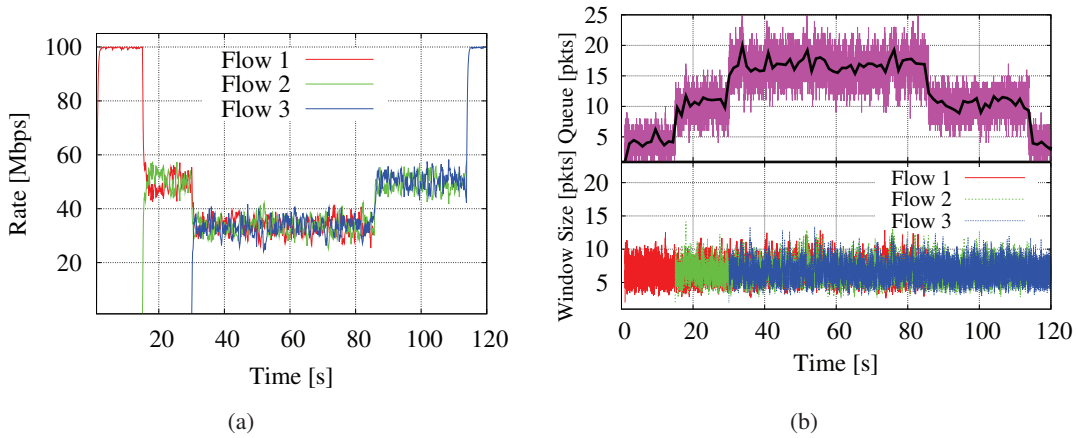
To assess the performance of the proposed RAAQM congestion control and distributed request forwarding we carried out extensive CCN simulations in different network scenarios using CCNPL-Sim (see Chapter 9). In this section, we gather a selected set of simulation results illustrating functioning, properties and benefits of our solution.

Single Path and Sensitivity Analysis

We first consider a single path linking a user to a content repository (characterized by a bottleneck capacity of 100 Mbps and 1ms of propagation delay) and study performance sensitivity to congestion control parameters. The user requests 10 different content items, of the same size, namely 5000 packets of 10KB each (50MB) (default content size in our simulations). The buffer size is initially set to a large value, in order to observe the evolution of queue dynamics under different parameters settings and in absence of packet losses. By varying $\Delta p_{\max} = p_{\max} - p_{\min}$ ($p_{\min} = 10^{-5}$ as default) and β, we observe RAAQM stability properties.

As reported in Tab.7.1, for $\beta \Delta p_{max} > 2 \cdot 10^{-3}$, the queue occupancy and hence the round trip delay stabilizes, while, outside of this region the queue keeps growing over time. Such dependency of the stability from $\beta \Delta p_{\max}$ is in accordance with Prop. 7.2.1. Also, as predicted by Eq.(7.7), the average queue increases as $\beta \Delta p_{\max}$ decreases, while bandwidth is always fully utilized and rate the maximum available capacity of 100 Mbps. Under a finite buffer of size $B = 100$ packets, instability clearly translates into an increasing loss probability (cfr. Tab.7.1), negatively affecting overall performance.

In Fig.7.2(a) and Fig.7.2(b) we report instantaneous throughput, measured as the number of bytes received every 100ms, as well as queue and windows time evolution. By varying the number



(a) (b)

Figure 7.2: Single path Scenario: Rates (a), Windows/Queues evolution (b)

of flows in progress N, we observe the level of efficiency and fairness of the protocol. The user sends out requests for three different items, at 1s, 15s, 30s, respectively for the red, green and blue curve in Fig.7.2(a). Link bandwidth is always fully exploited and fairly shared by ongoing flows.

It is worth observing that queue occupancy reflects the number of flows in progress, since the RAAQM mechanism controls the difference between minimum and maximum RTT estimates that increases at the arrival of the second and third flow respectively. More precisely, as long as the maximum RTT is smaller than the value associated to a full buffer, the queue grows with the number of ongoing flows, N. When close to buffer saturation, the difference between minimum and maximum RTT diminishes leading to a larger per-flow decrease probability, more window drops and hence queue reduction. Efficiency is guaranteed by the fact that the queue never empties assuring full link utilization.

Multipath scenario

In this section, we analyze the multipath network scenario. In particular in Fig.7.3, one user is connected to 4 repositories via 4 non-disjoint paths in a hierarchical topology. Each repository stores the entire content catalog and we neglect intermediate caches, in order to limit the number of routes to 4. The user generates 20 content requests (flows) simultaneously. Link capacities are directly reported in Fig.7.3. Note that the links between user and network, repositories and network, are set to 1Gbps as to avoid bottlenecks at the access and/or at the repositories that would impact our analysis of the multiple paths scenario.

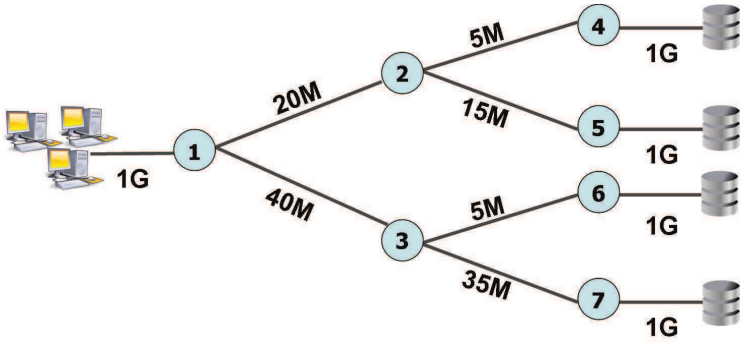


Figure 7.3: Multipath network scenario.

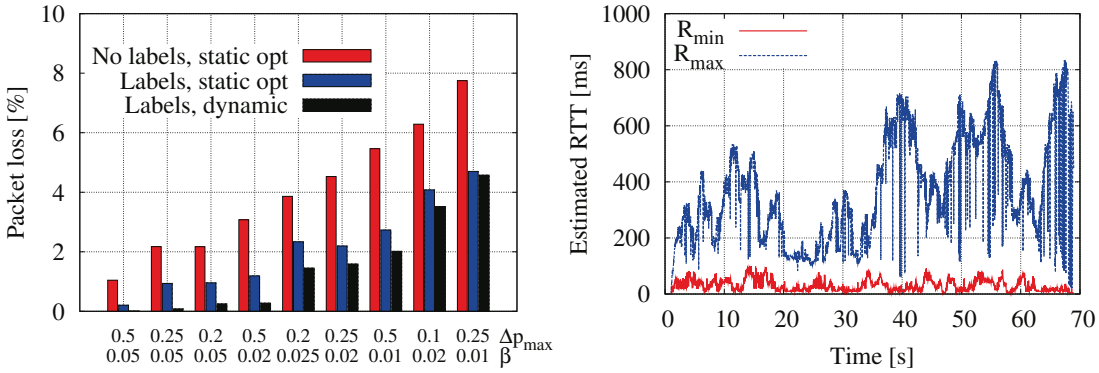


Figure 7.4: Multipath Scenario: Packet Loss (a), RTT estimation without route labels (b).

Route labeling

In the scenario described before, we study the route labeling that we previously introduced (see Sec.7.2). Route labeling is necessary because, in absence of that, the round trip delay estimates (Fig.7.4(b)) in the scenario described above and computed over all undistinguished samples capture a small minimum RTT (from the fastest route, node 7 to the user) and a very high maximum RTT (from the slowest route, node 4 to the user). The large fluctuations of the RTT estimate highlight the poor round trip delay control in absence of route labeling. More, the high value of $R_{\max} - R_{\min}$ results in a too small global decrease probability p and hence in inefficient congestion control, as confirmed by the number of packet losses in Fig.7.4(a). Even if route labels are needed to properly collect round trip delay estimates, they are not sufficient to guarantee optimal throughput if the request forwarding is not optimal as for random request forwarding in Fig.7.5(a). Indeed, under uniformly random interface selection, the average rate over all routes converges

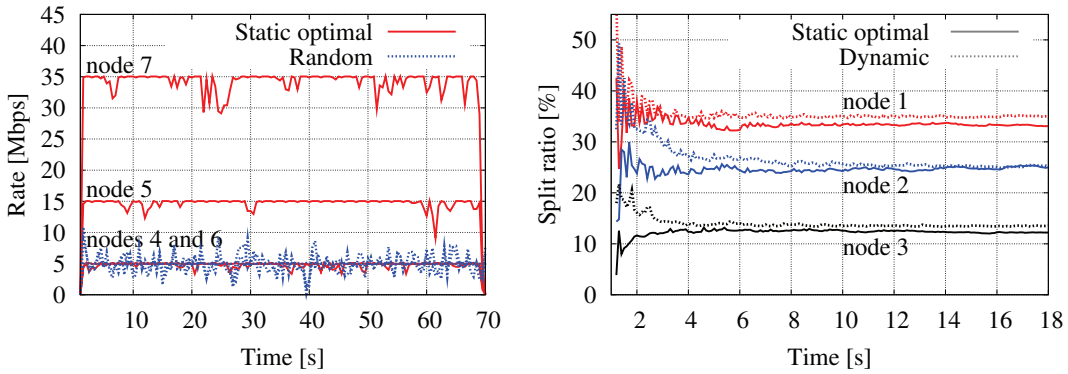


Figure 7.5: Multipath Scenario: Rate comparison (a), Split ratios (b).

to the value of the route with minimum capacity (5Mbps). On the contrary, rates are observed to fully exploit network resources under the optimal split ratios predicted by the solution of the optimization problem and imposed via static per-node and per-interface weights.

Request forwarding - Static optimal weights vs dynamic

In the same multipath scenario of Fig.7.3, we evaluate system performance under our dynamic request forwarding algorithm and compare it with the optimal flow allocation realized by imposing optimal forwarding weights (and hence split ratio) at output interfaces. Notice that in both cases the interface selection is random according to the forwarding weights. This explains the oscillations observed for the optimal split ratios in Fig.7.5(b) until the convergence predicted by the law of the large numbers. The split ratios computed by the dynamic algorithm converge fast to such optimal values. An additional advantage of the proposed algorithm is illustrated in Fig.7.4(a). Its adaptiveness allows to compensate a bad tuning of RAAQM parameters, by reacting to loose congestion control along the paths with a reduction of the weights associated to the congested routes. Indeed, as soon as a much higher number of pending Interests over a given output interface is detected, a dynamic adjustment of forwarding weights balances Interests over low congested interfaces. As a result, in terms of packet loss ratio (number of lost packets over transmitted ones), our algorithm outperforms the static optimal approach.

Network scenario

We focus here on a bigger topology with 15 nodes and 11 routes (Fig.7.6), available to each user to retrieve content from intermediate caches or repositories, and compare the performance

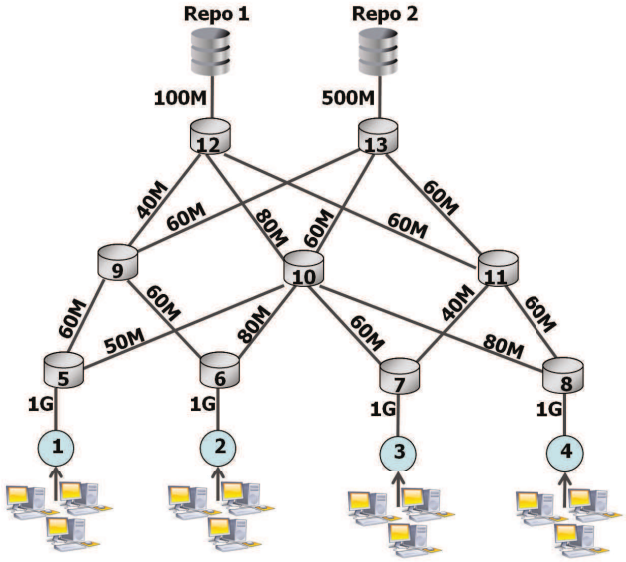


Figure 7.6: Complex network scenario.

of the proposed congestion control and request forwarding algorithms with the optimal allocation (Sec. 7.4). Each repository stores the entire content catalog and intermediate nodes have local LRU caches with sizes 40 000 packets (400 MB), at nodes 5 to 8, 80 000 packets (800MB) at nodes 9-13. Link capacities are indicated in Fig.7.6. Users content requests follow a Poisson process of rate $\lambda = 3.5$ items/s, for content items in a catalog of 20 000 elements, of average size σ=1000 packets (10kB each) and equally partitioned into K=2 000 classes of popularity. We assume a Zipf content popularity distribution with shape parameter α=1.7 and uniform probability between file in the same class. The problem is solved as described in Sec.7.4 using the Xpress optimization suite. We consider a single name prefix, and compute the stationary hit probabilities $p_k(i)$ using the formula obtained in Sec.2.3. The number of active flows per popularity class in each client node, is obtained by estimating the per class load using the formulas obtained in the bandwidth modeling Chapter (Chapter 3). We generate a series of instances for different realizations of the stochastic model described in Sec.7.4, and solve them with the Xpress tool. Results of different instances are averaged to obtain expected split ratios at each node and user throughput per popularity class (table in Fig.7.7 and Fig.7.8(a)). A very good agreement can be observed between the per node and per output interface split ratios achieved in the simulations by using our forwarding algorithm and the optimal values (table in Fig.7.7). In Fig.7.8(a) we show the average user throughput resulting from the dynamic load balancing of Interests according to such split ratios. The throughput is clearly a function of content popularity and more popular classes achieve higher average throughput due to the higher content proximity, besides the fact that the

split ratios are popularity-agnostic (associated to the name prefix). The value of the stationary miss probability obtained via simulation at different network level is reported in fig.7.8(top right) and compared against the expected theoretical value predicted by the LRU storage sharing model presented in Sec.2.3. It is also important to observe that the number of exploited routes is smaller than the maximum number of available routes (equal to 11) and decreases along with content popularity. In Fig.7.8(b) we report two metrics per popularity class k: (i) the total number of exploited routes (averaged over the items in class k) and (ii) the average number of exploited routes computed as the ratio between the total rate and the average route rate, weighting each route by its rate, i.e. $(\sum_{r\in\mathcal{R}} \widetilde{X}_r)^2 / \sum_{r\in\mathcal{R}} \widetilde{X}_r^2$. We remark that the limited number of routes exploited and monitored by the congestion control mechanism confirms the feasibility of a per-route delay monitoring at the receiver. A larger number of routes would not trigger a throughput increase, while making more difficult to control them all efficiently.

Split Ratios (left, right) (%)				
Node id	Optimal		Distributed Alg.	
5	54.9	45.1	55.2	44.8
6	49.0	51.0	49.9	50.1
7	60.0	40.0	59.1	40.9
8	49.0	51.0	49.6	50.4
9	33.1	66.9	32.4	67.6
10	55.4	44.6	55.4	44.6
11	23.0	77.0	30.8	69.2

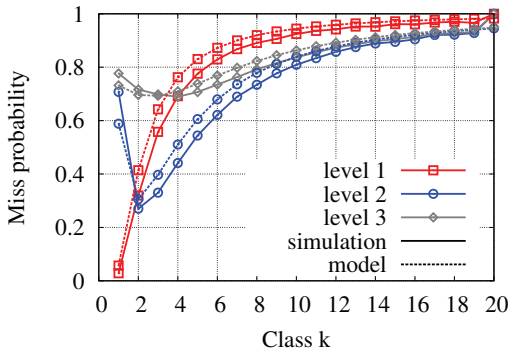


Figure 7.7: Tab. Split Ratios. and Miss probabilities in the network scenario described in 7.6

7.6 Conclusion

In this chapter, we present the design of a joint congestion control and request forwarding mechanism to realize fair and efficient multipath communication in content-centric network. By leveraging the receiver-driven nature of CCN transport, we build a probabilistic window decrease for remote bottleneck queue control fed by per-route round trip delay measurements. In the multipath case, we couple receiver-driven request control with a distributed forwarding algorithm implemented at network nodes and based on the average number of pending Interests per name prefix and per output interface.

We then formulate the underlying max throughput optimization problem handling jointly congestion control and request forwarding in the multipath case, in order to fully exploit network resources and to maximize overall user throughput. While optimality bounds of the proposed

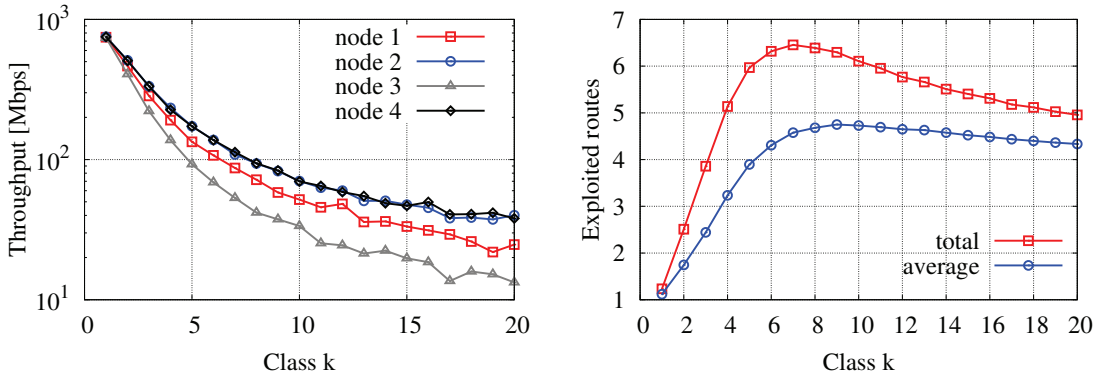


Figure 7.8: Network scenario: User Throughput (bottom left), Exploited Routes (bottom right) in the network scenario described in 7.6.

mechanisms have not been proved performance is assessed by means of CCN packet-level simulations with promising results when compared with the optimal solution.

From the performance evaluation we carried on, the remote active queue management solution applied on labeled routes appears to be a good candidate for enabling robust and largely stable multipath congestion control. More interestingly, when coupled with our dynamic request forwarding the stability region is enlarged as a result of the adaptiveness of the weight update scheme, leading to improved robustness w.r.t. delay variations and parameter setting. We plan to investigate further such phenomenon as well as the properties of our dynamic request forwarding.

Chapter 8

Conclusion of PART II

Content-Centric Networking (CCN) advocates a name-based communication, controlled by the receiver (pull-based), realized via name-based routing and forwarding of user requests in a point-to-multipoint fashion and supported by the presence of a pervasive network-embedded caching infrastructure. The multipath nature of CCN transport holds considerable promises for enhanced flexibility in terms of mobility management, improved end-user performance and network resources utilization. However, the definition of multipath transport control mechanisms adapted to CCN still lacks in the literature.

In this second part of the dissertation, we introduced some mechanisms that aim to bring a first transport protocol design for the CCN architecture. In order to do so, we compared the performance of the designed protocols against the theoretical optimal fair and efficient regime obtained through the performance analysis presented in Part I the .

In particular, in Chapter 5 we presented a first CCN transport protocol design namely ICP, that make use of an interest window W whose evolution is guided by the AIMD mechanism. Furthermore, we proved that ICP converges to the equilibrium described in the CCN performance analysis (developed in Part Part I) and tested the protocol using the CCNPL-Sim (Chapter 9). In addition to ICP, in Chapter 6, we introduced an Hop-by-Hop interest shaper whose objective are to accelerate congestion detection and reaction. We also proved that HBH reaches the same equilibrium of ICP in isolation and analyzed the shaper through some simulations. Finally in Chapter 7, we presented a modification of the original ICP proposition in order to overcome to its limitations in case of heterogeneous network delays. Furthermore, we introduced a per node, per interface and per prefix interest forwarding mechanism and compared it against the optimal forwarding strategy.

Some open questions and problems related to congestion control and interest forwarding mechanisms in CCN are object of current and/or future work. The congestion control protocol

proposed in 7 needs further analysis in order to better understand the stability region and improve the protocol parameter setting. Moreover, complex simulation scenario (or experimentation with the CCNx prototype [61]) can be considered to further prove the scalability of the proposed mechanisms.

Chapter 9

CCN Packet Level Simulator - CCNPL-Sim

In this chapter, we briefly present the Content Centric Network Packet Level Simulator (CCNPL-Sim). The simulator has been published under the GNU GPL v2 license and is available for download at `http://code.google.com/p/ccnpl-sim/`. It includes two main packages cbcbsim and cbnsim, an installation file and some toy examples.

CCNPL-Sim is built upon CBCBSim [69]. In particular, its development started from two packages of the original simulator namely cbcbsim-1.0.2 [1] and cbnsim-1.1.3 [2]. Other packages originally used in CBCBSim such as Simple Simulation Library (ssim), Siena Synthetic Benchmark Generator (ssbg) and Siena Simplifier (ssimp) were not modified and are still maintained by their original authors. The only package that required modifications was the Siena Fast Forwarding (sff) one in order to perform longest prefix match lookup during the forwarding phase. In the installation process sff-1.9.5, ssbg-1.3.5, ssim-1.7.5 and ssimp-1.1.5 packages are downloaded from their original website and a patch for the sff-1.9.5 package is applied.

In order to better present the simulator structure, we divide the chapter in three parts. Sec.9.1, is devoted to present the workload structure, its generation process and the adaptation of the Content Based Network (CBN) naming scheme to the CCN one. Sec.9.2 better explains the real functioning of the simulator. Finally Sec.9.3 briefly illustrates the simulator outputs.

[1] http://www.inf.usi.ch/carzaniga/cbn/routing/cbcbsim-1.0.2.tar.gz
[2] http://www.inf.usi.ch/carzaniga/cbn/routing/cbnsim-1.1.3.tar.gz

9.1 Workload generator

In this section we introduce the workload generator tool that is directly derived by the one used in the original CBCBSim.

Naming adaptation

In order to better understand the workload construction, let us first introduce the naming schema used in the simulator. CBCBSim makes use of the naming defined by the CBN architecture previously introduced while reviewing ICN propositions (see Sec.1.2). In order to be compliant to the current hierarchical URI-like naming schema adopted by CCN, we use the CBN names with some additional constraints.

The CBN names are of the type:

message: [class="warning", severity=1, device-type="hard-drive"]
selection predicate: [(class="warning") $\vee$ (alert-type="software-error" $\wedge$ severity=1)]

in which a message is composed by a set of attribute-value pairs called constraints and the selection predicate is composed by a disjunction (filters) of conjunctions (predicate) of constraints i.e. the selection predicate presented before is composed by two filters: [class="warning"] and [alert-type="software-error" $\wedge$ severity=1].

In order to have URI-like addresses, we impose that:

- each constraint corresponds to an element of the URI address i.e. /constraint1/constrant2/../
- constraints are listed in alphabetical order to represent a URI-like address i.e. the message A = "provider", B ="video", C = "video.mpg" is converted in /provider/video/video.mpg. Notice that also numbers can be used as constraints' name.
- selection predicates (prefixes used in the interest forwarding process) are composed by a single filter.

Workload syntax

The workload file is composed by a set of commands that we briefly introduce in this section. Here is the list of the accepted commands in the workload with their syntax and an example:

- **time_unit**: indicates the workload's time unit i.e. if the time unit is 0.0000010000 all the times in the workload file are indicated in μs.
  Example:

time_unit 0.0000010000 ;

- **sim_length**: indicates the length of the simulation expressed using the workload's time unit. Notice that the simulation length is overridden if a length is specified to the simulator.

  Example:

  sim_length 500000000000.0000000000 ;

- **publish_content**: indicates that the specified node in the topology will be the server for the specified content. Notice that a file can be published in different network nodes.
  Example:

  publish_content 0 0.0000000000 100 8000 1 100 0 A = "provider" B = "video.mpg" ;

column	syntax	example
1	command	publish_content
2	node_id	0
3	publication time	0.0000000000
4	file size [packets]	100
5	packet size [bit]	8000
6	class_id (popularity class in our settings)	1
7,8	unused fields	100 0
9, .., ;	content name	A = "provider" B = "video.mpg" ;

- **download_content**: indicates the beginning of the downloading process of the specified content from a particular node.

