Cache, process and forward in Information-centric networking

Leonce Mekinda Mengue

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-0075

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité “Informatique et Réseaux”

présentée et soutenue publiquement par

Léonce MEKINDA

le 01/12/2016

Cache, Process and Forward in Information-Centric Networking

Mécanismes de Cache, Traitement et Diffusion dans les Réseaux Centrés sur l’Information

Jury

M. Emilio LEONARDI, Professeur, *Politecnico di Torino*	Rapporteur
M. Fabio MARTIGNON, Professeur, *Université Paris-Sud*	Rapporteur