  Example:

  download_content 2 1000000.0000000000 0 A = "provider" B = "video.mpg" ;

column	syntax	example
1	command	download_content
2	node_id	2
3	download time	1000000.0000000000
4	unused field	0
5, .., ;	content prefix	A = "provider" B = "video.mpg" ;

- **set_predicate**: this command has been inherited from the CBCBSim. It constructs the shortest path to deliver interest messages from all the potential requester to the node indicated by the command. The command specifies the prefix that will be used during the interest forwarding phase in order to compute the longest prefix match. Notice that this primitive can also be omitted. In this case, routing tables need to be manually specified while launching the simulation run.
  Example:

set_predicate 0 0.0000000000 0 A = "provider" ;

column	syntax	example
1	command	set_predicate
2	node_id	0
3	time	0.0000000000
4	unused field	0
5, .., ;	content prefix	A = "provider" ;

Generation process

The workload generator takes as input different files that indicates servers, clients and the list of available files. In particular:

- **clients.dist**: indicates the nodes that will request and the file request rate e.g. Poisson of rate λ.
  Example:

2 poisson 1

column	syntax	example
1	node_id	2
2	request distribution	poisson
3	rate	1

- **prefix.dist**: indicates which prefix(es) a node will serve. Used only if the "automatic" (a la CBN) routing is used.
  Example:

0 A = "Orange"

column	syntax	example
1	node_id	0
2, .., ;	content prefix	A = "provider" ;

- **contents.dist**: indicates the contents that will be available for the download, their permanent location, their popularity and their size.
  Example:

0.4871169634416396 0.5 1 80000 0 0 100 0 A = "Orange" B = "dsaphonwmf" ;

column	syntax	example
1	class request cumulative probability (multiple entries with the same cum. prob. belongs to the same class)	0.4871169634416396
2	file request cumulative probability (cum. probability of requesting a file in the given class)	0.5
3	file size [packets]	1
4	chunk size [bit]	80000
5	node_id	0
6,7,8	unused	
6, .., ;	content name	A = "Orange" B = "dsaphonwmf" ;

During the workload generation process, **time_unit** and the **sim_length** are written at the beginning of the workload file. Then, **publish_content** commands are generated and written in the workload while reading the **contents.dist** input file. At this stage, if not otherwise specified to the generator, **set_predicate** commands are generated through the **prefix.dist** input file. Finally, using the input information given by **contents.dist** and clients.dist, the **download_content** commands are generated according to the request rate/law (i.e. Poisson(λ), $\lambda = 1$) and content popularity specified to the workload generator. In Appendix D a workload example file is reported.

9.2 Simulation run

In this section we introduce the simulator structure and briefly describe the implementation of the CCN data structures. The simulator needs some parameters including the topology file, the workload file, etc. that are specified using an option file whose structure is presented in Appendix D.

Execution Process

Once the simulation is running, CCNPL-Sim relies on a Driver (similar to the one in CBCBSim) to read workload events and on the ssim package to manage other type of events. The ssim package is basically a Discrete-Event Simulation Library that stores scheduled events in a complete binary min heap so that the next event to be executed is always at the top of the tree.

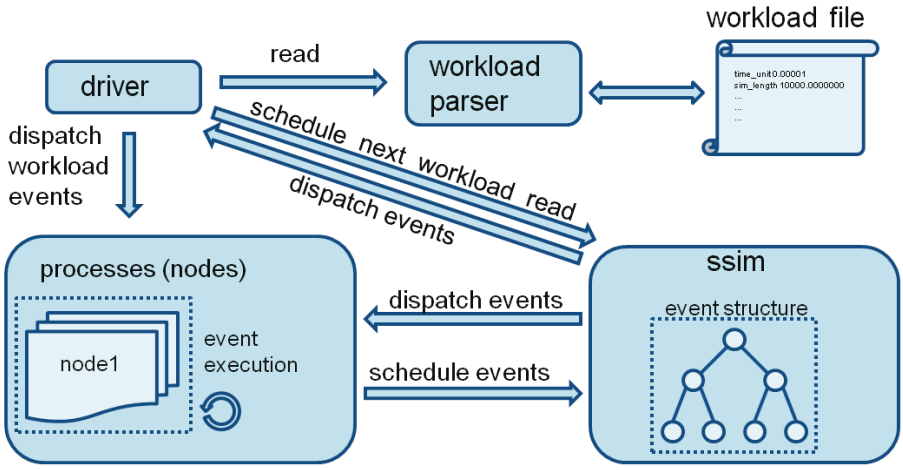


Figure 9.1: CCNPL-Sim execution process.

Fig. 9.1 presents an explicative schema of the CCNPL-Sim execution process. The driver reads and dispatch the events from the workload, loading workload events with the same start

time. When it finishes, it schedules the next workload event(s) read process to ssim. Notice that workload events dispatched to simulation processes (nodes), can generate delayed events (e.g. packet transmissions, timeouts, etc.) that are scheduled to ssim. Ssim dispatches events to the driver or to network nodes and the event "list" is empty (or the specified simulation time is passed), the simulation execution terminates.

Networking functionalities

To construct our network simulator we first develop networking functionalities such as output queues, that are not implemented in CBCBSim. In particular, we implement First In First Out (FIFO) and Fair Queuing (FQ, implemented using the Deficit Round Robin algorithm, DRR described in [42] by Varghese et al.) disciplines.

CCN data structures and packet types

In order the develop a CCN simulator, we need to implement CCN specific data structures (see Fig.1.7) such as Pending Interest Table (PIT), Forwarding Information Base (FIB) and Content Store (CS, Cache). The roles of PIT, CS and FIB structures are explained in Sec.1.3 while here, we briefly recall their usage and introduce their implementations in CCNPL-Sim. Notice that differently from the CCNx implementation [61], we strictly separate the three data structures.

Furthermore we briefly present the packet types used in CCNPL-Sim.

Pending Interest Table - PIT

The PIT keeps track of forwarded interests in order to guide data packets back to its requestor(s). If there exists another PIT entry for the same content name, the interest is not forwarded and the interface from which it has been received is saved. Notice that in order to be able to enable/disable the so called "filtering" feature, we keep a PIT entry for each received interest (differently to the CCNx implementation) in our implementation.

To have fast insert, search and delete operations in the PIT, we implement it with two data structures. This because the three basic PIT operations need to access the PIT in different ways:

- **insert and search**: look-up by content name.
- **delete**: remove expired PIT entries accessing elements by insertion time.

For this reasons, we use two mutually linked data structures. A hash table (unordered multimap - boost libraries) for accessing elements by packet name (content name + packet identifier) and an ordered list (map - std C++ libraries) in order to maintain records ordered by insertion time and delete PIT entries consequently. Fig. 9.2 illustrates the PIT implementation.

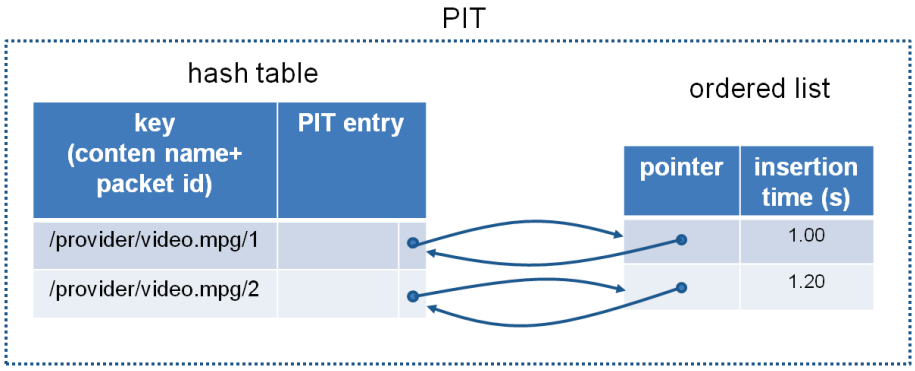


Figure 9.2: PIT data structure in CCNPL-Sim.

Content Store - CS

The Content Store (CS) in CCN is a temporary memory (cache) used by nodes to directly reply to interests if the requested packet is stored locally. Notice that in order to speed up the content repository we distinguish permanent and cached copies. Permanent copies are stored in a list at the content repository that is checked for the first interest only. Cached copies instead are stored in the CS of the network nodes.

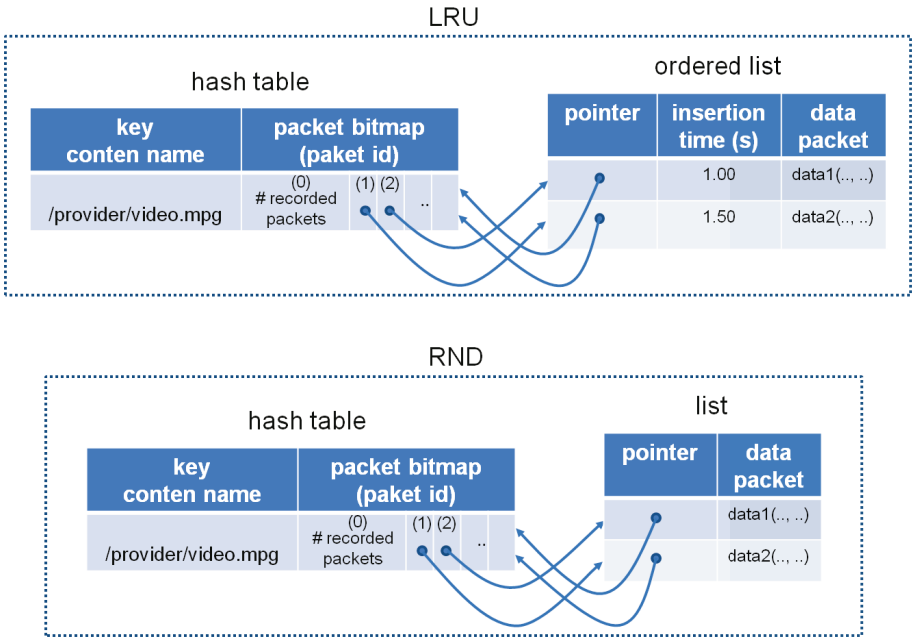


Figure 9.3: LRU and RND cache data structure in CCNPL-Sim.

As for the PIT, we need to have fast access operations in order to speed up the simulation. The LRU and RND replacement policies are slightly different in terms of insert, search and delete requirements:

- **insert and search**: both LRU and RND need to look-up the CS by content name.
- **delete**: LRU policy needs to remove least recently used packets while RND one pick a packet at random.

Therefore, LRU and RND cache structures are slightly different. The LRU is implemented with two mutually linked data structures. Like the PIT, LRU is constructed with a hash table (unordered map - boost libraries) and an ordered list (list - C++ libraries) respectively for search by name and delete least recently used packet(s). Notice that differently from the PIT, for the CS we used a standard list. This because the element that need to be inserted is always the least recently used and hence it is easy to keep the list ordered by inserting new elements on top of it. The RND replacement policy only needs a hash table to search by name operations as delete decision is taken randomly.

Contrarily from the PIT implementation, the CS hash table maintains a record for each content name (and not content name + packet identifier) in order to maintain few records and use less computationally expensive hash functions. Due to this implementation choice, we need that each packet carries the size (in number of packets) of the content it belongs to (this requirement can be relaxed in the future if the simulator need to manage contents whose size is not known a priori). Moreover, for the RND implementation we also need an auxiliary list in order to easily perform random packet(s) removal. Fig.9.3 illustrates the LRU and RND implementation.

Forwarding Information Base - FIB

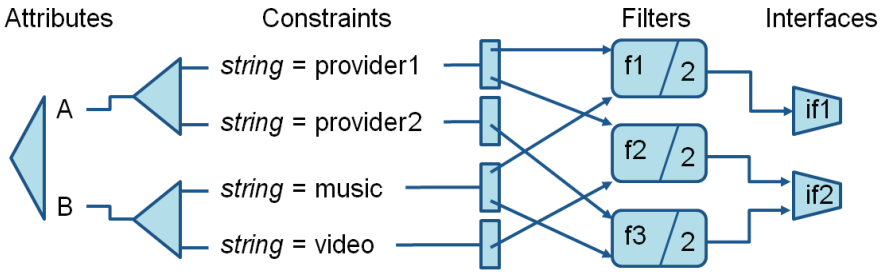


Figure 9.4: Forwarding table structure as used in CCNPL-Sim. Source: [12]

In CCN, the Forwarding Information Base is used to guide Interest toward possible destinations. The FIB structure in CCNPL-Sim is implemented starting from the one employed by

CBCBSim. The forwarding table in CBCBSim is an extension of the basic matching algorithm known as the *counting algorithm* and its implementation is described in [12] by simulator's authors. Due to the different naming scheme used in CBCBSim, we modify the original FIB match function in order to support the longest prefix match operation of CCN nodes. Fig.9.4 illustrates the structure of the forwarding table.

Packet types

In CCNPL-Sim we define packets as events that move from a node (alos referred to as process) to another. Obviously like in CCN there are two different packet types: Interests and Data packets.

- **interest**: contains the *message* packet used in the original CBCBSim that is matched against the forwarding table during the interest forwarding process. Furthermore, we add URI-like content name, packet identifier, content size (in packets) and other auxiliary information. If the packet identifier is 0, the interest is sent to acquire informations on the content that is requested i.e. number of packets, packet size, etc. Interest packet size is set to 25 Bytes as default.

- **data**: contains the informations also carried by the interest packet except from the *message* packet. If the packet identifier is 0, the data packet (also INFO packet) is sent to reply to an interest packet with packet id 0. Data packet size can vary and is defined by the **publish_content** command in the workload file (see Sec. 9.1).

9.3 Simulation outputs

The simulator produces two different output files. The first one is directly printed in **stdout** and records runtime events e.g. the termination of a content download process. Notice that in principle each event handled by the simulator (packet received, packet sent, packet lost, etc.) can be printed to **stdout**. However print operations significantly increment the simulation execution time and the **stdout** statistics collected by the released version of the simulator are limited to events that represent the end of a content download process. An example of **stdout** output is given in Tab.9.1.

The second output file is a log saved to the path/file name specified to the simulator through the **outputfile** input parameter. The statistics collected in this file are of three different types. The first one represents network performance (as the average queue occupancy) and is preceded by the **QUEUE stats:** keyword (see Tab.9.2). The second set of statistics collected in the **outputfile** represents per pupularity class cache performance (miss, hit, inpu/output request rate) and is preceeded by the **Stats:** keyword (see Tab.9.3). The third and last set of statistic represents per node

and per forwarding prefix split ratio/rate in case of multipath scenario. An example of the split statistic is represented in Tab.9.4. Notice that the three groups of statistics are printed out each t_i seconds as specified by the parameter **dci** (data collection interval) and at the end of the simulation (if $t_i = 0$ only final statistics are collected).

column	syntax	example
1	simulation time	3169053.0000000000
2,3	event	closed socket
4	node id	2
5	content_name+port_id	/Orange/wgpfowikcv/port:0
6	delivery time [sec]	1.023
7	file size [bits]	80000
8	average RTT [sec]	0.0010
9	average packaet delay (take into account retransmissions) [sec]	0.0015
10	keyword	class
11	popularity class	2
12	sockets still opened in this node	10
13	avg receiver interest window size	5.4
14	number of exploited routes	5
15	average number of exploited routes	2.4

Table 9.1: **stdout** output syntax.

column	syntax	example
1	keyword	QUEUE [FINAL]
2	time (only if not final queue statistic)	1.0
3	keyword	NODE
4	node id	1
5	keyword	to
6	node id	2
7	average queue size [packets]	10.43

Table 9.2: queue statistics in the **outputfile**

column	syntax	example
1	keyword	CACHE [FINAL] NODE
2	time (only if not final queue statistic) [sec]	2.0
3	node id	1
4	request miss rate	2
5	request hit rate	0.988
6	request rate	2.988
7	miss probability (with filtered requests as miss)	0.6711
8	hit probability	0.3289
9	request filter probability	0
10	miss probability (without filtered requests)	0.6711
11	keyword	class
12	class_id (popularity class in our settings)	1

Table 9.3: cache statistics in the **outputfile**

column	syntax	example
1	keyword	SPLIT [FINAL] NODE
2	time (only if not final queue statistic) [sec]	2.0
3	keyword	NODE
4	node id	1
5	keyword	to
6	node id	2
7	forwarded requests [%]	23.3
8	forwarded requests [rate]	1.4

Table 9.4: split statistics in the **outputfile**

Chapter 10

Conclusion

With the introduction of the World Wide Web the Internet communication model changed from host-to-host to content delivery and service access. The current network architecture successfully addressed new applications' requirements deploying content delivery services as an overlay on top of IP (e.g. HTTP is an application level protocol that implements pull-based content delivery mechanisms [70]). Despite to its success, the current Internet architecture has been designed with the objective of enabling machine-to-machine communication and not content retrieval/dissemination.

The evident mismatch between the current Internet and its usage generates many problems in terms of security, availability and data location dependence motivating the research community to investigate in future Internet's architectures. In this context, ICN propositions try to solve architectural problems redesigning the Internet around named content rather than hosts' addresses.

Contribution

In this thesis we study ICN architectures with particular focus on the CCN proposition. This new architecture concept brings potential benefits in terms of security, content dissemination, simplified network management, mobility, etc. CCN has been proposed in 2009 in [14] by Van Jacobson et al. at the Palo Alto Research Center (PARC) and is object of many research projects (e.g. NDN [15] USA - NSF, CONNECT[16] French - ANR). The architecture proposal is still in its infancy and many research questions remain to be addressed. With this thesis we address some open issues that we see in the original proposition. In particular, we investigate user perceived performance and propose congestion control and forwarding mechanisms for the CCN architecture.

In the first part of the thesis we analyze CCN performance. In traditional networks, storage and bandwidth sharing are studied separately as storage is usually deployed at application

level (e.g. CDNs). On the contrary, in CCN caches are directly embedded at network level influencing user performance. In order to capture caching behavior we derive closed formulas for Least Recently Used (LRU) and Random (RND) cache performance analysis neglecting bandwidth constraints. The two replacement policies are analyzed with different assumptions. The LRU cache performance characterization, differently from the one for RND, also consider packet level-caches in order to better understand the CCN architecture. Both models accounts for network of caches differently from other models proposed in the literature that study single cache behavior. The comparison between LRU and RND also shows that the two replacement policies has similar performance and can be used alternatively at different network levels. As a second step, we take into account capacity constraints and propose mathematical formulas for the analysis of the user perceived performance capturing the interplay between storage and bandwidth in the CCN architecture. The proposed modeling framework highlights the performance dependence from key system parameters such as content popularity, content and cache size. Even if tailored for the CCN architecture, our analysis can be easily extended to other ICN proposals with similar properties as receiver driven transport and in-network caching. To test the validity of the proposed formulas, we develop a dedicated CCN Packet Level Simulator (CCNPL-Sim). CCNPL-Sim is built upon some of the packages used by the CBCBsim [69] to which we add output queues, receiver driven transport and CCN specific packet and data structures. The CCNPL-Sim source code is published under the GNU GPL v2 license and is available for download at `http://code.google.com/p/ccnpl-sim/`.

In the second part of the thesis we focus our attention on CCN protocol design. To this aim exploiting the outcomes of the performance analysis, we propose congestion control and forwarding mechanism that we further implement and test in CCNPL-Sim. Concerning the congestion control, we propose a receiver-driven window-based protocol named Interest Control Protocol (ICP). ICP uses the Additive Increase Multiplicative Decrease mechanism to drive the interest window size evolution using timeouts as a congestion indication. We then prove the proposed protocol stability and its convergence to the efficient and fair regime obtained in the performance analysis part. Exploiting CCN path symmetry we also propose an hop-by-hop congestion control mechanism with the objective of accelerating congestion detection/reaction. The proposed mechanism consists in measuring the interest rate in order to queue interests belonging to aggressive flows shaping them at the measured fair rate. With respect to other proposals, our interest shaper controls interest instead of data avoiding packet losses and is implemented directly at output queues and hence immediately before the bottlenecked links. We then extend the ICP protocol in order to efficiently exploit multiple heterogeneous paths characterized by significantly different round trip delay. To this aim, we adopt a Remote Adaptive Active Queue Management (RAAQM) mechanism controlling a single AIMD congestion window with a probabilistic per-path (each path is

identified by a label) window decrease. Coupled with the modified congestion control mechanism, we also propose a distributed interest forwarding mechanism that bases its decision on local per-interface and per-prefix performance measurement. We then compare the proposed forwarding algorithm with the optimal solution demonstrating the quality of our mechanism with respect to other forwarding choices. From the simulations we carried on we think that the proposed remote active queue management control algorithm coupled with the forwarding mechanism is a good solution for a stable and efficient multipath congestion control.

All the results presented in this thesis, has been obtained using a certain number of assumptions. For example, we assumed the requests' Independence between nodes/levels in the cache performance evaluation, negligible propagation delay in the bandwidth modeling part and introduced node identifiers in the RAAQM protocol design. Even if we verified and motivated those assumptions through simulations and/or mentioning previous works, in a more complex scenario (as the real Internet) they could be no longer valid. The natural next step to consolidate the results presented in this dissertation is to perform a massive prototyping and experimentation phase to derive more precise and significant conclusions on the assumptions' validity and scalability of the proposed solutions.

Discussion

ICN is an interesting evolution opportunity for the current Internet and for this reason, it is one of the major research item in the networking community. Many research projects involving important universities and research centers are interested in developing such concept. At the same time, important actors of the Internet ecosystem as Internet Service Providers (Orange, British Telecom, Telefonica, etc.), Hardware and Chipset Manufacturers (Alcatel- Lucent, Cisco, Huawei, Samsung, Ericssonn, Nokia, etc.) are active in the field in order to explore potential benefits and identify business opportunities in ICN.

The original ICN objective is to replace the current Internet architecture. However, we believe that is essential for the ICN research community to be more concrete and find short-term deployment scenario. As for the Internet architecture evolution, we think that economic incentives will be fundamental for the future of ICN. In particular, it is important to demonstrate that ICN can enable new interesting services not deployable in the current architecture. An important step toward this direction has been made by PARC that recently initiate the Emerging Network Consortium (ENC) in order to attract potential industrial partners starting the discussion on potential CCN use cases.

Appendices

Appendix A

LRU cache modelling - proofs

Proof of Lemma 2.3.3

First notice that for a given packet in class k, $i(c_k)$,

$$\begin{aligned}\mathbf{E}[B^{i(c_k)}(u,t)] &= \mathbb{P}(B^{i(c_k)}(u,t)=1)\\ &= \mathbb{P}(r_{i(c_k)}(u,t)>0) = 1-e^{-\frac{\lambda q_k}{m}(t-u)}. \end{aligned} \tag{A.1}$$

This is due to the fact the contents with the same popularity are requested uniformly with probability $1/m$, that $\{r_{i(c_k)}(0,t), t\geq 0\}$ is a Poisson process of rate $\lambda q_k/m$ by MMRP properties and packets are indistinguishable.
In order to compute $\lim_{t\to\infty}\mathbf{E}[S(0,t)]$, let us first evaluate the following lower bound as $K\to\infty$,

$$\begin{aligned}\mathbf{E}[S(0,t)] &= \sum_{k=1}^{\infty}\sum_{c_k=1}^{m}\mathbf{E}\left[\sum_{i(c_k)=1}^{N_{ch}} B^{i(c_k)}(0,t)\right]\\ &\geq m\sigma\sum_{k=1}^{\infty}\int_k^{k+1}\left(1-e^{-\frac{\lambda q_u}{m}t}\right)du\\ &= m\sigma\int_1^{\infty}\left(1-e^{-\frac{\lambda}{m}\frac{c}{u^\alpha}t}\right)du\\ &= \left(\frac{\lambda c}{m}t\right)^{1/\alpha}\frac{m\sigma}{\alpha}\int_0^{\frac{\lambda c}{m}t} v^{-1/\alpha-1}(1-e^{-v})dv\\ &= \Gamma\left(1-\frac{1}{\alpha}\right)m\sigma\left(\frac{\lambda c t}{m}\right)^{1/\alpha}+o(1/t)\ \ t\to+\infty. \end{aligned} \tag{A.2}$$

Similarly one can derive the following upper bound:

$$\begin{aligned}
\mathbf{E}[S(0,t)] - m\sigma\left(1-e^{-\frac{\lambda}{m}ct}\right) &= m\sigma\sum_{k=2}^{\infty}\int_{k}^{k+1}\left(1-e^{-\frac{\lambda q_k}{m}t}\right)du \\
&\leq m\sigma\sum_{k=1}^{\infty}\int_{k}^{k+1}\left(1-e^{-\frac{\lambda q_u}{m}t}\right)du \\
&= m\sigma\int_{1}^{\infty}\left(1-e^{-\frac{\lambda}{m}\frac{c}{u^{\alpha}}t}\right)du \\
&= \left(\frac{\lambda c}{m}t\right)^{1/\alpha}\frac{m\sigma}{\alpha}\int_{0}^{\frac{\lambda c}{m}t}v^{-1/\alpha-1}(1-e^{-v})dv \\
&= \Gamma\left(1-\frac{1}{\alpha}\right)m\sigma\left(\frac{\lambda ct}{m}\right)^{1/\alpha} + o(1/t) \;\; t\to+\infty. \qquad \text{(A.3)}
\end{aligned}$$

As a consequence, if we consider $\lim_{t\to\infty}\frac{\mathbf{E}[S(0,t)]^{\alpha}}{t}$, both upper and lower bounds coincide with

$$1/g \equiv \lim_{t\to\infty}\frac{\mathbf{E}[S(0,t)]^{\alpha}}{t} = \lambda c\sigma^{\alpha}m^{\alpha-1}\Gamma\left(1-\frac{1}{\alpha}\right)^{\alpha}.$$

□

Proof of Lemma 2.3.4

Due to the property of LRU replacement policy, packet i_k is moved to the front of the list at each $t_n^{(i(c_k))}$. A miss event is defined by $\{S^{(i(c_k)}(t_{n-1}^{(i(c_k))}, t_n^{(i(c_k))}) \geq x\}$, i.e. the sum of different packets requested after the previous miss event for packet $i(c_k)$ is larger than the cache size, x. Therefore

$$\begin{aligned}
&\mathbb{P}\left(\begin{array}{c}\text{a miss on packet } i(c_k)\\ \text{occurs at} \quad t_1^{(i_k)},...,t_n^{(i_k)}\end{array}\right) = \\
&\mathbb{P}[S^{(i(c_k))}(t_j^{(i(c_k))}, t_{j+1}^{(i(c_k))}) \geq x, j=1,\ldots,n-1] \qquad \text{(A.4)}
\end{aligned}$$

The independence among $S^{(i(c_k))}(t_l^{(i(c_k))}, t_{l+1}^{(i(c_k))})$ and $S^{(i(c_k))}(t_{l+j}^{(i(c_k))}, t_{l+j+1}^{(i(c_k))})$, $j \geq 1$ is a consequence of the memory-less Poisson property at content level and of the geometric content size distribution in terms of requested packets for a given content request. By recalling its definition

(2.4), we have

$$\begin{aligned}
&\mathbb{P}[S^{(i(c_k))}(t_1^{(i(c_k))}, t_2^{(i(c_k))}) > x] = \\
&\sum_{v_1+...v_K=1}^{M} \sum_{k'=1}^{K} \mathbb{P}[B^{j_{k'}}(t_1,t_2)=1]^{v_{k'}} \mathbb{P}[vN_{ch} \geq x] = \\
&\sum_{v_1+...+v_K=1}^{M} \sum_{k'=1}^{K} \left(1 - e^{-\frac{\lambda}{m} q'_k (t_2^{(i(c_k))} - t_1^{(i(c_k))})}\right)^{v_{k'}} \mathbb{P}[vN_{ch} \geq x] \\
&= \sum_{v_1+...+v_K=1}^{M} \sum_{k'=1}^{K} \left(1 - e^{-\frac{\lambda}{m} q'_k (\tau_1^{(i(c_k))})}\right)^{v_{k'}} \mathbb{P}[N_{ch} \geq x/v],
\end{aligned}$$

where we sum over the probability to have at least one packet for v_1 contents in class $k = 1$, v_2 contents in class $k = 2$ and so on, so that $v_1 + v_2 + ... + v_K = v$, times the conditioned probability to have more than x different packets when having v different content requests. The memory-less property of the Poisson process at content level and of the geometric distribution at packet level allow to conclude that $\mathbb{P}[S^{(i(c_k))}(t_1^{(i(c_k))}, t_2^{(i(c_k))}) > x]$ only depends on the length of the interval $\tau_1^{(i(c_k))}$, which is equivalent to say that in steady state,

$$\mathbb{P}\left(\begin{array}{c} \text{a miss on packet } i(c_k) \\ \text{occurs at} \quad t_1^{(i_k)}, ..., t_n^{(i_k)} \end{array}\right) = \mathbb{P}[S^{(i_k)}(0, \tau^{(i_k)}) > x]^n \tag{A.5}$$

□

Proof of Proposition 2.3.5

We first evaluate the upper bound.

$$\begin{aligned}
&\mathbb{P}[S^{(i(c_k))}(t_n^{(i)}, t_{n+1}^{(i(c_k))}) \geq x] = \mathbb{P}[S^{(i(c_k))}(0, \tau^{(k)}) \geq x] \\
&= \mathbb{P}[S^{(i(c_k))}(0, \tau^{(k)}) \geq x, \tau^{(k)} \geq g x^{\alpha}] + \mathbb{P}[S^{(i(c_k))}(0, \tau^{(k)}) \geq x, \tau^{(k)} < g x^{\alpha}] \\
&\leq \mathbb{P}[S^{(i(c_k))}(0, \tau^{(k)}) \geq x, \tau^{(k)} \geq g x^{\alpha}] + \mathbb{P}[S^{(i(c_k))}(0, g x^{\alpha}) > x] \\
&\leq \mathbb{P}[\tau^{(k)} \geq g x^{\alpha}] + \mathbb{P}[S(0, g x^{\alpha}) > x] \\
&\sim \mathbb{P}[\tau^{(k)} \geq g x^{\alpha}], \;\; x \to \infty
\end{aligned} \tag{A.6}$$

where we have used monotonicity of $S(0,t)$ in t as well as $\mathbb{P}[S(0,gx^\alpha) > x] \to 0, x \to \infty$. The lower bound can also be obtained,

$$\begin{aligned}\mathbb{P}[S^{(i(c_k))}(0,\tau^{(k)}) \geq x] &\geq \mathbb{P}[S^{(i(c_k))}(0,\tau^{(k)}) \geq x, \tau^{(k)} > gx^\alpha] \\ &\geq \mathbb{P}[S(0,gx^\alpha) > x, \tau^{(k)} > gx^\alpha] \\ &= \mathbb{P}[S(0,gx^\alpha) \geq x]\mathbb{P}[\tau^{(k)} > gx^\alpha] \\ &\sim \mathbb{P}[\tau^{(k)} \geq gx^\alpha], \;\; x \to \infty \end{aligned} \tag{A.7}$$

Where we have used the law of large numbers for $S(0,gx^\alpha)/x \to \mathbb{E}[S(0,gx^\alpha)]/x = 1$ in probability. □

Proof of Theorem.2.3.6

The proof is divided into steps:

1. we show that $\forall k = 1, ..., K, \forall c_k$ (i.e. content in class k),

$$\begin{aligned}&\mathbb{P}[\mathcal{M}(t_1^{i(c_k)}), \mathcal{M}(t_2^{i(c_k)}), ..., \mathcal{M}(t_l^{i(c_k)}), \mathcal{M}(t_1^{(i+1)(c_k)}), \mathcal{M}(t_2^{(i+1)(c_k)}), ..., \mathcal{M}(t_l^{(i+1)(c_k)})] \\ &\mathbb{P}[\mathcal{M}(t_1^{i(c_k)}), \mathcal{M}(t_2^{i(c_k)}), ..., \mathcal{M}(t_l^{i(c_k)})], \qquad \forall l \geq 1;\end{aligned}$$

2. we apply Prop.2.3.5 to explicitly compute the miss probabilities.

1. Prop.2.3.5 allows to state that $\mathbb{P}[\mathcal{M}(t_l^{i(c_k)})] = \mathbb{P}[S^{i(c_k)}(t_{l-1}^{i(c_k)}, t_l^{i(c_k)}) \geq x] = \mathbb{P}[\tau^{i(c_k)} > gx^\alpha]$, $\forall l \geq 1$. Now, if the interval between two consecutive misses for packet i is $\tau^{i(c_k)}$, the interval between two consecutive misses for the following packet, $i+1$ of that specific content can not be shorter than $\tau^{i(c_k)}$. In fact, either the packet $i+1$ is requested after packet i both one $VRTT_k$ after $t_{l-1}^{i(c_k)}$ and one $VRTT_k$ after $t_l^{i(c_k)}$ or in one of the two cases the content request stops at packet i. Given that $\tau^{(i+1)(c_k)} \geq \tau^{i(c_k)} > gx^\alpha$, the miss event for packet i implies a miss event for packet $i+1$ (when requested) with probability equal to one, since $\tau^{(i+1)(c_k)} \geq \tau^{i(c_k)}$, which concludes the first step of the proof.
2. The second step follows by Prop.2.3.5, when applied on the first packet of each content of each popularity. The independence between misses of different content requests as $x \to \infty$ can be proved by using lower and upper bounds as in the proof of Prop.2.3.5 and it happens to be a consequence of the memoryless Poisson property of content-level requests. □

Proof of Proposition 2.3.9

First, observe that if the l-th packet of a given content is filtered, then packet $l+1$ is filtered with $1 \leq l \leq \sigma - 1$. This means that the l-th packet is filtered when the following event is verified: $F_l = \bigcup_{m=1}^{l-1} E_m$,
$E_m = \{\text{packet } m \text{ is filtered and packet } m-1 \text{ is not filtered}\}$.
$\mathbb{P}\{E_m\} = p(1-p)^{m-1}$ where p is the probability that the first packet is filtered, easily computed by observing that content request inter-arrivals $\sim \text{Exp}(\lambda_k)$, hence $p = 1 - e^{-\lambda_k \Delta_k}$ is the probability that the inter-arrival among two request of the first packet of the same content is smaller than Δ_k with $\Delta_k = \min(\Delta, VRTT_k)$. Therefore $F_l = \sum_{m=1}^{n} p(1-p)^{m-1} = 1 - (1-p)^n$, being n the number of packets. Defining $b_k = 1 - p = e^{-\lambda_k \Delta_k}$ and averaging over the number of packets belonging to a content N_{ch}, which is geometrically distributed with mean σ, we obtain:

$$p_{filt,k}(1) = 1 - \mathbb{E}[b_k^{N_{ch}}] = 1 - \frac{b_k/\sigma}{1-(1-1/\sigma)b_k}$$
$$= \frac{1-b_k}{1-(1-1/\sigma)b_k}$$

Just using the fact that the p.g.f of the geometric distribution $\mathbb{E}[z^{N_{ch}}] = \frac{z/\sigma}{1-(1-1/\sigma)z}$. □

Proof of Lemma 2.3.10

Let us consider $i = 2$ and proceed as for the proof of Lemma 2.3.3 by constructing a lower and an upper bound for $\mathbb{E}[S_2(0,t)]$.
The input rate at the second content store, here denoted as $\lambda(2)$, is a function of the miss rate of the first cache and of the topology under study.

Under the assumption of a MMRP miss process with intensity equal to $\mu(1) = \lambda \sum_j p_j(1) q_j$, the miss rate for popularity class k is $\mu_k(1) = \mu \frac{p_k(1) q_k}{\sum_j p_j(1) q_j}$. The popularity distribution of content requests in the miss process results, therefore, to be modified since probability to have a request for a content of class k becomes: $q_k(2) \equiv \frac{\mu_k(1)}{\mu(1)} = \frac{p_k(1) q_k}{\sum_j p_j(1) q_j}, \forall k = 1, ..., N$. Thus, the input content request rate at the second hop is equal to $\lambda(2) = \mu(2) = \lambda \sum_j p_j(1) q_j$ for the topology in Fig.2.1(a) and to $\lambda(2) = 2\mu(1) = 2\lambda \sum_j p_j(1) q_j$ for the topology in Fig.2.1(b). For both topologies, one can write

$$\begin{aligned}\mathbb{E}[S_2(0,t)] &= \sum_{k=1}^{\infty}\sum_{c_k=1}^{m}\mathbb{E}\left[\sum_{j(c_k)=1}^{N_{ch}} B^{j(c_k)}(0,t)\right] \\ &= \sum_{k=1}^{\infty} m\sigma\mathbb{E}[B_k^{(j)}(0,t)] \\ &= m\sigma\sum_{k=1}^{\infty}\frac{\lambda(2)q_k(2)}{\lambda q_k}\left(1-e^{-\frac{\lambda q_k}{m}t}\right) \end{aligned} \tag{A.8}$$

Now, if $\lambda(2) = \mu(1) = \lambda\sum_j p_j(1)q_j$,

$$\begin{aligned}\mathbb{E}[S_2(0,t)] &= m\sigma\sum_{k=1}^{\infty} p_k(1)\left(1-e^{-\frac{\lambda q_k}{m}t}\right) \\ &= m\sigma\sum_{k=1}^{\infty} e^{-\frac{\lambda q_k}{m}gx^{\alpha}}\left(1-e^{-\frac{\lambda q_k}{m}t}\right) \\ &\geq \left(\frac{\lambda c}{m}t\right)^{1/\alpha}\frac{m\sigma}{\alpha}\int_0^{\frac{\lambda c}{m}t} v^{-1/\alpha-1}(1-e^{-v})e^{-v\frac{gx^{\alpha}}{t}}dv \\ &\sim \Gamma\left(1-\frac{1}{\alpha}\right)\left(\frac{\lambda ct}{m}\right)^{1/\alpha} m\sigma,\ \ t\to+\infty \end{aligned} \tag{A.9}$$

Using the dominated convergence theorem, one can easily obtain the upper bound as for the proof in Lemma 2.3.3

$$\sup\mathbb{E}[S_2(0,t)] \sim m\sigma + \Gamma\left(1-\frac{1}{\alpha}\right)m\sigma\left(\frac{\lambda ct}{m}\right)^{1/\alpha} \tag{A.10}$$

Similarly when $\lambda(2) = 2\mu(1) = 2\lambda\sum_j p_j(1)q_j$ (tree topology), one gets

$$\sup\mathbb{E}[S_2(0,t)] \sim m\sigma + \Gamma\left(1-\frac{1}{\alpha}\right)m\sigma\left(\frac{2\lambda ct}{m}\right)^{1/\alpha}$$

Therefore, regardless of the topology, it results

$$1/g(2) \equiv \lim_{t\to\infty} \frac{\mathbb{E}[S_2(0,t)]^\alpha}{t} = \frac{\lambda(2)}{\mu(1)} \lambda c \sigma^\alpha m^{\alpha-1} \Gamma\left(1-\frac{1}{\alpha}\right)^\alpha = \frac{\lambda(2)}{\mu(1)} 1/g \tag{A.11}$$

By iteration, it suffices to consider the input rate at node $i > 2$, $\lambda(i)$ and the modified popularity profile at node i, $q_k(i) = \prod_{j=1}^{i-1} p_k(j) q_k / \sum_{l=1}^{K} \prod_{j=1}^{l-1} p_l(j) q_l, \quad k = 1, ..., K$, to derive the general statement. □

Proof of Proposition 2.3.11

Given a class k of content popularity, thanks to Prop.2.3.2

$$p_k(1) = e^{-\frac{\lambda}{m} q_k g x^\alpha}, \quad k = 1, ..., N$$

Under the MMRP assumption on the miss process and hence on the input process of the second (level) cache, one can compute the miss probability for class k at the second (level) by accounting for the modified rate and modified popularity in the miss rate. It results:

$$\begin{aligned} p_k(2) &= e^{-\frac{\lambda(2)}{m} q_k g(2) x(2)^\alpha \frac{p_k(1)}{\sum_{j=1}^{K} q_j p_j(1)}} \\ &= e^{-\frac{\lambda}{m} \sum_{j=1}^{K} q_j p_j(1) q_k g(2) x(2)^\alpha \frac{p_k(1)}{\sum_{j=1}^{K} q_j p_j(1)}} \\ &= p_k(1)^{(\frac{x(2)}{x(1)})^\alpha \frac{g(2)}{g} p_k(1)} = p_k(1)^{(\frac{x(2)}{x(1)})^\alpha p_k(1)}. \end{aligned} \tag{A.12}$$

where we used $g(i) = g$, $i \geq 1$, as it is the case for the considered topology (Fig.2.1(a)). By iteration, one gets the expression of the miss probability at node i, $i > 1$, which depends on previous nodes' miss probabilities in the following manner:

$$p_k(i) = p_k(1)^{\frac{g(i)}{g(1)} \prod_{l=1}^{i-1} \left(\frac{x(2)}{x(1)}\right)^\alpha p_k(l)} = p_k(1)^{\prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)}\right)^\alpha p_k(l)} \tag{A.13}$$

or alternatively, $\log p_k(i) = \log p_k(1) \prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)}\right)^\alpha p_k(l)$. □

Proof of Corollary 2.3.12

Let us consider first the case $i = 2$. Note that $g(i)/g = \frac{\mu(i)}{\lambda(i)} = 1/2$ in the binary tree topology (Fig.2.1(b)). So we have

$$\begin{aligned} p_k(2) &= e^{-\frac{\lambda(2)}{m} q_k g(2) x(2)^\alpha \frac{p_k(1)}{\sum_{j=1}^K q_j p_j(1)}} \\ &= e^{-\frac{2\lambda}{m} \sum_j p_j(1) q_j q_k g(2) x(1)^\alpha (\frac{x(2)}{x(1)})^\alpha \frac{p_k(1)}{\sum_{j=1}^K q_j p_j(1)}} \\ &= p_k(1)^{2\frac{g(2)}{g}(\frac{x(2)}{x(1)})^\alpha p_k(1)} = p_k(1)^{(\frac{x(2)}{x(1)})^\alpha p_k(1)}. \end{aligned} \tag{A.14}$$

The iteration procedure for any $i > 1$, follows the same calculations made for the linear topology. □

Proof of Corollary 2.3.13

Given the aggregation at the first hop only, we have $p_k^f(1) = p_k(1)$ since the filtering acts on the miss rate, whereas the miss rate filtered $\mu_k^f(1) = \mu_k(1)(1-p_{filt,k}(1))$. The miss probability at the second node results to be modified according the new popularity defined in $\mu_k^f(1)$. As in (A.14), one can derive

$$\begin{aligned} p_k^f(2) &= e^{-\frac{\mu^f}{m} q_k g(2) x(2)^\alpha \frac{p_k(1)(1-p_{filt,k}(1))}{\sum_{j=1}^K q_j p_j(1)(1-p_{filt,j}(1))}} \\ &= e^{-\frac{\lambda}{m} q_k g(2) x(2)^\alpha p_k(1)(1-p_{filt,k}(1))} \\ &= p_k(1)^{(\frac{x(2)}{x(1)})^\alpha \frac{g(2)}{g} p_k(1)(1-p_{filt,k}(1))} = p_k(2)^{(1-p_{filt,k}(1))}. \end{aligned} \tag{A.15}$$

By iterating eq.(A.15) one obtains the statement (2.15). □

Appendix B

RND cache modelling - proofs

Proof of Theorem 2.4.6

(i) Using (2.22), equation (2.31) reduces to $g(z) = x$ where

$$g(z) = z\frac{F'(z)}{F(z)} = \sum_{j\geq 1} \frac{q_j z}{1+q_j z}. \tag{B.1}$$

Continuous function $g : z \in [0, +\infty[\to g(z) \in [0, +\infty[$ vanishes at $z = 0$, is strictly increasing on $[0, +\infty[$ and tends to $+\infty$ when $z \uparrow +\infty$. Equation (2.31) has consequently a unique positive solution θ_x. Note that θ_x tends to $+\infty$ with x since $g(z) \leq \Sigma_{j\geq 1}(q_j z) = z$, hence $\theta_x \geq x$.

(ii) Consider the random variable R_x with distribution

$$\mathbb{P}(R_x = r) = \frac{G(x)}{F(\theta_x)}\theta_x^r, \quad r \geq 0, \tag{B.2}$$

where θ_x satisfies (2.31); note that definition (B.2) for R_x is equivalent to

$$G(x) = \frac{F(\theta_x)}{\theta_x^r}\mathbb{P}(R_x = r), \quad r \geq 0. \tag{B.3}$$

Using definition (B.2), the generating function of random variable R_x is $z \mapsto F(z\theta_x)/F(\theta_x)$; in view of (2.31), the expectation of variable R_x is then

$$\mathbb{E}(R_x) = \frac{\mathrm{d}}{\mathrm{d}z}\frac{F(z\theta_x)}{F(\theta_x)}\Big|_{z=1} = \theta_x\frac{F'(\theta_x)}{F(\theta_x)} = x$$

so that random variables $Y_x = (R_x - x)/\sqrt{x}$, $x \geq 0$, are all centered. Besides, the Laplace transform of variable Y_x is given by

$$\mathbb{E}(e^{-vY_x}) = e^{v\sqrt{x}}\mathbb{E}(e^{-vR_x/\sqrt{x}}) = e^{v\sqrt{x}}\frac{F(\theta_x e^{-v/\sqrt{x}})}{F(\theta_x)}$$

for all $v \in \mathbb{C}$. By Lévy's continuity theorem ([71], Theorem 4.2.4), assumption (2.32) entails that variables Y_x converge in distribution when $x \uparrow +\infty$ towards a centered Gaussian variable with variance σ^2; moreover, assumption (2.33) ensures that the conditions of Chaganty-Sethuraman's theorem ([72], Theorem 4.1) hold so that $\mathbb{P}(R_x = x) = \mathbb{P}(Y_x = 0)$ is asymptotic to

$$\mathbb{P}(R_x = x) \sim 1/\sigma\sqrt{2\pi x} \tag{B.4}$$

as $x \uparrow +\infty$. Equation (B.3) for $r = x$ and asymptotic (B.4) together provide estimate (2.34) for $G(x)$ □

Proof of Lemma 2.4.7

Let $q_y = A/y^\alpha$ and $f_z(y) = \log(1 + q_y z)$ for any real $y \geq 1$ and $z \in \mathbb{C}$; definitions (2.22), (2.30) and the above notation then entail that

$$\log F(z) = \sum_{k \geq 1} f_z(k);$$

function $\log F$ is analytic in the domain $\mathbb{C} \setminus \mathbb{R}^-$. For given $z \in \mathbb{C} \setminus \mathbb{R}^-$ and integer $J \geq 1$, the Euler-Maclaurin summation formula [38] reads

$$\sum_{k=1}^{J} f_z(k) = \int_1^J f_z(y)\mathrm{d}y + \frac{1}{2}\left[f_z(J) + f_z(1)\right] + \frac{1}{12}\left[f_z'(J) - f_z'(1)\right] + \frac{T_z(J)}{6} \tag{B.5}$$

with

$$T_z(J) = \int_1^J B_3(\{y\}) f_z^{(3)}(y)\mathrm{d}y,$$

where $B_3(y) = y(y-1)(2y-1)/2$ is the third Bernoulli polynomial and $\{y\}$ denotes the fractional part of real y; derivatives of f_z are taken with respect to y. Consider the behavior of the r.h.s. of (B.5) as J tends to infinity. We first have $f_z(1) = \log(1 + Az)$ and $f_z(J) = O(J^{-\alpha})$ for large J; differentiation entails

$$f_z'(1) = -\frac{\alpha A z}{1 + Az}$$

and $f_z'(J) = O(J^{-\alpha-1})$ for large J. Differentiating twice again with respect to y shows that the third derivative of f_z is $O(y^{-\alpha-3})$ for large positive y, and is consequently integrable at infinity. Letting J tend to infinity in (B.5) and using the above observations together with the boundedness of periodic function $y \geq 1 \mapsto B_3(\{y\})$, we obtain

$$\log F(z) = \int_1^{+\infty} f_z(y)\mathrm{d}y + \frac{1}{2}\log(1+Az) + \frac{\alpha A z}{12(1+Az)} + \frac{T_z}{6} \tag{B.6}$$

where

$$T_z = \int_1^{+\infty} B_3(\{y\}) f_z^{(3)}(y)\mathrm{d}y. \tag{B.7}$$

Now, considering the first integral in the r.h.s. of (B.6), the variable change $y = t(Az)^{1/\alpha}$ gives

$$\begin{aligned}\int_1^{+\infty} f_z(y)\mathrm{d}y &= (Az)^{1/\alpha}\Big[L - \int_0^{1/(Az)^{1/\alpha}} \log\left(1+\frac{1}{t^\alpha}\right)\mathrm{d}t\Big] \\ &= L(Az)^{1/\alpha} - \log(Az) - \alpha + o(1)\end{aligned} \tag{B.8}$$

where L is the finite integral [38]

$$L = \int_0^{+\infty} \log\left(1+\frac{1}{t^\alpha}\right)\mathrm{d}t = \frac{\pi}{\sin(\pi/\alpha)} = \alpha\nu_\alpha^{1/\alpha} \tag{B.9}$$

with ν_α introduced in (2.36) for $\alpha > 1$, and where

$$\int_0^{1/(Az)^{1/\alpha}} \log\left(1+\frac{1}{t^\alpha}\right)\mathrm{d}t = \frac{\log(Az)}{(Az)^{1/\alpha}} + \frac{\alpha}{(Az)^{1/\alpha}} + o(1). \tag{B.10}$$

Gathering (B.9) and (B.10) provides expansion (B.8). Using the explicit expression of $f_z^{(3)}$, the dominated convergence theorem finally shows that when $z \uparrow +\infty$, remainder T_z in (B.7) tends to some finite constant t_α depending on α only. Gathering terms in (B.6)-(B.8), we are finally left with expansion (2.35) with constant $V_\alpha = -\alpha + \alpha/12 + t_\alpha/6$. Some further calculations would provide $V_\alpha = -\alpha\log(2\pi)/2$, although this actual value does not intervene in our discussion $\square$

Proof of Proposition 2.4.8

Recall definition (B.1) of function g and write equivalently

$$g(z) = \sum_{k\geq 1} g_z(k).$$

where we let $g_z(y) = Az(y^\alpha + Az)^{-1}$. The Euler-Maclaurin summation formula [38] applies again in the form

$$\sum_{k=1}^{J} g_z(k) = \int_1^J g_z(y)\mathrm{d}y + \frac{1}{2}\left[g_z(J) + g_z(1)\right] + \frac{1}{12}\left[g_z'(J) - g_z'(1)\right] + \frac{W_z(J)}{6} \tag{B.11}$$

for given $z \in \mathbb{C} \smallsetminus \mathbb{R}^-$, integer $J \geq 1$ and where

$$\mid W_z(J) \mid \leq \frac{12}{(2\pi)^2} \int_1^J \mid g_z^{(3)}(y) \mid \mathrm{d}y$$

(derivatives of g_z are taken with respect to variable y). Consider the behavior of the r.h.s. of (B.11) as J tends to infinity. Firstly, $g_z(1) = Az/(Az+1)$ and $g_z(J) = O(J^{-\alpha})$ as $J \uparrow +\infty$; secondly, $g_z'(1) = -A\alpha z(1+Az)^{-2}$ together with $g_z'(J) = O(J^{-\alpha-1})$ for large J. Differentiating twice again shows that the third derivative of g_z is $O(y^{-\alpha-3})$ for large positive y and is consequently integrable at infinity. Letting J tend to infinity in (B.11) therefore implies equality

$$g(z) = \int_1^{+\infty} g_z(y)\mathrm{d}y + \frac{1}{2}\frac{Az}{Az+1} + \frac{1}{12}\frac{A\alpha z}{(1+Az)^2} + \frac{W_z}{6} \tag{B.12}$$

where

$$\mid W_z \mid \leq \frac{12}{(2\pi)^2} \int_1^{+\infty} \mid g_z^{(3)}(y) \mid \mathrm{d}y.$$

Using the explicit expression of the derivative $g_z^{(3)}$, it can be simply shown that

$$\mid W_z \mid = O(z^{-2/\alpha}),\ \mid W_z \mid = O(z^{-1}\log z),\ \mid W_z \mid = O(z^{-1}) \tag{B.13}$$

if $\alpha > 2$, $\alpha = 2$ and $1 < \alpha < 2$, respectively. Now, considering the first integral in the r.h.s. of (B.12), the variable change $y = t(Az)^{1/\alpha}$ gives

$$\int_1^{+\infty} g_z(y)\mathrm{d}y = I(Az)^{1/\alpha} - 1 + O\left(\frac{1}{z}\right) \tag{B.14}$$

where $I = L/\alpha = \nu_\alpha^{1/\alpha}$, with integral L introduced in (B.9) for $\alpha > 1$. Expanding all terms in powers of z for large z, it therefore follows from (B.12) and (B.14) that

$$g(z) = \nu_\alpha^{1/\alpha}(Az)^{1/\alpha} - \frac{1}{2} + W_z$$

with W_z estimated in (B.13). For large x, equation (B.1), *i.e.*$g(\theta_x) = x$, then reads

$$A\theta_x = \left[\frac{x}{I} + \frac{1}{2I} + O(W_{\theta_x})\right]^{\alpha} = \left(\frac{x}{I}\right)^{\alpha} + \frac{\alpha}{2I^{\alpha}}x^{\alpha-1} + O(x^{\alpha-2}) \tag{B.15}$$

for $\alpha > 2$ since (B.13) implies $W_{\theta_x} = O(\theta_x^{-2/\alpha}) = O(x^{-2})$ in this case. The case $1 < \alpha < 2$ gives a similar expansion since the remainder is $W_{\theta_x}/x = O(x^{-\alpha}/x) = O(x^{-\alpha-1})$. Finally, the case $\alpha = 2$ yields

$$A\theta_x = \left[\frac{x}{I} + \frac{1}{2I} + O(W_{\theta_x})\right]^{2} = \left(\frac{x}{I}\right)^{2} + O\left(\frac{\log x}{x}\right). \tag{B.16}$$

Gathering results (B.15)-(B.16) finally provides expansions (2.37) for θ_x □

Proof of Proposition 2.4.9

We here verify that conditions (2.32) and (2.33) of Theorem 2.4.6 are satisfied in the case of a Zipf popularity distribution with exponent $\alpha > 1$. Let us first establish convergence result (2.32). Using Lemma 2.4.7, we readily calculate

$$e^{v\sqrt{x}}\frac{F(\theta_x e^{-v/\sqrt{x}})}{F(\theta_x)} = e^{v\sqrt{x}}\exp\left[\alpha(\nu_\alpha A\theta_x)^{1/\alpha}\left(e^{-v/\alpha\sqrt{x}} - 1\right) + \frac{v}{2\sqrt{x}} + \varepsilon(\theta_x e^{-v/\sqrt{x}}) - \varepsilon(\theta_x)\right] \tag{B.17}$$

for any given $v \in \mathbb{C}$ with $\Re(v) = 0$, $|\,\Im(v)\,| \leq a$ and where $\varepsilon(\theta) \to 0$ as $\theta \uparrow +\infty$. By Lemma 2.4.8, we further obtain $\alpha(\nu_\alpha A\theta_x)^{1/\alpha} = \alpha x + \alpha\nu_\alpha k_x + o(k_x)$ and the expansion of $e^{-v/\alpha\sqrt{x}} - 1$ at first order in $1/x$ entails that

$$\alpha(\nu_\alpha A\theta_x)^{1/\alpha}\left(e^{-v/\alpha\sqrt{x}} - 1\right) = -v\sqrt{x} + \frac{v^2}{2\alpha} + O\left(\frac{1}{\sqrt{x}}\right);$$

letting x tend to infinity, we then derive from (B.17) and the previous expansions that

$$e^{v\sqrt{x}}\frac{F(\theta_C = xe^{-v/\sqrt{x}})}{F(\theta_x)} \to \exp\left(\frac{v^2}{2\alpha}\right)$$

so that assumption (2.32) is satisfied with $\sigma^2 = 1/\alpha$.

Let us finally verify boundedness condition (2.33). Rephrasing (B.17) for $v = -iy\sqrt{x}$, we

have

$$\left(\frac{F(\theta_x e^{iy})}{F(\theta_x)}\right)^{1/x} = \exp\left[\frac{\alpha(\nu_\alpha A\theta_x)^{1/\alpha}}{x}\left(e^{iy/\alpha}-1\right) - \frac{iy}{2x} + \frac{\varepsilon(\theta_x e^{iy})-\varepsilon(\theta_x)}{x}\right] \tag{B.18}$$

for any $y \in \mathbb{R}$. But as above, $\alpha(\nu_\alpha A\theta_x)^{1/\alpha}/x$ tends to the constant α when $x \uparrow +\infty$ so that

$$\left|\frac{F(\theta_x e^{iy})}{F(\theta_x)}\right|^{1/x} \leq \mid h(y) \mid^\beta \times \left|\exp\left[\frac{\varepsilon(\theta_x e^{iy})-\varepsilon(\theta_x)}{x}\right]\right| \tag{B.19}$$

for some positive constant β and where

$$h(y) = \left|\exp\left(e^{iy/\alpha}-1\right)\right| = \exp\left(\cos\left(\frac{y}{\alpha}\right)-1\right).$$

Function h is continuous, even and given $\delta > 0$, h is decreasing on interval $[\delta, \pi]$ since $\alpha > 1$, hence $h(y) \leq h(\delta) = \eta_\delta < 1$ for $\delta \leq y \leq \pi$. Using the estimates derived in Appendix B, it is further verified that, given any compact $\mathcal{K} \subset \mathbb{C}$ not containing the origin, we have $\lim_{x\uparrow+\infty} \varepsilon(\theta_x u) = 0$ uniformly with respect to $u \in \mathcal{K}$; this entails that the exponential term in the right-hand side of (B.19) tends to 1 when $x \uparrow +\infty$ uniformly with respect to $u = e^{iy}$, $y \in [\delta, \pi]$. We finally conclude that condition (2.33) is verified. □

Proof of Proposition 2.4.11

Let $q_k(i+1)$, $k \geq 1$, denote the distribution of the input process at cache $(i+1)$, $i \geq 1$. By the same reasoning than that performed in Lemma 2.4.3, we can write

$$q_k(i+1) = q_k(i)\frac{p_k(i)}{p(i)} = q_k\frac{p_k(i)...p_k(1)}{p(i)...p(1)} \tag{B.20}$$

for all $k \in \mathbb{N}$, where $p_k(i)$ (resp. $p(i)$) is the local miss probability of a request for object k at cache i (resp. the averaged local miss probability for all objects requested at cache i) introduced in (2.45) and with notation $q_k = q_k(1)$. As $p_k(i') \to 1$ for all $i' \leq i$ when $k \uparrow +\infty$, we deduce from (B.20) that

$$q_k(i+1) \sim \frac{A(i)}{k^\alpha}$$

when $k \uparrow +\infty$, where $A(i) = A/p(1)p(2)..p(i)$. Apply then estimate (2.42) to obtain

$$p_k(i+1) \sim \frac{1}{1+\theta(i+1)q_k(i+1)} \tag{B.21}$$

where $\theta(i+1) \sim C(i+1)^\alpha / A(i)\nu_\alpha$ and with $q_k(i+1)$ given by (B.20); using the value of $A(i)$ above and the definition $q_k = A/k^\alpha$, the product $\theta(i+1)q_k(i+1)$ reduces to

$$\theta(i+1)q_k(i+1) \sim \frac{x(i+1)^\alpha}{A(i)\nu_\alpha} \cdot q_k \frac{p_k(i)..p_k(1)}{p(i)..p(1)} = \frac{x(i+1)^\alpha}{\nu_\alpha k^\alpha} p_k(i)...M_k(1). \tag{B.22}$$

Writing then $p_k^*(i+1) = p_k^*(i)p_k(i+1)$, asymptotics (B.21) and (B.22) together yield

$$p_k^*(i+1) \sim p_k^*(i) \left[1 + \frac{x(i+1)^\alpha}{\nu_\alpha k^\alpha} p_k(i)...p_k(1)\right]^{-1}$$

so that

$$\frac{1}{p_k^*(i+1)} \sim \frac{1}{p_k(i)} + \frac{x(i+1)^\alpha}{\nu_\alpha k^\alpha}$$

since $\Pi_{j=1}^i p_k(j) = p_k^*(i)$ after (2.44); the latter recursion readily provides expression (2.46) for $p_k^*(i)$, $1 \leq i \leq N$.
Using relation $p_k^*(i+1) = p_k^*(i)p_k(i+1)$ again together with expression (2.46) of $p_r^*(i)$ provides in turn expression (2.46) for $p_k(i)$, $1 \leq i \leq N$ □

Proof of Proposition 2.5.1

We follow the same derivation pattern as the proof of Proposition 2.4.11 detailed in Appendix B.
I) When cache 1 uses the RND replacement policy, we know from Appendix B that the request process at cache 2 is IRM with popularity distribution:

$$q_k(2) = q_k \frac{p_k(1)}{p(1)} \sim \frac{x(1)^{\alpha-1}}{x(1)^\alpha + \nu_\alpha k^\alpha}, \quad k \geq 1,$$

and is asymptotically Zipf for large k. We then follow the proof of Proposition 2.3.11 in Appendix A. Let $S_2(0,t)$ be the number of different objects requested at cache 2 in the time interval $[0,t]$; it verifies

$$\mathbb{E}\left[S_2(0,t)\right] = \sum_{k \geq 1} \left(1 - e^{-q_k(2)t}\right).$$

We then first deduce that

$$\mathbb{E}\left[S_2(0,t)\right] \geq \int_1^{+\infty} \left(1 - e^{-q_u(2)t}\right) \mathrm{d}u. \tag{B.23}$$

Using the variable change $v = x(1)^{\alpha-1}t/(x(1)^\alpha + \nu_\alpha u^\alpha)$ in the latter integral, we further obtain

$$\mathbb{E}\left[S_2(0,t)\right] \geq \left(\frac{x(1)^{\alpha-1}t}{\nu_\alpha}\right)^{\frac{1}{\alpha}} \int_0^{\frac{x(1)^{\alpha-1}t}{x(1)^\alpha+\nu_\alpha}} \frac{1}{\alpha}\left(1-e^{-v}\right) v^{-1-\frac{1}{\alpha}} \left(1-\frac{x(1)v}{t}\right)^{\frac{1}{\alpha}-1} \mathrm{d}v\,.$$

Letting $t \uparrow +\infty$, the monotone convergence theorem for function $v \mapsto (1-x(1)v/t)^{-1+1/\alpha}$ together with a further integration by parts yield

$$\lim_{t\uparrow+\infty} \frac{\mathbb{E}\left[S_2(0,t)\right]^\alpha}{t} \geq \left(\frac{x(1)^{\alpha-1}}{\nu_\alpha}\right)\left[\Gamma\left(1-\frac{1}{\alpha}\right)\right]^\alpha. \tag{B.24}$$

Starting integral (B.23) from $u = 0$ instead of $u = 1$, the latter asymptotic bound is seen to hold also as an upper bound of $\mathbb{E}\left[S_2(0,t)\right]^\alpha/t$, thus showing that (B.24) actually holds as an equality. The local per-object miss rate on the second cache for an LRU cache is then

$$p_k(2) \sim \exp\left[-q_k(2)x(2)^\alpha \left(\lim_{t\uparrow+\infty}\frac{\mathbb{E}\left[S_2(0,t)\right]^\alpha}{t}\right)^{-1}\right]$$

which proves expressions (2.48).

II) When cache 1 applies the LRU replacement policy, the local per-object miss rate at cache 1 is known [20] to equal

$$p_k(1) \sim \exp\left[-\frac{x(1)^\alpha}{k^\alpha\left[\Gamma\left(1-\frac{1}{\alpha}\right)\right]^\alpha}\right] = \exp\left[-\frac{x(1)^\alpha}{\alpha\xi_\alpha k^\alpha}\right]$$

and that the local average miss rate is

$$p(1) \sim \frac{1}{\alpha}\left[\Gamma\left(1-\frac{1}{\alpha}\right)\right]^\alpha \frac{A}{x(1)^{\alpha-1}} = \frac{\xi_\alpha A}{x(1)^{\alpha-1}}.$$

Using IRM assumption for the request process at cache 2, it then follows that the input process at cache 2 is IRM with popularity distribution given by

$$q_k(2) = q_k\frac{p_k(1)}{p(1)} \sim \frac{x(1)^{\alpha-1}}{\xi_\alpha k^\alpha}\exp\left[-\frac{x(1)^\alpha}{\alpha\xi_\alpha k^\alpha}\right], \quad k \geq 1.$$

Note that this distribution is asymptotically Zipf for large k, with coefficient $A'(2) = A/p(1)$. Applying estimate (2.42) to the above defined distribution $q_k(2)$, $k \geq 1$, it then follows that

$$p_k(2) \sim \left(1+q_k(2)\theta'(2)\right)^{-1},$$

where the associated root $\theta'(2)$ is easily estimated by $\theta'(2) = x(2)^\alpha / A'(2)\nu_\alpha$ by using Lemma 2.4.8. We hence derive that

$$p_k(2) \sim \left(1 + \frac{x(2)^\alpha}{A'(2)\nu_\alpha} \frac{Ap_k(1)}{k^\alpha p(1)}\right)^{-1}$$

which leads to expressions (2.49)

Appendix C

RAAQM proofs

Proof of Proposition 7.2.1

Let us denote with $(\widetilde{\mathbf{X}}, \widetilde{p})$ the equilibrium point of Eqs.(7.4-7.5-7.6), where $\mathbf{X}(t) = (X_1(t), \ldots, X_N(t))$. Let us consider the following Lyapunov function

$$V(\mathbf{X}(t), p) = \frac{1}{2}\sum_{i=1}^{N} \Delta X_i(t)^2 + \frac{1}{2}\zeta \Delta p(t)^2 \tag{C.1}$$

where $\Delta X_i(t) = X_i(t) - \widetilde{X}_i$, $\Delta p(t) = p(t) - \widetilde{p}_i$ and $\zeta = \frac{\Delta R_{\max} \beta C \widetilde{X}_i^2}{\Delta p_{\max}}$. Clearly, $V(\widetilde{\mathbf{X}}, \widetilde{p}) = 0$, $V(\mathbf{X}, p) \geq 0$ and

$$\begin{aligned}
\dot{V} &= \sum_{i=1}^{N} \Delta X_i \dot{X}_i + \zeta \Delta p\, \dot{p} \\
&= \sum_{i=1}^{N} \Delta X_i \left(\frac{\eta}{\bar{R}^2} - \beta X_i^2 p \right) + \frac{\beta C^2 \Delta p}{N^2} \sum_{i=1}^{N} \Delta X_i \\
&= \sum_{i=1}^{N} \Delta X_i \left(\frac{\eta}{\bar{R}^2} - \beta X_i^2 p + \frac{\beta C^2 p}{N^2} - \frac{\beta C^2 \eta N^2}{\bar{R}^2 \beta N^2 C^2} \right) \\
&= -\beta p \sum_{i=1}^{N} \Delta X_i (X_i^2 - \widetilde{X}_i^2) = -\beta p \sum_{i=1}^{N} \Delta {X_i}^2 (X_i + \widetilde{X}_i) < 0
\end{aligned}$$

where we omitted the indication of the time variable t. Therefore $\dot{V}(\mathbf{X}, p)$ is negatively semidefinite for any ball including the equilibrium point. This proves that $(\widetilde{\mathbf{X}}, \widetilde{p})$ is a globally stable equilibrium as per [73]. Let us consider the case with delays and start from the same Lyapunov

function.

$$\dot{V} = \sum_{i=1}^{N} \Delta X_i \dot{X}_i + \zeta \Delta p \dot{p} = -\beta \sum_{i=1}^{N} \Delta X_i (X_i^2 p(t-R) - \widetilde{X}_i^2 p(t))$$
$$= -\beta \sum_{i=1}^{N} \Delta X_i \left(X_i^2 p - \widetilde{X}_i^2 p - X_i^2 \int_{t-R}^{t} \dot{p}(u) du \right)$$
$$\leq -\beta \sum_{i=1}^{N} \Delta X_i \left(X_i^2 p - \widetilde{X}_i^2 p - \frac{X_i^2 \Delta p_{\max}}{C \Delta R_{\max}} \int_{t-R}^{t} \Delta X_i(u) du \right)$$

We apply the Lyapunov-Razumikhin theorem [74] to compute a stability region for the set of parameters. Hence, by assuming $V(\mathbf{X}(u), p(u)) \leq V(\mathbf{X}(t), p(t))$ as $u \in [t-R, t]$ we need to show that $\dot{V}(\mathbf{X}(t), p(t)) < 0$ in a non empty region including the equilibrium.

$$\dot{V}(\mathbf{X}(t), p(t)) \leq -\beta \sum_{i=1}^{N} \Delta X_i \left(p(X_i^2 - \widetilde{X}_i^2) - X_i^2 \kappa R \sqrt{\zeta + \Delta X_i^2} \right)$$

with $\kappa = \frac{\Delta p_{\max}}{C \Delta R_{\max}}$. It is easy to see that $\dot{V}(\mathbf{X}(t), p(t)) < 0$ as long as $X_i < \widetilde{X}_i$ for all i. Note that by assuming $X_i > \widetilde{X}_i$ for all i we are computing a smaller stability region, as in general $X_i < \widetilde{X}_i$ may hold for some i only. To have $\dot{V}_t < 0$, when $X_i > \widetilde{X}_i \ \forall i$ it is sufficient to impose $p(1 - \widetilde{X}_i^2 / X_i^2) > \kappa R \sqrt{\zeta + \Delta X_i^2}$. The exact X_i range where the inequality holds can be numerically computed. However, since we are interested in a region of attraction including the equilibrium point, we can provide a smaller region of attraction for $X_i \approx \widetilde{X}_i$. In this case, $\sqrt{\zeta + \Delta X_i^2} \approx \sqrt{\zeta}$ and $X_i + \widetilde{X}_i \approx 2\widetilde{X}_i$. Thus, a region of attraction for the considered equilibrium point is

$$\left\{ (\mathbf{X}(t), p(t)) : X_i < \widetilde{X}_i \left(1 + \frac{\kappa \bar{R} \sqrt{\zeta}}{p_{min}} \right), \forall i \right\}$$

Remark that $k\sqrt{\zeta} = \widetilde{X}_i \sqrt{\frac{\Delta p_{max} \beta}{\Delta R_{max}}} \propto \sqrt{(p_{max} - p_{min})\beta}$.

Appendix D

CCNPL-Sim files

Workload file example

time_unit 0.0000010000 ;
sim_length 500000000000.0000000000 ;
publish_content 0 0.0000000000 100 80000 1 100 0 A = "Orange" B = "dsaphonwmf" ;
publish_content 0 0.0000000000 100 80000 2 100 0 A = "Orange" B = "apcqkuzdae" ;
publish_content 0 0.0000000000 100 80000 3 100 0 A = "Orange" B = "xqqfpidjtj" ;
download_content 2 987642.0000000000 0 A = "Orange" B = "dsaphonwmf" ;
download_content 3 2764038.0000000000 0 A = "Orange" B = "apcqkuzdae" ;
download_content 3 4246918.0000000000 0 A = "Orange" B = "xqqfpidjtj" ;
download_content 2 9876423.0000000000 0 A = "Orange" B = "dsaphonwmf" ;
download_content 2 12456894.0000000000 0 A = "Orange" B = "apcqkuzdae" ;
download_content 4 17589330.0000000000 0 A = "Orange" B = "dsaphonwmf" ;

CCNPL-Sim option file

dci	time	collect and print some stats in output-file every [time] sec.
sim_length	time	sim length expressed in sec (override the length written in the workload).
output	outputfile	output file.
queue_discipline	FIFO/DRR	queuing discipline, FIFO or DRR (applied at each node).
cache	AUTO/LRU/RND/NO [cache size]	caching policy, AUTO means "cache policy/size specified in the topology file; [cache size] in kbit.
startup	time	does not collect stats for [time] sec.
req_order	SEQ/RND	chunk request order (SEQ = 1,2,3,...N or RND).
flow_controller	ICP/FIX win var/fix timer limit	flow controller (applied at each download), Interest window ICP (AIMD) or fix, with size [win], var/fix receiver-time out of [timer] sec., limit indicates the max number of interest re-expressions allowed.
	RAQM pmax beta win limit S	Interest window RAQM (AIMD) with max probability decrese [pmax] and decrease factor [beta]. [win] is the initial window size and [limit] indicates the max number of interest re-expressions allowed. S = [source / no_source] indicates if RAQM collects global or per source RTT stats.
	CBR win rate	Controller with fix window size [win]. Sends an entire window of interests [rate] times per second. No interest re-expression is allowed.
PIT_timer	timer	PIT timer of [timer] sec.
filtering	YES/NO	enable filtering in the PIT table (applied at each node).
workload	fileneame.wl	workload of the simulation.
classes_stat	class	print out statistics for the fisrt [class] popularity classes.
routing	routing.dist	manual routing file.
forwarding	RND/BALANCE	interest forwarding policy.
topology	filename.brite	topology filename.

Annexe E

Synthèse en français

E.1 Introduction

L'évolution d'Internet

Historiquement, les réseaux téléphoniques ont été les premières formes d'architecture de télécommunication à émerger. Le problème qu'ils avaient à résoudre était de trouver un moyen d'établir des circuits permanents entre les téléphones afin qu'ils communiquent. Au début, les circuits étaient assurés par des opérateurs humains qui connectaient manuellement les deux interlocuteurs. Le nombre de terminaux alors grandissant, l'établissement des circuits a dut devenir mécanique, puis ce procédé a finit par devenir totalement électrique afin de réduire les temps de connexion.

Les ordinateurs sont apparus environ un siècle plus tard et, de même que pour les téléphones, un réseau d'interconnexion est devenu nécessaire avec l'évolution des machines. L'objectif principal était de connecter un nombre limité de machines, afin de transférer une petite quantité des données. Le seul exemple de réseau de communication à cette époque était le réseau de téléphonie. Les chercheurs se sont donc concentrés à réduire le délai nécessaire à la mise en place automatique des chemins entre deux ordinateurs. Cependant, pour les ordinateurs, le problème à résoudre était différent : un réseau informatique nécessitait de transporter efficacement des informations codées d'un point à un autre.

La solution retenue par les chercheurs, consistait à envoyer des paquets de données indépendants, transmis en incluant l'adresse de destination finale dans le paquet lui-même. Les premiers réseaux de commutation de paquets sont apparus autour des années 1970, (ARPANET - états-Unis et MINITEL -France). Une étape importante pour le développement de l'architecture d'Internet a été d'introduire les protocoles TCP/IP (1980), qui ont permis l'évolution d'Internet.

Avec l'introduction du World Wide Web (1990), l'utilisation d'Internet a changée ; d'un mode

centré sur les machines (*host-to-host*) elle a évoluée à un mode centré sur les contenus (*content-centric*). Même si l'architecture d'origine a été capable de satisfaire les besoins des nouvelles applications (avec des solutions à niveau applicatif comme DNS, NAT, Proxy, CDN, IPv6, multicast IP, Mobile IP, etc), son objectif initial était de transmettre de petites quantités de données entre deux machines et non la diffusion de contenus à grande échelle.

Plusieurs projets de recherche sont apparus pour réfléchir à une évolution de l'architecture d'Internet qui soit adaptée à l'utilisation qu'on en fait aujourd'hui. Information Centric Networking (ICN), en particulier, propose de résoudre ces problèmes architecturaux en utilisant les contenus nommé comme l'élément central de l'architecture de communication au lieu des adresses des machines.

Le paradigme Information Centric Networking

Afin de répondre à l'inadéquation du design initial d'Internet, des architectures ICN ont été proposées par la communauté de recherche.

Dans les architectures ICN, les données sont identifiées par leur nom (unique dans le réseau). Différents types de schémas de nommage ont été proposés. Parmi les différentes propositions, deux principales catégories peuvent être identifiées : les noms "plats" (étiquettes sans hiérarchie) et les noms hiérarchiques (étiquettes ordonnées par importance e.g. URI). Les deux types de nommage présentent des avantages et des inconvénients en termes de sécurité, d'extensibilité, etc. Le nommage des données reste donc un sujet de recherche ouvert.

La sécurité, est également un aspect important des architectures ICN. En effet, comme les données peuvent provenir de n'importe quel élément du le réseau (cache ou serveur), l'architecture doit prévoir des mécanismes pour vérifier l'authenticité des données reçues. Avec les architectures ICN, la sécurité est strictement liée au nommage. Toutes les propositions ICN présentent des systèmes de sécurité légèrement différents qui exploitent les mécanismes de sécurité traditionnels comme la signature, les certificats, etc.

Grâce au nommage unique, les réseaux ICN connaissent les données qu'ils livrent et chaque noeud du réseau peut utiliser un cache pour stocker les paquets potentiellement intéressants pour les autres utilisateurs. De cette façon, les architectures ICN offrent un système de caches au niveau réseau contrairement à Internet d'aujourd'hui qui transporte simplement les paquets de bout en bout.

En général, dans le modèle de communication ICN, la livraison des données est entièrement contrôlée par le récepteur via des requêtes ou des "abonnements" à des données ou des services. L'architecture ICN a la responsabilité de la résolution du nom des données afin de localiser la copie la plus proche et de la livrer au destinataire. Ils existent deux différentes résolutions des

noms : la recherche dans un système distribué ou bien noeud par noeud.

Différentes propositions ICN ont été formulées :

- **Data Oriented Network Architecture** (DONA) : présentée dans [4], DONA est une des premières propositions ICN ayant pour objectif de redéfinir le système de nommage d'Internet. Il remplace les noms DNS, avec des noms "plats" qui permettent l'auto-certification et fournit des primitives pour les resoudre ;

- **Network of Information** (NetInf) : une autre contribution importante à la recherche sur ICN, a été apportée par les projets européens 4WARD [5] et SAIL [6]. Les deux projets proposent des éléments d'architecture et de dénomination, une résolution de nom, des caches et un système de transport afin de fournir l'accès aux objets nommés dans le réseau. Le principal résultat de ces deux projets est NetInf, un prototype qui implémente l'architecture proposée ;

- **Publish Subscribe Internet Technology** (PURSUIT) : aussi connu sous le nom Publish and Subscribe Internet Routing Protocol (PSIRP), le projet européen précèdent PURSUIT. PURSUIT est une proposition ICN qui fait du paradigme *publish and subscribe* le point central de l'architecture d'Internet du futur. Il prévoit, des points de rendez-vous à travers lesquels les utilisateurs du réseau publient des objets ou s'inscrivent à des données nommées pour les recevoir ;

- **Content Based Networking** (CBN) : CBN [12],[13] propose un mécanisme de *publish and subscribe* couplé avec un routage basé sur le nom. Dans l'architecture CBN les noeuds ne sont pas identifiés en utilisant une adresse réseau unique et les paquets ne doivent pas être directement adressés au noeud spécifique. Les messages sont routés de l'émetteur au récepteur par le réseau CBN en utilisant le nom du message ;

- **Content-Centric Networking** (CCN) : CCN [14] propose une architecture pour Internet du futur, en incluant un schéma de nommage, des fonctionnalités de transfert de données, de sécurité et de diffusion du contenu. Cette proposition a été l'objet de recherche au sein d'un projet [15] financé par le National Science Foundation (NSF) americain.

Content-Centric Networking (CCN)

Les travaux décrits dans cette thèse se concentrent principalement sur la proposition CCN mais, les résultats obtenus, ont une application plus large dans le contexte ICN.

Les noms hiérarchiques et la sécurité

Au lieu d'identifier les machines comme le fait aujourd'hui Internet, l'architecture CCN, identifie chaque paquet d'un objet avec des noms uniques. Les noms contiennent un nombre variable de composants, organisés dans une structure hiérarchique. Par exemple, un fichier vidéo peut être identifié par le nom */parc.com/video/WidgetA.mpg/_v<timestamp>/* et ses paquets avec des noms du type */parc.com/video/WidgetA.mpg /_v<timestamp>/_s_id* où s_id représente l'identifiant du paquet faisant partie de la vidéo.

Afin d'assurer l'intégrité et l'authenticité de l'objet, CCN authentifie la liaison entre le nom et les données identifiées. Le propriétaire de contenu qui souhaite rendre accessibles ses données, distribue le triplet $M_{(N,P,C)} = (N, C, Sign_P(N, C))$ où C représente l'objet, P la clé publique du propriétaire des donnés et N le nom choisi pour représenter C. Lorsqu'un utilisateur reçoit M (N, P, C), il peut attester de l'intégrité et l'authenticité des données en utilisant la clé publique pour vérifier la signature.

Envoi des requêtes

Dans l'architecture CCN, il y a deux différents types de paquets :

- **INTEREST** : utilisé pour demander un objet en utilisant son nom N ;
- **DATA** : tous les noeuds recevant un paquet INTEREST, peuvent y répondre avec un paquet DATA qui contient le nom de l'objet, la signature de sécurité ainsi que les données demandées. Un DATA correspond à un seul et unique INTEREST et le récepteur ne peut pas recevoir de paquets qu'il ne demande pas.

Avec CCN la communication est initiée par les utilisateurs qui envoient un paquet INTEREST. Il est ensuite acheminé vers les localisations possibles des données. Plus précisément, une fois que le routeur reçoit un INTEREST, il vérifie dans son Content Store (CS, un cache local du routeur) s'il a une copie du contenu demandé. Si c'est le cas, le noeud répond par un paquet DATA. Sinon, il vérifie le Pending Interest Table (PIT). Le tableau PIT est utilisé pour garder une trace des INTEREST reçus et retransmis (pas encore satisfaits) et les interfaces ayant envoyé les INTEREST. Si un INTEREST du même nom a déjà été transmis, celui reçu est stoppé et l'interface d'origine est enregistrée. Enfin, si aucune correspondance n'est trouvée dans le CS, ni dans le PIT, le routeur effectue une recherche du plus long préfixe correspondant dans la Forwarding Information Base (FIB, tableau de transfert similaire à celui utilisé par le protocole IP). S'il y a une correspondance dans la FIB, l'INTEREST est transmis et une nouvelle entrée est ajoutée à la table PIT, sinon l'INTEREST est rejeté.

Contrairement aux INTEREST, les DATA ne sont pas acheminés par leur nom sur le réseau et exploitent les informations des PIT pour trouver le chemin de retour vers les demandeurs. Quand un noeud CCN reçoit un paquet DATA, il vérifie d'abord si les données sont déjà stockées dans le CS et, dans ce cas, il rejette le paquet. Après, s'il y a une entrée dans la PIT il transmet le DATA aux interfaces correspondantes et éventuellement le mémorise dans le CS. S'il n'y a pas de correspondance dans le PIT, les données sont simplement jetées. Avec ce mécanisme de transmission de données, le trajet suivi par un DATA, est symétrique au chemin suivi par l'INTEREST (ce qui n'est pas toujours le cas dans l'architecture Internet actuelle).

Routage

Dans l'architecture CCN, le nom des objets est utilisé à la place de l'identifiant des machines (adresse IP). Ainsi, un protocole de routage pour CCN, devrait distribuer des préfixes de noms au lieu de préfixes d'adresses IP, afin de mettre à jour le Routing Infromation Base (RIB - tableau de routage). Comme le routage dans CCN ne devrait pas beaucoup différer de celui adopté avec succès par le protocole IP, des solutions proches des solutions existantes telles que OSPF ou IS-IS (intra-domaine) et BGP (inter-domaine) pourraient être adoptées comme protocole de routage CCN. Cependant, les préfixes de CCN sont plus nombreux que ceux d'IP et le routage fait encore l'objet de recherches.

Gestion du Trafic et des Ressource

Dans l'architecture actuelle d'Internet, les fonctions de la couche transport telles que le contrôle de flux et de congestion, la détection d'erreurs, etc. sont mises en oeuvre au niveau de l'expéditeur et du récepteur. Au contraire, dans CCN, le rôle des machines est complètement différent. Le réseau comporte un récepteur qui demande les données et de nombreux noeuds intermédiaires qui participent au processus de récupération du contenu puisqu'ils peuvent stocker une partie des données demandées. En conséquence, le transport a un seul point de terminaison : le récepteur.

L'architecture originale CCN, ne précise pas les mécanismes de contrôle de transport. Les récepteurs sont tout simplement autorisés à émettre des paquets INTEREST afin de recevoir des DATA en réponse. Cependant, comme pour le protocole TCP, la taille de la fenêtre qui tient compte des INTEREST envoyés peut être variable pour pouvoir atteindre le débit théorique maximum du réseau.

E.2 Modélisation des réseaux CCN

Une compréhension préliminaire des questions de transport et de mise en cache dans le réseau est nécessaire afin d'analyser la performance de l'architecture, de quantifier les avantages potentiels et de guider la conception des protocoles. Dans la première partie de cette thèse, nous analysons les performances perçues par les clients de l'architecture CCN (plus généralement n'importe quel réseau caractérisé à la fois par la présence de cache et par une communication guidée par le récepteur). Dans ce but, nous devons tenir compte de l'interaction entre les deux ressources qui influencent les performances : le stockage et la bande passante.

Dans un premier temps, nous analysons les performances de stockage en étudiant les politiques de remplacement Least Recently Used (LRU - efface l'élément utilise le moins récemment) et Random (RND - efface un élément au hasard). Même pour un seul noeud, l'analyse de la dynamique du cache n'est pas simple. Lorsqu'on passe à plusieurs caches en réseau, il faut prendre en compte le processus de sortie du premier noeud comme le processus de requête aux caches successifs. Dans ce scénario, nous fournissons des formules asymptotiques pour le taux de requêtes non satifaites (taux d'échec) pour des réseaux de caches et dans le cas d'une popularité des objets à queue lourde (la distribution Zipf).

Une fois les performances de stockage étudiées, nous prenons en compte la limitation des ressources en bande passante. Le problème est que dans CCN, les paquets appartenant à un même objet peuvent être récupérés à partir des noeuds différents et peuvent ainsi avoir des temps de téléchargement différents. Dans la dernière partie de la section de modélisation des performances, nous considérons le partage de bande passante max-min (qui vise un partage équitable) et fournissons un outil mathématique pour l'analyse des performances des réseaux CCN.

Modélisation du partage de la mémoire

Description du systéme

L'architecture ICN sur laquelle nous nous sommes concentrés est celle décrite dans [14]. Même si l'on considère une architecture spécifique, notre modèle pour la prévison des performances des caches, a une plus large applicabilité.

Nous considérons des objets identifiés par un nom unique, et stockés de façon permanente dans un (ou plusieurs) serveur(s). Les utilisateurs récupérent un objet à l'aide d'un protocole de transport au récepteur qui prevoit l'envoi d'une requête par paquet. Un protocole de routage garantit le correct acheminement des requêtes vers le(s) serveur(s). Chaque noeud intermédiaire mémorise des requêtes en attente, afin de livrer la donnée demandée vers le récepteur quand il les reçoit. En outre, en utilisant ce mécanisme, les noeuds intermédiaires peuvent éviter de transmettre des

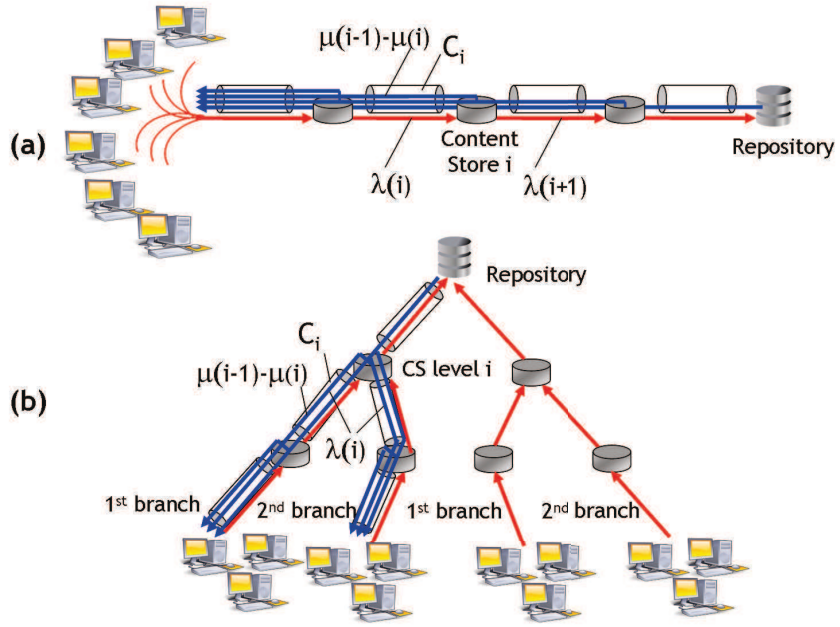


FIGURE E.1 – Topologies réseaux : ligne (a), arbre binaire (b).

demandes multiples pour le même paquet alors que la première est encore en attente (dans ce résumé, nous omettons l'analyse de ce mécanisme). Chaque routeur est aussi équipé d'un cache qui sert pour mémoriser les paquets qui traversent le noeud, afin de satisfaire des futures demandes pour le même objet.

Politique de remplacement "élément utilisé le moins récemment" (LRU)

Description du modèle

Dans le système qu'on analyse, les données peuvent provenir du serveur (qui a une copie permanente de l'objet) ou d'un cache qui est positionné entre le récepteur et le serveur même. Les paquets d'un même objet peuvent donc être récupérés à partir d'emplacements multiples avec différents délais aller-retour (Round Trip Times - RTT). Le débit résultant, et donc le temps de livraison des contenus, sont fortement affectés par cette notion de distance moyenne entre l'utilisateur et les données que nous allons définir explicitement comme *Virtual Round Trip Time* (VRTT temps d'aller-retour virtuel).

Dans cette section, pour l'analyse de la politique de remplacement qui efface du cache l'élément utilisé le moins récemment (LRU pour la suite), nous formulons les hypothèses suivantes :

- Nous considérons un ensemble d'objets différents M répartis dans K classes de popularité ($m = M/K$ fichiers pour chaque classe). Un élément de classe k est demandé avec une

N	Nombre de noeuds du réseau (ou niveaux de l'arbre)
M	Nombre d'objets ($m = M/K$ pour chaque classe k)
$x(i)$	Taille du cache pour le noeuds (ou niveaux de l'arbre) i
$\lambda, \lambda(i)$	Taux des requêtes d'objets au première noeud, noeud $i > 1$
$\mu(i)$	Taux de requêtes non satisfaites au noeud $i > 1$
λ_k	Taux de requêtes pour la classe k
σ	Nombre moyen de paquets d'un objet
$q_k(i)$	Distribution de popularité au noeud i
$p_k(i; x(1)..x(i)) \equiv p_k(i)$	Probabilité d'échec pour une requête de classe k au noeud i
$R(i)$	Délai d'aller retour entre le client et le noeud i
$\text{VRTT}_k(x(1)..x(N)) \equiv \text{VRTT}_k$	Temps d'aller retour virtuel de la classe k,
$\text{RVRTT}_k(i; x(1)..x(i)) \equiv \text{RVRTT}_k(i)$	Résidu VRTT_k au noeud i
T_k	Temps de livraison d'un objet de classe k
W	Numéro maximal de requêtes que le client peut faire en parallèle

TABLE E.1 – Notation

certaine probabilité. Les objets appartenant à la même classe, sont équiprobables ; q_k qui suit une loi Zipf, où $q_k = c/k^\alpha$ pour $k = 1, \ldots, K$ ($\alpha > 1$ et $1/c = \sum_{j=1}^{K} 1/j^\alpha$)

- Les fichiers sont segmentés en σ paquets ou σ représente la taille moyenne du contenu en nombre de paquets (tous de la même taille) ;
- Un noeud i dans le réseau a une taille de mémoire cache de x_i paquets ;
- Les utilisateurs implémentent un protocole de transport qui prévoit l'expression d'un INTEREST par paquets et une fenêtre de taille W qui définit le numéro maximale de requêtés que le client peut faire en parallèle ;
- On définit le temps d'aller-retour virtuel de la classe k, $VRTT_k$, comme le temps moyen (dans l'état d'équilibre) qui s'écoule entre l'envoi d'une demande et la réception du paquet correspondant.

$$\text{VRTT}_k(x(1)..x(N)) = \sum_{i=1}^{N} R(i)(1 - p_k(i; x(1)..x(i)) \prod_{j=1}^{i-1} p_k(j; x(1)..x(j)) \qquad \text{(E.1)}$$

avec $k = 1, \ldots, K$. $\text{VRTT}_k(x(1)..x(i))$ est définie comme la somme pondérée des retards aller-retour $R(i)$ entre le noeud (niveau pour une topologie à arbre) i et l'utilisateur, où les poids correspondent à la probabilité de trouver un paquet de classe k $(1 - p_k(i; x(1)..x(i))$ au noeud i, étant donné qu'il n'a pas été trouvé dans les caches des noeuds précédents.

Pour simplifier la notation, nous omettrons la taille du cache $x(1)..x(i)$ dans la probabilité d'échec (probabilité q'une requête ne soit pas satisfaite) d'un cache.

Processus des requêtes

Le processus des requêtes est structuré en deux niveaux, objet et paquet. Dans cette section, nous construisons un modèle du processus de requêtes fluide. L'hypothèse principale est que les noeuds des plus bas niveau (Fig.E.1) représentent une agrégation des demandes émises par un grand nombre d'utilisateurs. Le processus d'arrivée des requêtes est modélisé par un processus de Markov connu comme Markov Modulated Rate Process (MMRP) [33]. Les demandes des paquets d'un objet de classe k sont générées selon un processus de Poisson d'intensité $\lambda_k = \lambda q_k$, et l'objet est uniformément choisi parmi les m éléments différents dans la classe de popularité donnée. Une demande d'objet coïncide avec la requête de son premier paquet. Une fois qu'un paquet est reçu, une demande d'un nouveau paquet est émise et ainsi de suite jusqu'à la réception du dernier paquet (en cas de fenêtre de requêtes $W = 1$). Notez que le modèle, s'applique aussi au cas plus général d'une fenêtre $W > 1$ paquets demandéS en parallèle.

Simple cache LRU

Nous allons maintenant caractériser la probabilité d'échec du première cache à l'état d'équilibre pour le classes $k = 1, \ldots, K$, selon le processus de requêtes décrit précédemment. Dans [20], les auteurs donnent une caractérisation de la probabilité d'échec du cache pour des objets de 1 paquet, selon des requêtes suivant un processus de Poisson et un nombre d'objets différents qui tend vers l'infini.

Nous étendons leur résultat au cas de : (i) Processus de requêtes MMRP avec moyenne λ_k, (ii) K classes de popularité avec le même nombre de contenus $m = M/K > 1$ dans chacune ; (iii) Des objet de de taille moyenne σ paquets ; (iv) Caractérisation de l'agrégation des requêtes (non décrit dans cette synthèse).

Proposition E.2.1. *Étant donné un processus des requêtes MMRP avec une intensité λ_k, une distribution de popularité $q_k = \frac{c}{k^\alpha}$, $\alpha > 1$, $c > 0$, et la taille moyenne des fichier σ et $x \equiv x(1) \equiv x > 0$ la taille du cache en nombre de paquets, la probabilité stationnaire d'échec pour la requête d'un paquet de la classe k au cache x, indiqué avec $p_k(1, x)$, est donnée par :*

$$p_k(1) \equiv p_k(1; x) \sim e^{-\frac{\lambda}{m} q_k g x^\alpha} \tag{E.2}$$

pour large x, avec $1/g = \lambda c \sigma^\alpha m^{\alpha-1} \Gamma\left(1 - \frac{1}{\alpha}\right)^\alpha$.

Le taux d'échec pour les objets de popularité k, qui constitue le processus d'entrée des caches du second noeud (niveau), a une intensité :

$$\mu_k = \lambda_k \exp\left\{-\frac{\lambda}{m} q_k g x^\alpha\right\}, \qquad \forall k = 1, ..., K. \tag{E.3}$$

Réseaux de caches LRU

Dans cette section, nous appliquons les résultats du modèle LRU du cache simple pour l'étude des topologies en Fig.E.1. Comme dans la section précédente, nous considérons le cas $m = M/K$ et $q_k = c/k\alpha, \forall k = 1, ..., K$.

Nous introduisons ici la notation suivante : $\mu(i)$ et $\lambda(i)$ indique respectivement le taux d'échec et le taux de demande au noeud (niveau) i. Cela signifie que pour un ligne $\lambda(i+1) = \mu(i)$ et pour un arbre binaire $\lambda(i+1) = 2\mu(i)$ (on impose $\mu(0) = \lambda(1) = \lambda$).

Proposition E.2.2. *Compte tenu d'une cascade de caches N Fig.E.1(a), un processus des requête MMRP avec un taux $\lambda(i)$, probabilité d'échec $p_k(i; x(1)..x(i)) \equiv p_k(i)$ et la distribution de popularité $q_k(i) = \prod_{j=1}^{i-1} p_k(j) q_k / \sum_{l=1}^{K} \prod_{j=1}^{l-1} p_l(j) q_l, \quad k = 1, ..., K$, (où $q_k = c/k^\alpha$, $\alpha > 1$) comme entrée pour le cache du i^{me} noeud (niveau), alors $\forall\ 1 < i \leq N$*

$$\log p_k(i) = \prod_{l=1}^{i-1} \left(\frac{x(l+1)}{x(l)}\right)^\alpha p_k(l) \log p_k(1) \tag{E.4}$$

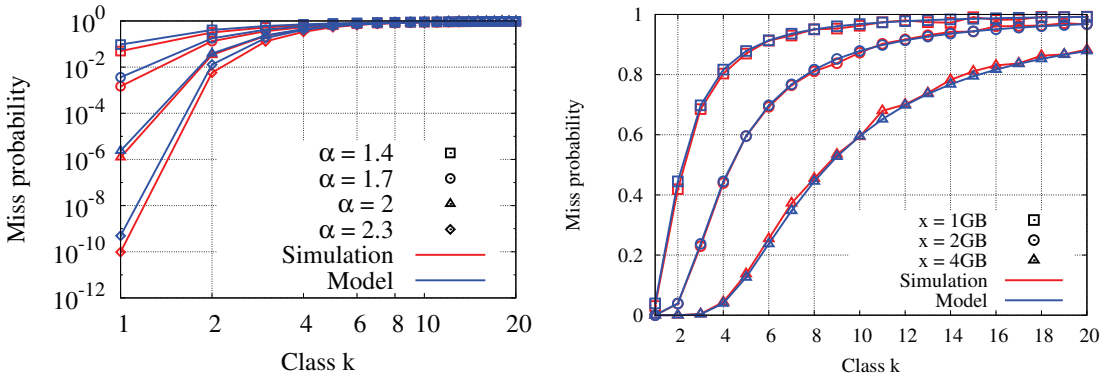
Dans la même façon on peut obtenir la probabilité d'échec dans le cas d'une topologie comme celle indiquée dans Fig. E.1(b) avec la seule différence que $\lambda(i+1) = 2\mu(i)$. Dans ce cas, la probabilité d'échec est la même que dans une ligne de cache.

Résultats de simulation

Dans cette section, nous considérons un catalogue d'objets de $M = 20000$ éléments, divisé en $K = 400$ classes de popularité (décroissante avec k), et $m = 50$ fichiers chacune. La popularité d'une classe q_k est Zipf et un élément de classe k est demandé avec une probabilité q_k/m. Les objets sont divisées en paquets de 10 koctets chacun, et leur taille est géométriquement distribuée avec une moyenne de 690 paquets (voir [35],[36] ; [37]). Les utilisateurs génèrent des demandes de fichiers selon un processus de Poisson ($\lambda = 40$objets / s) et demandent un paquet à la fois (fenêtre de requêtes $W = 1$). Les caches sont de taille $x = 200000$ paquets (2 Goctets) et utilisent LRU. Enfin, les liens réseau ont la même capacité limitée de bande passante (10 Gbps) et le même temps de propagation (2 ms). A noter que dans ces simulation le temps d'aller-retour d'un paquet (RTT - Round Trip Time) est égal au délai de propagation comme le temps de transfert d'un paquet (P/C, où P est la taille d'un paquet) est négligeable.

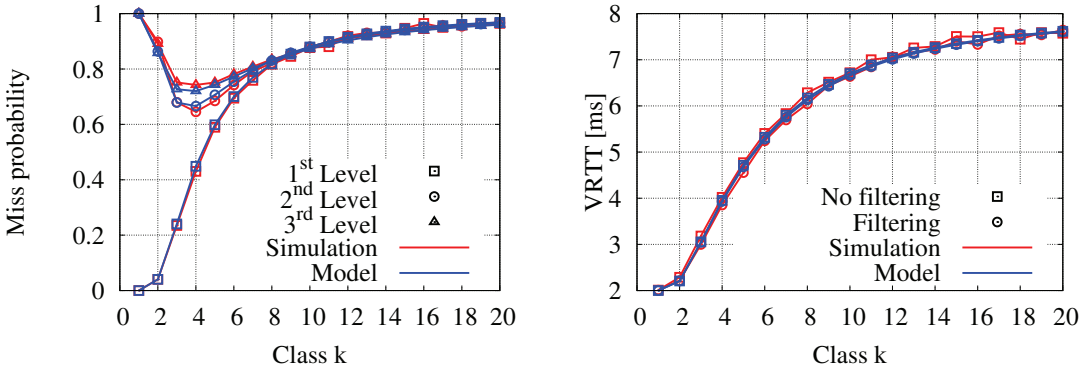
Figs.E.2,E.3 représentent la comparaison entre les formules analytiques proposées et les simulations obtenues avec CCNPL-Sim (Chapitre 9) dans différents cas et montrent une bonne

prévision des performances à travers notre modéle.



(a) (b)

FIGURE E.2 – Simple cache. Approximation asymptotique et simulation de la probabilité d'échec en fonction de la classe de popularité avec différents α(a) et avec différentes tailles de cache et $\alpha = 2$(b) de la politique de remplacement LRU.



(a) (b)

FIGURE E.3 – Arbre binaire de caches ($N = 4$). Approximation asymptotique et simulation de la probabilité d'échec en fonction de la classe de popularité à différentes niveau (a) et VRTT (b) dans un arbre binaire avec $\alpha = 2$ et cache LRU.

Politique de remplacement "élément au hasard" (RND)

Dans cette section nous traitons les problèmes de performances sur des réseaux de caches qui utilisent la politique de remplacement RND (effacement au hasard en cas de cache plein). Contrai-

rement à la section précédente, dans laquelle nous avons présenté les résultats des caches LRU, nous considérons ici des objets de taille $\sigma = 1$ paquet et des classes de popularité composées par un seul objet $m = 1$ ($M = K$).

Simple cache RND

On considère un cache de taille finie qui reçoit des demandes d'objets. Lorsqu'une demande pour un objet ne peut pas être satisfaite (échec), l'objet est récupéré depuis le serveur qui a une copie de tous les objets disponibles. De la même façon que les réseaux LRU analysés, l'objet est mémorisé dans les caches des noeuds qu'il traverse pour arriver au client qui a généré la requête. Contrairement aux caches LRU, quand un objet doit être stocké, si la mémoire est pleine, le fichier qui doit être remplacé est choisi au hasard, uniformément entre les objets présents dans le cache. Compte tenu du nombre total des objets M, la probabilité que l'objet k soit demandé au cache suit une loi Zipf défine comme $q_k = \frac{A}{k^\alpha}$, $1 \leq k \leq M$ où $A = 1/\zeta(\alpha)$ (ζ est la fonction Zeta de Riemann).

Soit $x(1) \equiv x \leq M$ la taille du cache, les probabilités stationnaires d'échec totales et par objet au premier noeud sont identifiées par $p(1;x)$ et $p_k(1;x)$. Une expression générale de la formule combinatoire de $p(1;x)$ a été donnée dans [25] pour toute distribution de popularité. Nous calculons ici les formules asymptotiques pour $p(1;x)$ et $p_k(1;x)$ avec grand x et une distribution de popularité Zipf avec $\alpha > 1$.

Proposition E.2.3. *Pour une distribution de popularité Zipf avec un exposant $\alpha > 1$, un cache de taille x :*

$$\begin{aligned} p(1) \equiv p(1;x) &\sim \nu_\alpha x q_x = \frac{A\nu_\alpha}{x^{\alpha-1}} \\ p_k(1) \equiv p_k(1;x) &\sim \frac{\nu_\alpha k^\alpha}{x^\alpha + \nu_\alpha k^\alpha} \end{aligned} \tag{E.5}$$

pour large x, avec $\nu_\alpha = \left(\frac{\pi/\alpha}{\sin(\pi/\alpha)}\right)^\alpha$.
où $\lim_{\alpha\to+\infty} \nu_\alpha = 1$ et $\nu_\alpha \sim 1/(\alpha-1)$ avec $\alpha \to 1$.

Réseaux de caches RND

Nous généralisons le modèle de cache simple pour une ligne (suite) de cache de taille $x(i)$ (i représente le noeud). Comme dans le cas de réseau de caches LRU, les demandes d'objets sont acheminées vers le serveur jusqu'à ce qu'elles soient satisfaites par un cache (ou le serveur même). Une fois l'objet trouvé, la data est expédiée en arrière vers les clients et stockée dans le routeur qui parcourt le chemin inverse de la requête.

En supposant que tous les caches considérés en isolement se comportent comme un seul cache avec entrées indépendantes (Independence Reference Model - IRM) generées par les échecs aux

noeuds fils, on calcule les asymptotes pour la probabilité d'échec globale $p(i; x(1)..x(i))$ et par objet $p_k(i; x(1)..x(i)))$ aux différents niveaux du réseau.

Proposition E.2.4. *Pour une ligne de caches, on suppose un processus de demande qui satisfait le modèle IRM qui suit une loi de popularité Zipf avec $\alpha > 1$ et un modèle IRM pour tous les caches $i \geq 2$. Pour tout $i \in \{1, ..., N\}$ et caches de grande taille $x(1), ..., x(i)$, les probabilités d'échec $p_k(i)$ et $p(i)$ sont données par :*

$$p_k(i) \equiv p_k(i; x(1)..x(i)) \sim \frac{\nu_\alpha k^\alpha + \sum_{j=1}^{i-1} x(j)^\alpha}{\nu_\alpha k^\alpha + \sum_{j=1}^{i} x(j)^\alpha} \qquad (E.6)$$

$$p(i) \equiv p(i; x(1)..x(i)) = \sum_{k \geq 1} p_k(i) q_k(i)$$

pour large x et ν_α introduit dans la Proposition E.2.3
où $q_k(i)$, $k \geq 1$, définit la distribution de popularité du processus d'entrée au cache i.

Corollary E.2.5. *Considérons un arbre homogène, où toutes les feuilles sont situées à une profondeur commune de la racine, et la taille des caches situés au même niveau est égale à $x(i)$. Si l'hypothèse de requêtes indépendantes est valable pour l'ensemble des noeuds dans l'arbre, les résultats de la Proposition E.2.4 peut s'étendre à l'arbre ou le processus des requêtes aux feuilles a une distribution de popularité Zipf avec $\alpha > 1$.*

Réseaux de caches mixte LRU et RND

Nous considérons aussi dans cette section le cas d'un réseau ligne à deux noeuds, où le mécanisme de remplacement RND est utilisé à un noeud et le mécanisme LRU est utilisé à l'autre. Comme dans la section précédente, ces résultats sont aussi valables dans le cas d'un arbre homogène.

Proposition E.2.6. *Pour une suite de 2-caches, on suppose que les requêtes au premier cache suivent une loi qui respecte le modèle IRM avec une loi de popularité Zipf ($\alpha > 1$) et que l'hypothèse d'indépendance des requêtes tiens pour le deuxième cache.* ***I)*** *Si le cache 1 utilise la politique de remplacement RND (et le cache cache 2 LRU), alors $p_k(2)$ est :*

$$p_k(2) \sim \exp\left(-\frac{\nu_\alpha x(2)^\alpha}{\alpha \xi_\alpha \left(\nu_\alpha k^\alpha + x(1)^\alpha\right)}\right) \qquad (E.7)$$

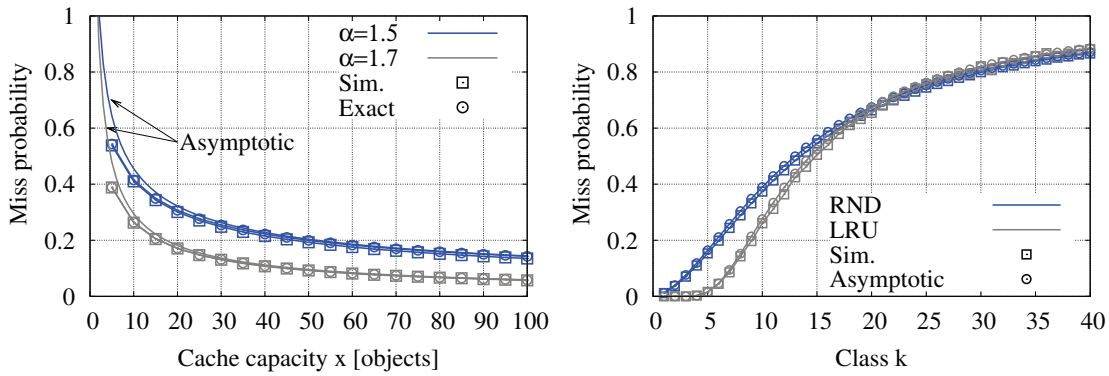
II) *Si le cache 1 utilise la politique de remplacement LRU (et le cache cache 2 RND), alors $p_k(2)$*

est :

$$p_k(2) \sim \frac{\nu_\alpha k^\alpha}{\nu_\alpha k^\alpha + x(2)^\alpha \exp\left(-\frac{x(1)^\alpha}{\alpha \xi_\alpha k^\alpha}\right)} \tag{E.8}$$

pour cache sizes $x(1)$, $x(2)$, ν_α *introduit dans E.2.3*
et $\xi_\alpha = \frac{1}{\alpha}\left[\Gamma\left(1-\frac{1}{\alpha}\right)\right]^\alpha$

Résultats de simulation



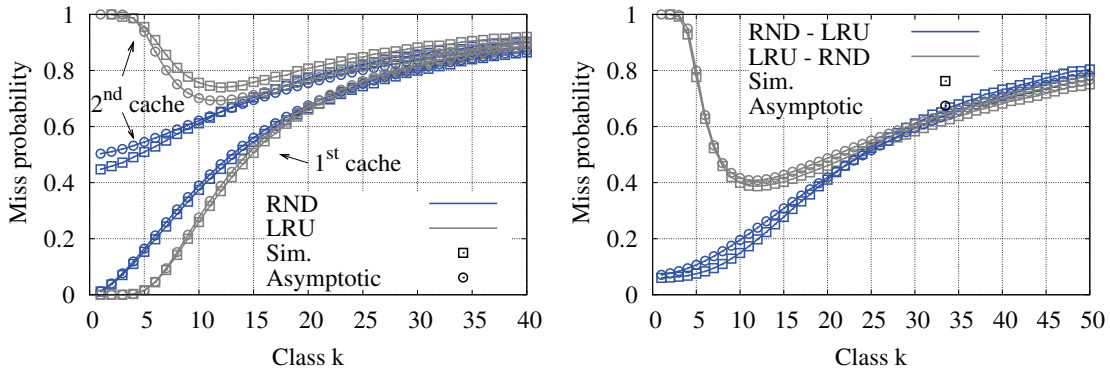
(a) (b)

FIGURE E.4 – Simple cache. Formule exacte, simulation et formule asymptotiques pour $p(1)$ dans le cas RND (a) et comparaison entre simulation et formule asymptotiques pour $p_k(1)$ (b) avec $\alpha = 1.7$.

Nous considérons dans cette section un catalogue d'objets de $M = 20000$ éléments et requêtes qui suit une loi IRM pour montrer la bonne approximation de notre modèle, à travers des simulations (obtenues avec CCNPL-Sim Chapitre 9), et la comparaison entre RND et LRU. Les Figs. E.4, E.5, montrent le comportement des formules asymptotiques dans différents cas de cache simple ou de réseaux de caches.

Modélisation du partage de la bande passante

Afin de caractériser les performances des réseaux des caches, nous prenons en compte l'interaction entre le stockage et le partage de la bande passante. Nous considérons la probabilité d'échec calculée pour la politique de remplacement LRU pour caractériser le partage de la bande passante en CCN (même si le modèle peut être étendu à RND).



(a) (b)

FIGURE E.5 – Ligne de deux caches. Formule asymptotique et simulation pour la probabilité d'échec au noeuds 1,2 $p_k(1)$, $p_k(2)$ (a) et probabilité d'échec dans un réseau de cache RND et LRU (b) avec $\alpha = 1.7$, $x(1) = x(2) = 25$ objets.

En CCN, un récepteur peut télécharger des données à partir de plusieurs noeuds et partager la bande passante avec d'autres transferts simultanés. Nous considérons max-min comme objectif d'équité pour le partage de la bande passante (généralement utilisé dans les réseaux d'aujourd'hui) qui consiste à allouer de manière équitable la bande passante disponible par incréments infinitésimaux, jusqu'à que tous les flux soient satisfaits ou que le lien soit complètement saturé.

Description du modèle

Dans les réseaux des caches, une requête satisfaite par le noeud i déclenche une transmission du paquet correspondant vers le client à travers la route i (flèches bleues dans la Fig.E.1), ou le chemin i représente l'ensemble des liens $l \ni i$ de l'utilisateur au i-ème noeud.

Chaque chemin i est caractérisé par un certain nombre de transferts en parallèles N_i, qui partagent la même bande passante. N_i est un processus aléatoire de naissance et de mort de Markov, avec un taux de naissance $\mu(i-1) - \mu(i)$ ($\mu(0) \equiv \lambda$), et de mortalité $n_i\gamma_i/(\sigma P)$ lorsque n_i transferts parallèles sont en cours. Le taux de mortalité, $n_i\gamma_i/(\sigma P)$, est déterminé par le partage équitable (max-min) de la bande passante défini comme γ_i octets/s (associé à chaque route i). Nous supposons que la bande passante est équitablement répartie entre chaque transfert dans le sens max-min où, pour un lien donné l et pour tous les flux étranglés par l, γ_l est égal à :

$$f_l = \frac{C_l - \sum_{j \in N_l^{nb}} \gamma_{j,l}}{N_l - N_l^{nb}}, \tag{E.9}$$

où N_l est le nombre de flux qui traversent le lien l et $\sum_{j \in N_l^{nb}} \gamma_{j,l}$, N_l^{nb} sont respectivement la somme des débits et le nombre de flux dont la demande est inférieure au débit de partage équitable du lien (flux qui ne sont pas étranglés par le lien l).

En tenant compte de la contrainte de bande passante $\sum_{i \ni l} n_i \gamma_i \leq C_l, \qquad$ pour tous l, il existe au moins un lien $l \ni i$ qui représente le goulot d'étranglement pour une route i tel que :

$$\sum_{i \ni l} n_i \gamma_i = C_l \qquad \text{and} \qquad \gamma_i = \bigvee_{i' \ni l} \gamma_{i'} \tag{E.10}$$

où $\bigvee$ est l'operateur de maximum.

Temps moyen de transfert des objets

Comme le montre la Fig.E.1, le lien descendant $i > 1$ est partagée entre tous les chemins j, avec $j > i$. Par définition, la charge du lien i est :

$$\rho_i \equiv \frac{r_i}{C_i}, \tag{E.11}$$

où r_i est le débit moyen pour le lien i. Le débit des DATA est donné par l'expression $\sigma \sum_{j=i}^{N} (\mu(j-1) - \mu(j))$, ce qui tient compte de tous les paquets qui traversent le lien i pour redesendre vers le client. Si on substitue ce debit dans la formule de la charge du lien i, on obtient :

$$\rho_i \equiv \frac{\sigma P}{C_i} \sum_{j=i}^{N} (\mu(j-1) - \mu(j)) = \frac{\sigma P}{C_i} \mu(i-1), \tag{E.12}$$

où ρ_i satifait la condition de stabilité $\rho_i < 1 \; \forall i = 1, \ldots, N$.

Si on suppose que le temps de propagation de paquet (PD_{up}, PD_{down}) est négligeable par rapport au temps de transfert P/γ_i, le délai de livraison d'un paquet R_i en utilisant la route i est $R_i = PD_{up} + PD_{down} + P/\gamma_i \approx P/\gamma_i$.

Proposition E.2.7. *Si $\rho_i < 1$, $\forall i \geq 1$, le temps moyen de transfert des objets de classe k, $\forall k =$*

$1, ..., K$ *est borné par*

$$\mathbb{E}[T_k] \leq \sum_{i=1}^{N} \bigvee_{j \ni i} \frac{\sigma P}{C_j(1-\rho_j)}(1-p_k(i)) \prod_{j=1}^{i-1} p_k(j), \tag{E.13}$$

où le temps de transfert $\bigvee_{j \ni i} \frac{\sigma P}{C_j(1-\rho_j)}$ pour la route i est multiplié par la probabilité de l'utiliser.

Résultats de simulation

Pour vérifier les conclusions analytiques et évaluer la précision du modèle, nous utilisons CCNPL-Sim (Chapitre 9) avec des files d'attente qui implémentent le mécanisme Deficit Round Robin [42] afin d'imposer l'équité max-min.

Si on considère une ligne de cache avec $N = 3$ (2 caches, 1 serveur), en changeant le débit maximal des liens, quatre cas différents peuvent être identifiés :

Cas I) $C_1(1-\rho_1) < C_2(1-\rho_2), C_1(1-\rho_1) < C_3(1-\rho_3)$.

Dans ce cas, Eq. (E.13) devient

$$\mathbf{E}[T_k] = \frac{\sigma P}{C_1(1-\rho_1)},$$

Cas II) $C_2(1-\rho_2) < C_1(1-\rho_1), C_2(1-\rho_2) < C_3(1-\rho_3)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)) + \frac{\sigma P}{C_2(1-\rho_2)}p_k(1).$$

Cas III) $C_3(1-\rho_3) < C_2(1-\rho_2) < C_1(1-\rho_1)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)) \quad + \frac{\sigma P}{C_2(1-\rho_2)}p_k(1)(1-p_k(2)) + \frac{\sigma P}{C_3(1-\rho_3)}p_k(1)p_k(2)$$

Cas IV) $C_3(1-\rho_3) < C_1(1-\rho_1) < C_2(1-\rho_2)$.

$$\mathbf{E}[T_k] \leq \frac{\sigma P}{C_1(1-\rho_1)}(1-p_k(1)p_k(2)) + \frac{\sigma P}{C_3(1-\rho_3)}p_k(1)p_k(2).$$

Nous considérons $M = 20000$ objets, également partitionné en $K = 2000$ classes, de taille moyenne σ = 100 paquets de taille $P = 10$ koctets. La distribution de la popularité des fichiers est supposé être Zipf(α) avec $\alpha = 2$ et le retard de propagation des liens est égale à 10 μ s. Chaque cache implémente la politique de remplacement LRU est a un taille de $x(i) = 5000$ paquets pour $i = 1, 2$. Enfin, le taux des demande d'objets est $\lambda = 1$ objets / s et la fenêtre d'envoi des requêtes $W = 2$ (2 interest en parallèle).

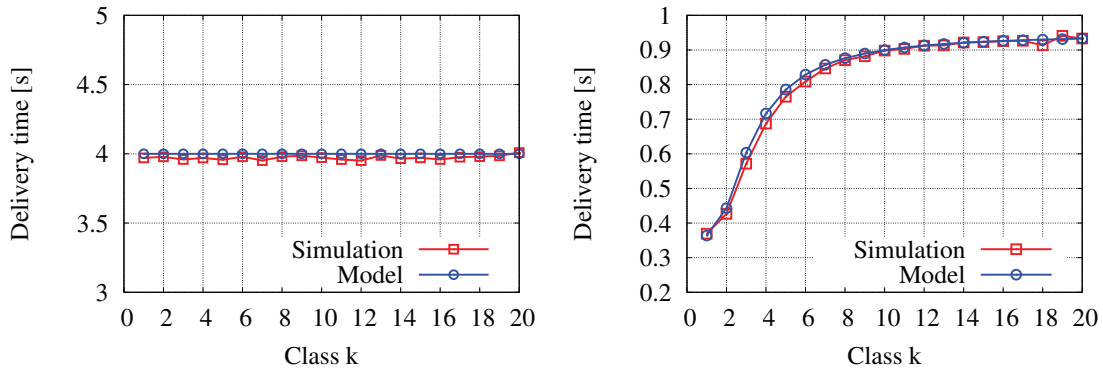


FIGURE E.6 – Ligne de deux caches. Temps moyen de transfert des objets en fonction de la classe de popularité k dans les cas I $(C_1, C_2, C_3) = (10, 20, 30)$Moctets/s (a) et II $(C_1, C_2, C_3) = (30, 10, 20)$ Moctets/s (b).

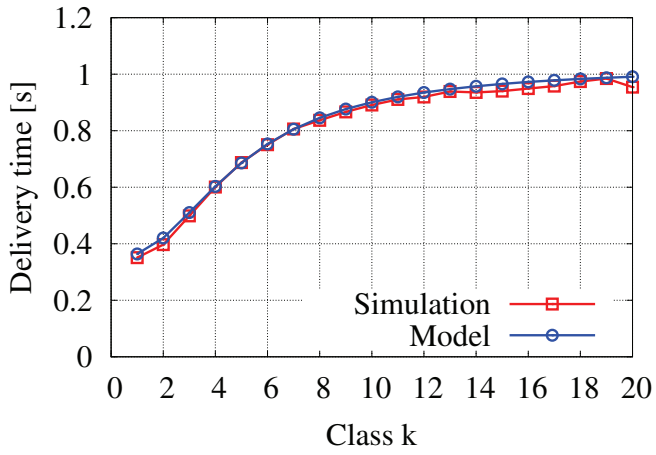


FIGURE E.7 – Arbre binaire $N = 3$. Temps moyen de transfert des objets en fonction de la classe de popularité k dans le cas de $(C_1, C_2, C_3) = (30, 20, 10)$ Moctets/s

Figs.E.7,E.7 montrent la comparaison entre les résultats analytiques et ceux des simulations en confirmant la bonne approximation de notre modèle.

E.3 Protocoles de transport et mécanismes d'acheminement des requêtes pour CCN

Dans les sections précédentes, nous avons analysé les performances de l'architecture CCN en étudiant le partage du stockage et de la bande passante. Nous utilisons les résultats analytiques obtenus dans la première partie comme référence pour la conception et l'analyse des mécanismes de transport et d'acheminement des requêtes dans CCN. En raison de l'absence de connexion dans le modèle de communication CCN, un protocole de contrôle de congestion au récepteur est le choix le plus naturel.

Dans un premier temps nous introduisons un protocole de contrôle de congestion, Interest Control Protocol (ICP), qui adapte le nombre maximal d'INTEREST que le récepteur est autorisé à envoyer en parallèle (fenêtre) en utilisant le mécanisme Additive Increase Multiplicative Decrease (AIMD) communément adopté dans le protocole TCP. Grâce au modèle de communication "un à plusieurs" et au fait que les demandes (INTEREST) et les données (DATA) suivent le même chemin en direction opposée, l'architecture CCN peut aussi prévoir un mécanisme de contrôle de congestion noeud par noeud. Dans la deuxième partie de cette section on présente ce type de mécanisme (couplé avec le protocole de transport au récepteur) qui a pour but d'accélérér la détection/réaction à la congestion.

En analysant ICP à travers des nombreuses simulations, on à trouvé que le protocole est inefficace lorsque les chemins ont des différences significatives en termes de capacité et de temps de propagation. Pour faire face à cette limitation, on a modifié notre protocole de contrôle de congestion en utilisant le mecanisme Remote Adaptive Active Queue Management (RAAQM) pour contrôler efficacement plusieurs chemins en parallèle. En outre, nous introduisons aussi un mécanisme d'acheminement des requêtes qui prends la décision à travers des mensurations de performance locale par interface e par préfixe.

Mécanisme de contrôle des requêtes (ICP - Interest Control Protocol)

Objectifs du Protocole

Un protocole de transport est responsable du téléchargement des données de manière efficace. Pour ce faire en CCN, le protocole doit contrôler le débit de demande (et donc implicitement de réception) et l'adapter en fonction des ressources disponibles dans le réseau. En CCN, une session de transport est réalisée par des requêtes (INTEREST) progressivement envoyées par le récepteur, routé par le nom vers un serveur. Dans le processus de conception d'un protocole de transport et de contrôle de congestion pour CCN (ICP), les objectifs sont les suivants :

- **Fiabilité** : le premier but est de garantir les transferts de données de manière fiable en ré-

exprimant les INTEREST pour récupérer des pertes de paquets. La retransmissions d'une requête est déclenchée après l'expiration d'un temporisateur τ, maintenu au niveau du récepteur.

- **Efficacité** : le deuxième objectif d'ICP est de minimiser le temps d'exécution des transferts de données. Pour cela, ICP utilise AIMD (Additive Increase Multiplicative Decrease) comme mécanisme d'adaptation de la taille de la fenêtre des requêtes. Le choix de ce mécanisme est motivé par le succès avec lequel AIMD a été utilisé avec dans le cas de TCP (même appliqué au récepteur).

- **Equité** : le troisième but d'ICP est de parvenir à une répartition équitable de la bande passante entre les flux, partageant la même route.

Description du protocole

$\mathcal{P}$	ensemble de chemins entre l'utilisateur et le serveur(s)
$\mathcal{R}^{\phi}$	ensemble des routes pour le chemin ϕ (i.e. nombre de noeuds traversés par un chemin)
$R(r_i^{\phi})$	Délai d'aller-retour pour la route r_i^{ϕ} (du client j'usqu'au i^{me} noeud dans le chemin ϕ)
W	Fenêtre des requetes
η,β	Paramètres d'augmentation et diminution pour W
τ	Temporisateur pour la retransmission des requêtes
$W_k(t)$	Fenêtre des requêtes pour un objet de classe de popularité k au temps t
$X_k(t)$	Débit de requêtes pour la classe de popularité k au temps t
$Q(i,t)$	Taille se la file d'attente pour le noued i au temps t

TABLE E.2 – Notation

Dans CCN, les paquets de données sont identifiés par un nom unique (le nom du contenu plus un identifiant de paquet) et sont demandés par le client dans l'ordre décidé par l'application. Dans notre conception, la fenêtre de réception W, définit le nombre maximum des requêtes qu'un récepteur est autorisé à envoyer en parallèle. La taille de la fenêtre W, augmente ou diminue suivant un mécanisme AIMD, mieux détaillé dans les sections suivantes.

Augmentation de la fenêtre des requêtes

W est augmenté de η/W à la réception de chaque paquet. Cette opération est réalisée en utilisant une arithmétique entière, de telle sorte que W est incrémenté de η (mis à 1 comme valeur par défaut) quand une fenêtre complète a été reçue. Fig. E.8(a) montre l'évolution de la fenêtre au début d'un flux.

Diminution de la fenêtre des requêtes

Chaque INTEREST dans notre protocole est associé à un temporisateur τ. Pour ICP, l'expiration

de τ est considérée comme une indication de congestion et le protocole réagit en multipliant W par un facteur $\beta < 1$, pas plus qu'une fois pour une durée égale à τ comme montré en Fig.E.8(b), pour éviter de diminuer W plusieurs fois consécutives. Notez que la ré-expression d'une requête, est nécessaire pour compenser les pertes, même si les données peuvent seulement être retardées au goulot d'étranglement et non perdues. Dans le même temps, une retransmission envoyée juste après un délai d'attente peut être bloquée par la PIT au premier noeud (INTEREST 7,8 dans Fig.E.8(b)). Pour ces raisons, le choix de τ est très important pour ICP.

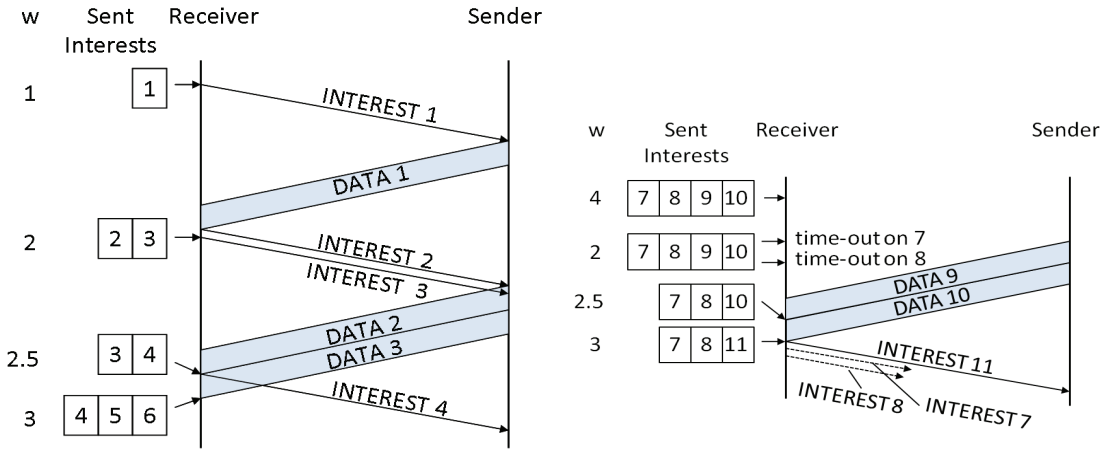


FIGURE E.8 – Évolution de la fenêtre dans ICP : augmentation (gauche), diminution (droite).

Temporisateur

Le bon réglage de la valeur des temporisateurs pour les INTERESTS (τ) et les temporisateurs du tableau PIT (qui permet de supprimer une instance dans le tableau PIT sans avoir satisfait l'INTEREST) est essentiel pour CCN. En général, τ doit être réglé plus grand que le retard de réseau minimum pour eviter une expiration du temporisateur PIT avant la réception du paquet. En outre, une valeur minimale de τ (non connue a priori) est nécessaire pour garantir une utilisation élevée de la bande passante disponible. Un compromis, est proposé par un temporisateur variable en fonction de l'estimation du retard aller-retour minimal (voir par exemple TCP-LP [53]). τ peut donc être variable ; dans ce cas, ICP estime les délais aller-retour max et min ($\mathrm{RTT}_{\max}$ et $\mathrm{RTT}_{\min}$) en mettant à jour, à chaque réception de paquet, une fenêtre coulissante (30 paquets dans notre proposition, en excluant les paquets retransmis).

$$\tau = \mathrm{RTT}_{min} + (\mathrm{RTT}_{max} - \mathrm{RTT}_{min})\delta \tag{E.14}$$

avec $\delta < 1$ (example $\delta = 0,8$ par défaut dans notre proposition). Notez que dans notre conception, un historique complet d'échantillons RTT est nécessaire avant de déclencher une diminution de la fenêtre.

Le temporisateur PIT est fondamental en CCN pour limiter le nombre de requêtes en attente de réponse (*pending* dans le PIT) et, par conséquent, la taille du tableau PIT. Plus cette valeur est élevée, plus grand est le nombre de requêtes filtrées (non transmises vers le serveur parce qu'un autre INTERST identique est en attente). En même temps, une valeur petite pour le temporisateur PIT, implique un grand nombre de retransmissions des requêtes inutiles. Dans cette section, le temporisateur PIT optimale n'est pas étudié, et nous utilisons simplement un temporisateur suffisamment grand et jamais plus petit que τ.

Analyse du protocole en cas de goulot d'étranglement simple

Dans le cas d'un seul goulot d'étranglement sur un lien de C paquet/s, pour un seul telechargement à la fois, en considérant un délai de propagation des paquets constant R et temporisateur de retransmission $\tau > R$, on peut écrire les équations qui décrivent l'évolution du protocole :

$$\begin{aligned} \dot{X}(t) &= \frac{\eta}{(R+Q(t)/C)^2} - \beta X(t)\mathbb{1}_{\{R+Q(t)/C=\tau\}}, \\ \dot{Q}(t) &= X(t) - C\mathbb{1}_{\{Q(t)>0\}} \end{aligned} \tag{E.15}$$

Le débit des requêtes est proportionnel à la taille de la fenêtre des INTEREST ($W(t)$) et donc $X(t) = W(t)/(R + Q(t)/C)$ où $Q(t)/C$ est le délai de file d'attente.

Proposition E.3.1. *La solution à l'état d'équilibre des Eqs.(E.15) est périodique avec un cycle de longueur $\widetilde{T}$, $0 < \widetilde{T} < \frac{2C\beta\tau^2}{(2-\beta)\eta}$. Étant donné $\tau > \tau_{min}$, avec $\tau_{min} = R + \frac{2\eta}{C}(\frac{2-\beta}{\beta})^2$, les valeurs moyennes à l'équilibre pour $X(t)$ et $Q(t)$ sur $\widetilde{T}$ sont :*

$$\begin{aligned} \widetilde{X} &= \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} X(t)dt = C, \\ \widetilde{Q} &= \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} Q(t)dt \in [Q_L, C(\tau - R)] \end{aligned} \tag{E.16}$$

où $Q_L = C(\tau - R) - \frac{1}{3\eta}(\frac{C(\tau-R)\beta}{2-\beta})^2$

Pour le cas de plusieurs flux en parallèle qui partagent le même goulot d'étranglement, Prop.E.3.1, peut être généralisée.

Proposition E.3.2. *Étant donné un nombre variable de flux qui partagent le même goulot d'étranglement, qui sont générés suivant un processus de Poisson $\mathcal{N}$ avec taux λ, chacun composé de σ paquets en moyenne, le débit des requêtes est $\mathbb{E}[\widetilde{X}] = C(1-\rho)$, où $\rho = \lambda\sigma/C$ représente la charge du lien.*

Analyse du protocole en cas de plusieurs goulot d'étranglement

On étend dans cette section notre analyse au cas dans lequel plusieurs goulots d'étranglement existent (pour la présence des caches). Comme décrit dans la partie d'analyse des performances, chaque chemin est caractérisé par un temps de transfert aller-retour moyen (VRTT). En particulier :

Definition E.3.3. *$VRTT_k$ représente l'intervalle de temps moyen entre l'envoi d'un INTEREST et la réception du paquets de donnée, pour les paquets des objets de classe de popularité k.*

$VRTT_k^\phi$ est définie pour un chemin donné ϕ depuis l'utilisateur vers un serveur comme une somme pondérée des délais d'aller-retour $R(r_i^\phi)$ associés aux sous-chemins r_i^{phi}, c'est à dire la route entre l'utilisateur et le i^{me} noeud, avec $i = 1...N$. Le délai $R(r_i^\phi)$ est multiplié par la probabilité stationnaire de trouver un paquet de classe k dans le noeud i, étant donné qu'il n'a pas été trouvé dans les caches précédents. Pour $k = 1, \ldots, K$,

$$VRTT_k^\phi = \sum_{i:r_i^\phi \in \mathcal{R}^\phi} R(r_i^\phi)(1 - p_k(r_i^\phi)) \prod_{j=1}^{i-1} p_k(r_j^\phi) \tag{E.17}$$

Comme dans le cas de goulot d'étranglement simple, on peut écrire les équations qui décrivent l'évolution du protocole dans le cas de plusieurs goulot d'étranglement :

$$\begin{aligned}
X_k(t) &= \sum_{i=1}^{N} X(i,t)(1 - p_k(i)) \prod_{j=1}^{i-1} p_k(j) \\
\dot{X}(i,t) &= \frac{\eta}{\text{VRTT}(i,t)^2} - \beta X(i,t) \mathbb{1}_{\{\text{VRTT}(i,t)=\tau\}}, \\
\dot{Q}_i(t) &= n_i X(i,t) - C_i \mathbb{1}_{\{Q_i(t)>0\}} + \sum_{l=i+1}^{N} \Big(\prod_{j=i+1}^{l} \mathbb{1}_{\{Q_j(t)=0\}} n_j X(l,t) + \mathbb{1}_{\{Q_l(t)>0\}} C_l \Big)
\end{aligned} \tag{E.18}$$

où $\text{VRTT}(i,t) = \sum_{j=1}^{i} \frac{Q_j(t)}{C_j}, \quad i = 1, ..., N$ est le délai aller-retour calculé comme la somme des délais des files d'attentes associées au chemin r_i.

Proposition E.3.4. *La solution à l'état d'équilibre des Eqs.(E.18) est périodique avec un cycle de*

longueur $\widetilde{T}$ *:*

$$\widetilde{X}_k = \sum_{i=1}^{N}(1-p_k(i))\prod_{j=1}^{i-1}p_k(j)\widetilde{X}(i),$$
$$\widetilde{X}(i) = \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} X(i,t)dt = \gamma(i),$$
$$\widetilde{VRTT(i)} \equiv \frac{1}{\widetilde{T}}\int_0^{\widetilde{T}} VRTT(i,t)dt = \frac{1}{\gamma(i)}, \tag{E.19}$$

où $\gamma(i)$ *est le débit max-min de la route* i *définie dans Sec.E.2.*

Résultats de simulation

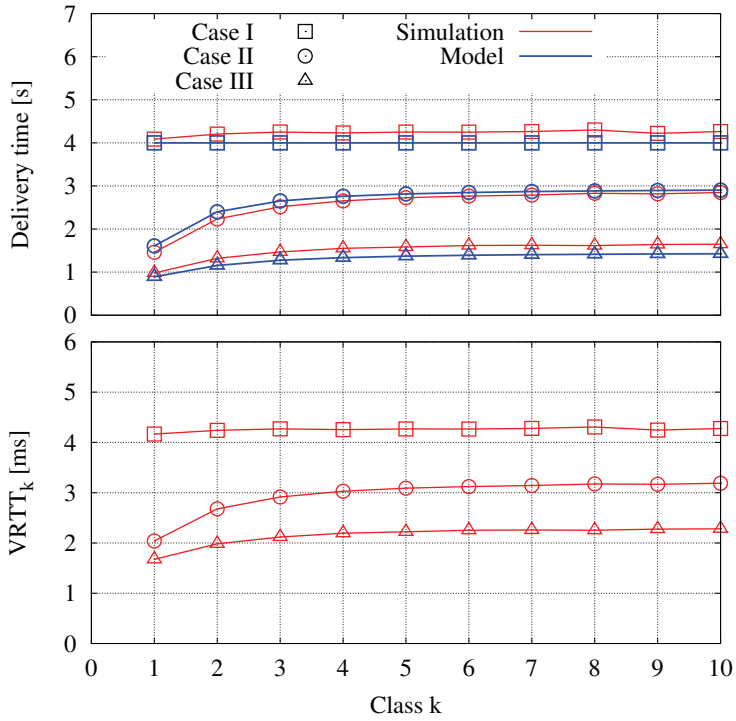


FIGURE E.9 – Cas I,II,III avec τ variable cas I : $C_1 = 50$Mbps, $C_2 = C_3 = 100$Mbps ; cas II : $C_1 = C_3 = 100$Mbps, $C_2 = 40$Mbps ; cas III : $C_1 = C_2 = 100$ Mbps, $C_3 = 50$Mbps

On considère des cas où plusieurs goulots d'étranglement sont présents ; en particulier, une ligne avec 3 noeuds (2 cache et 1 serveur) et trois liens. Fig.E.9 montre le temps de transfert

moyen et le $VRTT$ pour une classe de popularité donnée dans le cas d'un délai de propagation négligeable, $\sigma = 5000$ paquets comme taille moyenne des objets, cache LRU de 50000 paquets, requêtes des objet selon un processus de Poisson avec $\lambda = 1$ et popularité Zipf $\frac{1}{m}\frac{c}{k^\alpha}$ (où $m = M/K = 5$ et $\alpha = 1.7$). Les résultats des simulations sont aussi comparés avec les résultats analytiques obtenus dans la première partie et confirme que ICP est capable d'exploiter les ressources disponibles de façon efficace et équitable.

Contrôle de congestion noeud par noeud

Description du protocole

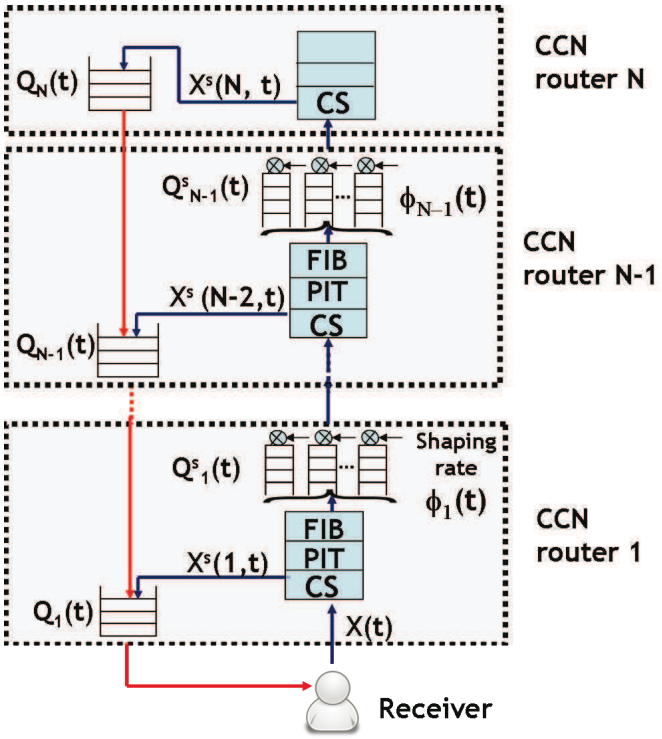


FIGURE E.10 – N hops network topology.

Cette section présente un mécanisme de contrôle de congestion noeud par noeud (HBH, Hop-by-Hop), qui réalise un contrôle du débit des requêtes par flux et par interface de sortie de chaque routeur CCN (comme illustré dans Fig.E.10.). La justification pour un contrôle de congestion noeud par noeud en plus de ICP (entièrement au récepteur) est double :

- anticiper la détection de la congestion,

- réduire le débit des requêtes par les noeuds intermédiaires pour une réaction à la congestion plus rapide.

Dans chaque interface de sortie, nous maintenons une file d'attente virtuelle par flux, identifiée par le nom du contenu. A chaque file d'attente virtuelle est associé un compteur de crédit initialisé à une valeur maximale B (en octets), indiquant le nombre d'octets de données que le flux peut transmettre, sans retard supplémentaire. Le compteur est incrémenté au débit équitable estimé pour le lien de retour des data, et diminué pour chaque envoi de requête de la taille de la DATA que l'INTEREST demande (il est nécessaire une information supplémentaire dans les paquets INTEREST). L'évolution de la file d'attente virtuelle et le compteur sont décrit par l'Algo.3.

Algorithm 3 Gestion de files d'attentes virtuelles f

Pour chaque réception d'INTEREST (interest, f)
$bottlenecked[f] = ((length(f) > 0) OR (counters[f] <= 0))$
if $(bottlenecked[f])$ **then**
    $shaping_queue_tail(interest, f)$
else
    $link.enqueue(interest, t = 0)$
    $update_rate(rate, interest.data_size)$
end if
$counter[f] = max(0, counter[f] - interest.data_size)$

A l'envoi d'un INTEREST()
    $interest = shaping_queue_head(current_q)$
    $reduced_rate = downlink_rate - rate$
    $link.enqueue(interest, t = interest.data_size/reduced_rate)$
    $current_q = next_q$

Le débit d'envoi des INTEREST au noeud i, au temps t $\phi_i(t)$, qui représente le débit équitable estimé pour la voie descendante du lien sur lequel les INTEREST son envoyés (les DATA suivent toujours le chemin inverse des INTEREST dans CCN), est :

$$\phi_i(t) = (C_{i+1} - \sum_{j>i, j \in \mathcal{R}_i^{nb}} X^s(j,t))/(N - \|\mathcal{R}_i^{nb}\|), \tag{E.20}$$

défini comme le débit maximale du lien C_{i+1}, moins le taux des flux qui ne sont pas étranglés (estimé à travers une fênetre coulissante de taille ΔT ms pour le calcul de la moyenne pondérée mobile) par le lien $i+1$ ($N - \mathcal{R}_i^{nb}$ est le nombre des flux étranglés par $i+1$ estimé avec le numéro des files d'attentes virtuelles pleines).

Analyse du système

L'analyse du protocole HBH utilisé en couple avec ICP (HBH seul n'est pas suffisant pour atteindre un partage équitable de la bande passante), est très similaire a celle faite pour ICP seul, avec la différence que le delai des files d'attente $Q_i(t)$ est remplacé par celui des files d'attente virtuelles $Q_i^S(t)$ qui memorisent les INTEREST. Pour cette raison, dans cette synthèse, nous ne décrivons pas les détails de l'analyse de HBH+ICP (Chapitre 6 de la thése).

Évaluation du système

Dans cette section, pour évaluer les performances du protocole HBH et souligner ses propriétés, on utilise le simulateur décrit dans le Chapitre 9 dans deux scénarios différents.

Scenario 1 *HBH seul n'est pas suffisant*
On considère un réseau ligne de deux noeuds, le téléchargement d'un fichier de 5Moctets, liens de (C_1, C_2)= $(100, 40)$ Moctets/s, délai de propagation $1ms$ et files d'attente virtuelles avec $B = 2$ koctets, ΔT = 100ms. Dans le tableau on montre les résultats pour le temps de transfert dans les cas avec fenêtre des requêtes de taille fixe ou réglée par ICP et de HBH active ou non. On peut observer par ces résultats que HBH permet de réduire le taux de perte de paquets mais aussi que le mécanisme n'est pas suffisant pour garantir l'utilisation efficace de la bande passante, car la valeur optimale de W n'est pas connue a priori et doit changer selon les ressources disponibles.

W	Temps de transfert[s]		Débit[Moctets/s]		Pertes[%]	
	w HbH	w/o	w	w/o	w	w/o
2	2.42	2.42	16.30	16.30	0	0
10	1.00	1.00	39.70	39.60	0	0
15	1.00	2.08	39.60	19.20	0	11.20
20	1.00	1.90	39.60	20.90	0	15.30
ICP	1.00	1.00	39.80	39.80	0	0

TABLE E.3 – Scénario 1. ICP - fenêtre des requêtes de taille fixe.

Scénario 2 *Les avantages du mécanisme HBH*
Avec les même paramètres utilisés pour le scénario 1, on considère 2 flux ICP (le premier démarre a $t = 0$s, le second à $t = 0.5$s) et un flux à débit constant (égal à 40 Moctets/s qui démarre a 1s). Dans Fig.E.11, on peut observer comme les files d'attente pleines sont celles en voie montante ("uplink") qui mémorisent les INTEREST et que le mécanisme HBH permet de protéger les flux ICP en présence d'un (ou plusieurs) flux très agressif(s).

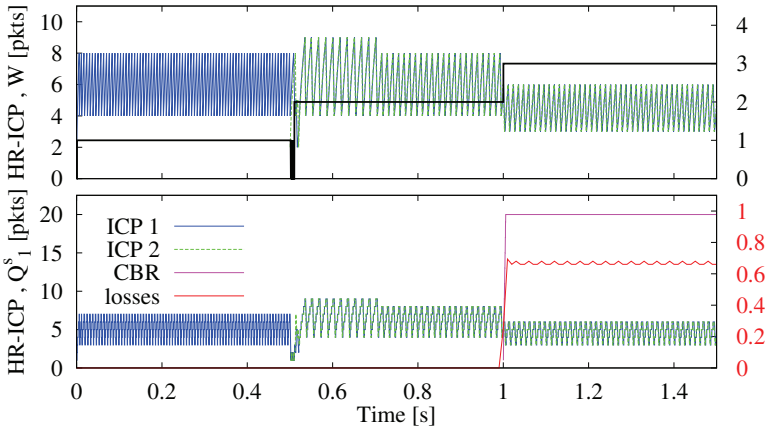


FIGURE E.11 – Scenario 2. ICP with HBH interest shaping.

Protocole de transport multi-chemin et mécanisme d'acheminement des requêtes

Mécanisme de contrôle de congestion

Le protocole de transport ICP, est basée sur un mécanisme AIMD qui mesure l'évolution du délai aller-retour pour détecter la congestion. Les simulations que nous avons menées mettent en évidence deux limites d'une telle approche :

1. la diminution déterministe de la fenêtre des requêtes est très sensible aux erreurs d'estimation ;

2. le seuil τ est très difficile à régler, surtout en présence de trajets multiples et de délai aller-retour hétérogènes.

Pour ces raisons, nous introduisons un mécanisme de diminution probabiliste de la fenêtre W basé sur le mecanisme RAAQM, qui nous avons demontré être stable et efficace. La motivation derrière l'utilisation de ce mécanisme est d'anticiper la diminution fenêtre tant qu'une variation du retard aller-retour est détectée, et de réaliser une décroissance moins aggressive de la fenêtre.

Contrôle des requêtes AIMD avec Remote Adaptive Active Queue Management (RAAQM) La fenêtre des requêtes (ou de congestion) W, est maintenue au niveau du récepteur et définit, comme pour ICP, le nombre maximum d'INTEREST que le récepteur est autorisé à envoyer. La fenêtre est augmentée de η/W à la réception d'un paquet (augmentation de η pour chaque fenêtre des requêtes satisfaites).

Quand un paquets DATA est reçu, le récepteur mesure le délai d'aller-retour instantané et estime le délai minimum et maximum ($R_{\min}$ et $R_{\max}$) sur une histoire de plusieurs échantillons (30 paquets par défaut). Comme pour ICP, une fenêtre complète d'échantillons est nécessaire avant de déclencher la première diminution de la fenêtre. Quand un paquet est reçu, une diminution de la fenêtre (W est multiplié par $\beta < 1$) est effectuée selon une probabilité croissante avec le délai du paquet même. La mesure instantanée du retard aller-retour $R(t)$ et les estimations de $R_{\min}$ et $R_{\max}$ concourent donc à déterminer la probabilité p d'une diminution de la fenêtre au fil du temps. L'évolution de p est représentée par :

$$p(t) = p_{\min} + \Delta p_{\max} \frac{R(t) - R_{\min}(t)}{R_{\max}(t) - R_{\min}(t)} \tag{E.21}$$

où $\Delta p_{\max} = p_{\max} - p_{\min}$. Si $R_{\min} = R_{\max}$ dans une fenêtre d'échantillons, alors $p(t) = p_{\min}$.

En analogie avec la représentation fluide de la dynamique d'ICP, l'évolution de la fenêtre peut être décrite par l'équations différentielle suivante (plus détaillé dans le Chapitre 7) :

$$\dot{W}(t) = \frac{\eta}{R(t)} - \beta W(t) p(t - R(t)) W(t - R(t)) \tag{E.22}$$

L'extension de l'algorithme décrit au cas multi-chemins demande quelques précisions. En particulier, nous utilisons :

- *Une fenêtre commune W* : on utilise une seule fenêtre des requêtes, au lieu d'une fenêtre par route, pour pouvoir passer à l'échelle ;
- *Etiquetage des routes* : chaque paquet de données doit porter une étiquette pour identifier la route utilisée (composée par exemple par la séquence d'identifiants de noeuds traversés) ;
- *Mesures par route* : le récepteur doit maintenir des estimations du minimum et du délai maximum d'aller-retour pour chaque route séparément en distinguant les échantillons selon leur étiquette.
- *Diminution de W probabiliste* : sur la base des estimations du délai de route, une route à une probabilité de diminution de la fenêtre p_r.

Si l'on note avec $\mathcal{R}$ l'ensemble des routes, l'équation de l'évolution de la fenêtre devient :

$$\dot{W}(t) = \frac{\eta}{R(t)} - \beta W(t) \sum_{r \in \mathcal{R}} p_r(t - R(t)) s_r W(t - R(t)), \tag{E.23}$$

Acheminement des requetes

L'acheminement optimal des demandes exigerait une connaissance globale du réseau. Dans la pratique, cela n'est pas possible à cause de la taille du réseau. Dans cette section, nous nous concentrons sur un algorithme distribué pour la sélection de l'interface de sortie d'un INTEREST (parmi celles selectionnées à travers une recherche dans le FIB), qui prend une décision à travers des mesures de performance par préfixe et par interface de sortie à chaque noeud. Cette métrique de performance doit représenter :

1) la proximité du contenu, c'est à dire la probabilité de trouver le contenu cherché prés de l'interface de sortie ;

2) état de congestion de l'interface.

Nous affirmons qu'une mesure indirecte est déjà disponible à chaque noeud dans l'architecture CCN. Cette mesure est représentée dans la PIT avec le numéro d'INTEREST en attente d'une réponse (PI - Pending Interest) divisé par interface et par préfixe. Une telle mesure peut être directement extraite à partir des informations dejà stockées dans la PIT et introduite dans la FIB pour aider à la décision de l'interface plus performante.

Les détails de l'algorithme sont décrits dans l'algorithme 4. A chaque envoi de paquet d'INTEREST ou réception de paquet DATA, le nombre de PI associé à une entrée FIB (préfixe) et à une interface de sortie particulière est mise à jour (augmenté pour l'envoi d'un INTEREST / diminué pour la réception d'un DATA). La valeur de PI est une moyenne pondérée (avec un paramètre $0,1$ pour pondérer le nouvel échantillon) est régulièrement mis à jour pour calculer les poids des interfaces. Un algorithme aléatoire pondéré est utilisé pour sélectionner l'interface de sortie pour chaque INTEREST où les poids sont utilisés pour favoriser des interfaces plus performantes. Au début, chaque interface a le même poids et le choix est donc complètement aléatoire.

Résultats de simulation

Pour vérifier les performances des algorithmes proposés, on utilise le simulateur CCNPLsim décrit dans le Chapitre 9. En particulier dans le réseau (sans cache) illustré dans Fig.E.12 20 flux téléchargent des objets stockés dans les 4 différents serveurs en utilisant tous les quatre des chemins en parallèle. Fig.E.13(a), E.13(b) montrent que notre algorithme de transport est capable d'exploiter efficacement la bande passante disponible et que l'algorithme d'acheminement converge aux poids optimaux (calculés à la main pour ce petit réseau).

Algorithm 4 Algorithme d'acheminement des INTEREST

```
1:  A la réception d'un DATA(name,face_in)
2:  if ( !PIT_miss) then
3:      Forward_Data(face_out);
4:      FIB_PI_Decrease(face_in, prefix);
5:      FIB_Weight_Update(face_in, prefix);
6:  else Drop_Data;
7:  end if
8:
9:  A la réception d'un INTEREST(name, face_in)
10: if (CACHE_miss && PIT_miss) then
11:     (prefix,weight_list)=FIB_Lookup(name);
12:     f*=RND_Weight(weight_list);
13:     FWD_I(f*);
14:     PIT_I_Update(name,f*);
15:     FIB_PI_Incr(f*, prefix);
16:     FIB_Weight_Update(f*, prefix);
17: else CCN_standard_processing;
18: end if
19:
20: A l'expiration d'un temporisateur PIT(name, prefix, face)
21:    FIB_PI_Decr(face, prefix);
22:    FIB_Weight_Update(face, prefix);
23:
24: procedure FIB_WEIGHT_UPDATE(face, prefix)
25:  avg_PI(face, prefix) ← α · avg_PI + (1 − α) PI(face, prefix);
26:  w(face, prefix) ← 1/avg_PI(face, prefix);
27: end procedure
28:
29: procedure FIB_PI_INCR(face, prefix)
30:     PI(face, prefix) + +;
31: end procedure
32:
33: procedure FIB_PI_DECR(face, prefix)
34:     PI(face, prefix) − −;
35: end procedure
```

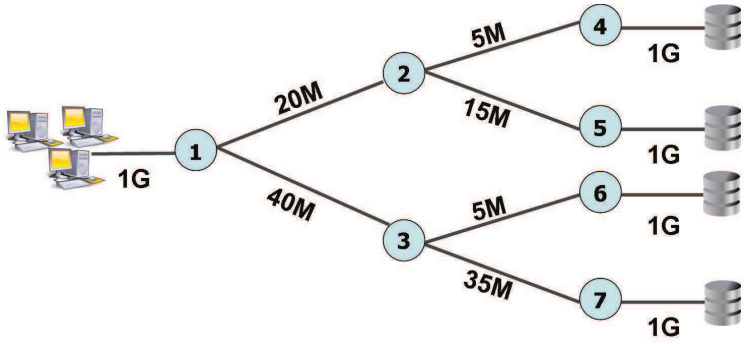
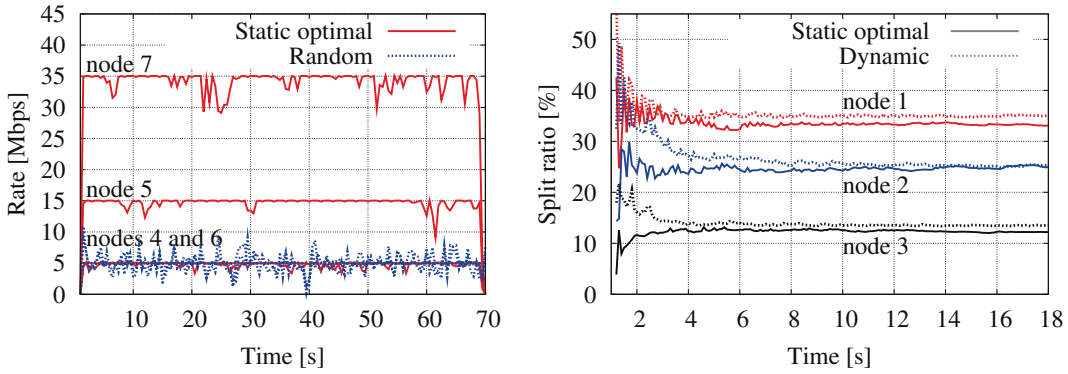


FIGURE E.12 – Scénario multi-chemin.



(a) (b)

FIGURE E.13 – Scénario multi-chemin : débit des liens (a), pourcentage d'envoi des INTEREST par interface (b).

Conclusion

Avec l'introduction du World Wide Web, Internet a changé son modèle de communication en passant d'un modèle hôte à hôte a un modèle centré sur la livraison des contenus et l'accès aux services. L'architecture d'Internet a réussi à répondre aux exigences de nouvelles applications en déployant des services de diffusion de contenu au niveau applicatif, au-dessus d'IP (par exemple, HTTP [70]). En dépit de son succès, l'architecture actuelle d'Internet a été conçue afin de permettre la communication entre deux machines et non pour la récupération / diffusion des contenus.

Le déséquilibre évident entre l'architecture d'Internet et son utilisation, génère de nombreux problèmes en termes de sécurité, dépendance de l'emplacement du contenu, mobilité, etc. et motive les chercheurs à étudier des possibles évolutions pour Internet. Dans ce contexte, les propositions ICN essayent de résoudre les problèmes architecturaux en centrant Internet sur le noms des

contenus plutôt que sur les adresses des hôtes.

Contribution

Dans cette thèse, nous étudions les architectures ICN et en particulier CCN. Cette nouvelle architecture apporte des avantages considérables en termes de diffusion des contenus, sécurité, gestion des réseaux simplifiée, mobilité, etc. CCN a été proposé en 2009 dans[14] par Van Jacobson et al. au centre de recherche PARC (Palo Alto Research Center). La proposition est encore à ses débuts et de nombreuses contraintes doivent encore être prises en compte. Avec cette thèse nous abordons certaines questions que nous avons identifiées dans la proposition originale. En particulier, nous étudions la performance perçue par l'utilisateur et proposons des mécanismes de contrôle de congestion et d'acheminement des requêtes.

Dans la première partie de la thèse, nous analysons les performances CCN. Dans un réseaux traditionel, les performances de stockage et de partage de la bande passante sont étudiées séparément. Au contraire, dans CCN les caches sont directement intégrés au niveau réseau est influencent les performances de l'utilisateur. Afin de capturer le performance des caches, nous dérivons des formules fermées pour les politiques de remplacement Least Recently Used (LRU) et aléatoire (RND) en négligeant la bande passante. Les deux politiques de remplacement sont analysées avec différentes hypothèses. Les performances du cache LRU, à la différence de celui RND, prends en compte les paquets afin de mieux comprendre l'architecture CCN. Les deux modèles prennent en compte des réseaux de caches, différemment des autres modèles proposés dans la littérature. La comparaison entre LRU et RND montre également que les performances des deux politiques de remplacement sont trés similaires (avec un avantage pour LRU) et peuvent être utilisées alternativement dans le reséau (RND demande moins de resources et peut étre utilisé avec un trés grand débit de requêtes. Dans un deuxième temps, nous prenons aussi en compte les contraintes de bande passante. Nous proposons des formules mathématiques pour l'analyse de la performance perçue par l'utilisateur, qui capturent l'interaction entre le stockage et la bande passante de l'architecture CCN. Même si les modéles proposés sont adaptés à l'architecture CCN, ils peuvent être étendus à d'autres propositions ICN avec des caractéristiques similaires (par example le transport orienté au récepteur et dans les cache au niveau réseau.

Pour tester la validité des formules proposées, nous avons développeé un simulateur dédié CCNPL-Sim (CCN Packet Level Simulator). CCNPL-Sim est construit sur un simulateur existant, CBCBSim [69] auquel on a ajouté les files d'attente, le protocol de transport, le cache etc. CCNPL-Sim est publié sous la licence GNU GPL v2 et est disponible pour téléchargement à l'addresse `http://code.google.com/p/ccnpl-sim/`.

Dans la deuxième partie de la thèse, nous nous concentrons sur la conceptionn des protocoles

réseau pour CCN. Nous exploitons à cette fin les résultats de l'analyse des performances comme réference pour le design des protocoles. Nous proposons un contrôle de congestion et un mécanisme d'acheminement des requêtes que nous avons encore testé avec CCNPL-Sim. En ce qui concerne le contrôle de congestion, nous proposons un mécanisme de contrôle à fenêtre pour limiter l'envoi d'INTEREST (ICP - Interest Control Protocol). ICP utilise le mécanisme AIMD pour controler l'évolution de la taille de la fenêtre en utilisant la mesure de délais d'aller-retour des paquets comme indication de congestion. En utilisant la symétrie des chemins des INTERESTs et des DATAs dans CCN, nous proposons également un mécanisme de contrôle de congestion noeud par noeud ayant pour but d'accélérer la détection/reaction à la congestion. Nous étendons ensuite ICP afin d'exploiter efficacement les chemins multiples caractérisés par des performances hétérogènes en terme de délai. Dans cette optique, nous adoptons un mécanisme RAAQM (Remote Adaptive Active Queue Management) qui permet de contrôler une seule Fenêtre de congestion AIMD avec un diminution de la fenêtre probabiliste et par-chemin (chaque chemin est identifié par une étiquette). Nous proposons également un mécanisme d'acheminement des requêtes qui prends sa décision sur une mesure de la performance locale par interface et par préfixe. Nous comparons ensuite ce mécanisme avec la solution optimale en démontrant la qualité de notre algorithme.

Bibliography

[1] "The Global Environment for Network Innovations." (GENI) http://www.geni.net.

[2] "European Future Internet Assembly." (FIA) http://www.future-internet.eu.

[3] "Asia Future Internet forum." (FIA) http://www.asiafi.net/.

[4] Koponen, T. and Chawla, M. and Chun, B. and Ermolinskiy, A. and Kim, K.H. and Shenker, S. and Stoica, I., "A data-oriented (and beyond) network architecture," in *Proc. of ACM SIGCOMM*, 2007.

[5] "4WARD Project." http://www.4ward-project.eu/.

[6] "SAIL Project." http://www.sail-project.eu/.

[7] Farrell, S. and Dannewitz, C. and Ohlman, B. and Kutscher, D., "URIs for named information." IETF Internet-Draft draft-farrell-ni, March 2011.

[8] Kutscher, D. et al., "SAIL, Deliverable D.B.2: Content Delivery and Operations," 2012.

[9] "PSIRP Project." http://www.psirp.org/.

[10] "PURSUIT Project." http://www.fp7-pursuit.eu/.

[11] Trossen, D. et al., "PURSUIT, Deliverable D2.3: Architecture Definition, Components Descriptions and Requirements," 2011.

[12] Carzaniga, A. and Wolf, A. L., "Forwarding in a content-based network," in *Proc. of ACM SIGCOMM*, 2003.

[13] Carzaniga, A. and Rutherford, M. J. and Wolf, A. L., "A Routing Scheme for Content-Based Networking," in *Proc. of IEEE INFOCOM*, 2004.

[14] Jacobson, V. and Smetters, D.K. and Thornton, J.D. and Plass, M.F. and Briggs, N.H. and Braynard, R.L., "Networking named content," in *Proc. of ACM CoNEXT*, 2009.

[15] "NDN Project." `http://www.named-data.net/`.

[16] "CONNECT Project." `http://www.anr-connect.org/`.

[17] King, W.F., "Analysis of paging algorithms," in *Proc. of IEEE IFIP*, 1971.

[18] Flajolet, P. and Gardy, D. and Thimonier, L., "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Appl. Math.*, vol. 39, no. 3, pp. 207–229, 1992.

[19] Jelenković, P. R., "Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *The Annals of Applied Probability*, vol. 9, no. 2, pp. 430–464, 1999.

[20] Jelenković, P. R. and Kang, X., "Characterizing the miss sequence of the LRU cache," in *Proc. of ACM SIGMETRICS, MAMA Workshop*, 2008.

[21] Coffman, E.G. and Jelenković, P.R., "Performance of the move-to-front algorithm with Markov-modulated request sequences," *Oper. Res. Lett.*, pp. 109–118, 1999.

[22] Jelenković, P. R. and Radovanović, A., "Least-recently-used caching with dependent requests," *Theor. Comput. Sci.*, vol. 326, no. 1-3, pp. 293–327, 2004.

[23] Jelenković, P. R. and Radovanović, A., "Optimizing LRU Caching for Variable Document Sizes," *Comb. Probab. Comput.*, vol. 13, no. 4-5, pp. 627–643, 2004.

[24] T. Osogami, "A Fluid Limit for Cache Algorithms with General Request Processes," in *Proc. of IEEE INFOCOM*, 2009.

[25] Gelenbe, E., "A Unified Approach to the Evaluation of a Class of Replacement Algorithms," *IEEE Transactions on Computer*, vol. 22, no. 6, pp. 611–618, 1973.

[26] Elisha J. Rosensweig and Jim Kurose and Don Towsley, "Approximate Models for General Cache Networks," in *Proc. of IEEE INFOCOM*, 2010.

[27] Dan, A. and Towsley, D., "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proc of. ACM SIGMETRICS*, 1990.

[28] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, pp. 609–634, 2006.

[29] O. Babaoğlu, "Hierarchical replacement decisions in hierarchical stores," in *Proc. of ACM SIGMETRICS*, 1982.

[30] Che, H. and Tung, Y. and Wang, Z., "Hierarchical Web caching systems: modeling, design and experimental results," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 7, pp. 1305 – 1314, 2002.

[31] Laoutaris, N. and Che, Hao and Stavrakakis, I., "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, pp. 609–634, 2006.

[32] Fricker,C. and Robert, P. and Roberts, J. and Sbihi,N.., "Impact of traffic mix on caching performance in a content-centric network," in *Proc. of IEEE INFOCOM NOMEN Workshop*, 2012.

[33] Stern, T. E. and Elwalid, A. I., "Analysis of Separable Markov-Modulated Rate Models for Information-Handling Systems," *Advances in Applied Probability*, vol. 23, no. 1, pp. 105–139, 1991.

[34] Edward C. and Jordy B., "Nonstationary Poisson modeling of web browsing session arrivals," *Information Processing Letters*, vol. 102, no. 5, pp. 187 – 190, 2007.

[35] Cha, M. and Kwak, H. and Rodriguez, P. and Ahn, Y. and Moon, S., "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proc. ACM SIGCOMM IMC*, 2007.

[36] Mitra, S. and Agrawal, M. and Yadav, A. and Carlsson, N. and Eager, D. and Mahanti, A., "Characterizing Web-Based Video Sharing Workloads," *ACM Transactions on the Web*, vol. 5, pp. 8:1–8:27, May 2011.

[37] Costa, C. P. and Cunha, I. S. and Borges, A. and Ramos, C. V. and Rocha, M. M. and Almeida, J. M. and Ribeiro-Neto, B., "Analyzing client interactivity in streaming media," in *Proc. of ACM WWW*, 2004.

[38] Abramowitz, M. and Stegun, I., *Handbook of Mathematical Functions with formulas, graphs and mathematical tables*. Dover, 10th ed., 1972.

[39] Massoulié, L. and Roberts, J., "Bandwidth sharing: objectives and algorithms," *IEEE/ACM Transactions on Networking*, 2002.

[40] Fred, S. Ben and Bonald, T. and Proutiere, A. and Régnié, G. and Roberts, J. W., "Statistical bandwidth sharing: a study of congestion at flow level," in *Prof. ACM SIGCOMM*, 2001.

[41] Kortebi, A. and Muscariello, L. and Oueslati, S. and Roberts, J., "Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing," in *Prof. of ACM SIGMETRICS*, 2005.

[42] Shreedhar, M. and Varghese, G., "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, 1996.

[43] Arianfar, S. and Nikander, P. and Eggert, L. and Ott, J., "A Transport Protocol for Content Centric Networks," in *Proc. of IEEE ICNP, Poster session.*, 2010.

[44] Jelenković, P. R. and Kang, X., "LRU Caching with Moderately Heavy Request Distributions," in *Proc. of Analytic Algorithmics and Combinatorics*, 2007.

[45] Sinha, P. and Nandagopal, T. and Venkitaraman, N. and Sivakumar, R. and Bharghavan, V., "WTCP: a reliable transport protocol for wireless wide-area networks," *Wireless Networks*, 2002.

[46] Floyd, S. and Handley, M. and Padhye, J. and Widmer, J., "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM* , 2000.

[47] Tsaoussidis, V. and Zhang, C., "TCP-Real: receiver-oriented congestion control," *Elsevier Computer Networks*, 2002.

[48] Gupta, R. and Chen, M. and Mccanne, S. and Walrand, J., "WebTP: A Receiver-Driven Web Transport Protocol," in *Proc. of IEEE INFOCOM* , 1999.

[49] Hsieh, H.-Y. and Kim, K.-H. and Zhu, Y. and Sivakumar, R., "A Receiver-Centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces," in *Proc. of ACM MOBICOM*, 2003.

[50] Jung-Shan C. and Sheng-Ching L. and Ray-I C. and Chia-Hui W., "Priority-based R2CP for multipoint-to-point video streaming," in *IEEE International Conference on Computer and Information Technology*, 2008.

[51] Kuzmanovic A. and Knightly E.W., "Receiver-centric congestion control with a misbehaving receiver: Vulnerabilities and end-point solutions," *Computer Networks*, 2007.

[52] Papadimitriou, P. and Tsaoussidis, V. and Mamatas, L., "A receiver-centric rate control scheme for layered video streams in the Internet," *Systems and Software*, 2008.

[53] Kuzmanovic, A. and Knightly, E.W., "TCP-LP: low-priority service via end-point congestion control," *IEEE/ACM Transactions on Networking*, 2006.

[54] M. Ajmone Marsan, M. Garetto, P. Giaccone, E. Leonardi, E. Schiattarella, and A. Tarello, "Using partial differential equations to model TCP mice and elephants in large IP networks," in *IEEE INFOCOM*, 2004.

[55] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. of ACM SIGCOMM Workshop on ICN*, 2011.

[56] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks," *ACM SIGCOMM CCR.*, pp. 101–114, 1994.

[57] H. T. Kung and K. Chang, "Receiver-oriented adaptive buffer allocation in credit-based flow control for ATM networks," in *Proc. of IEEE INFOCOM*, 1995.

[58] P. P. Mishra, H. Kanakia, and S. K. Tripathi, "On hop-by-hop rate-based congestion control," *IEEE/ACM Transactions on Networking*, 1996.

[59] S. Fdida and N. Rozhnova, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *Proc. of IEEE INFOCOM NOMEN Workshop*, 2012.

[60] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware traffic control for a content-centric network," in *Proc. of IEEE INFOCOM*, 2012.

[61] Carzaniga, A. et al., "CCNx." `http://www.ccnx.rg`.

[62] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the Internet," *IEEE/ACM Transaction on Networking Networking*, 2006.

[63] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *SIGCOMM Computer Communication Review*, 2005.

[64] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of Usenix ATEC*, 2004.

[65] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. of Usenix NSDI*, 2011.

[66] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck," in *Proc. of PFLDNeT Workshop*, 2009.

[67] C. Hollot and Y. Chait, "Nonlinear stability analysis for a class of tcp/aqm networks," in *Proc. of IEEE Conference on Decision and Control*, 2001.

[68] D. Ardelean, E. Blanton, and M. Martynov, "Remote active queue management," in *Proc. of ACM NOSSDAV*, 2008.

[69] Carzaniga, A. et al., "CBCBSim." `http://www.inf.usi.ch/carzaniga/cbn/routing/index.html#cbcbsim`.

[70] L. Popa, A. Ghodsi, and I. Stoica, "HTTP as the narrow waist of the future internet," in *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[71] M. M. Rao and R. J. Swift, *Probability Theory with Applications*. Springer, 2nd ed., 2006.

[72] N. R. Chaganty and J. Sethuraman, "Multidimensional large deviations local limit theorems," *Journal of Multivariate analysis*, vol. 20, pp. 190–204, 1986.

[73] F. Verhulst, *Introduction to Functional Differential Equations*. New York, NY, USA: Springer-Verlag New York, Inc., 1993.

[74] J. K. Hale and S. M. Lunel, *Nonlinear differential equations and dynamical systems*. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

List of Publications and Submissions

[75] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Proc. of ITC23*, 2011.

[76] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," in *Proc. of ACM SIGMETRICS 2012 (poster session), full version at. http://arxiv.org/abs/1202.4880*, 2012.

[77] G. Carofiglio, M. Gallo, and L. Muscariello, "Bandwidth and storage sharing performance in information-centric networking," in *Proc. of ACM SIGCOMM ICN*, 2011.

[78] G. Carofiglio, M. Gallo, and L. Muscariello, "On the Performance of Bandwidth and Storage Sharing in Information-Centric Networks." Submitted to Elsevier Computer Networks, 2012.

[79] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance Evaluation of the Random Replacement Policy for Networks of Caches." Submitted to Performance Evaluation, 2012.

[80] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking," in *Proc. of IEEE INFOCOM NOMEN*, 2012.

[81] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint Hop-by-hop and Receiver-Driven Interest Control Protocol for Content-Centric Networks," in *Proc. of ACM SIGCOMM ICN*, 2012, awarded as best paper.

[82] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Evaluating per-application storage management in content-centric networks." Submitted to Elsevier Computer Communication - Special Issue on Information-Centric Networking, 2012.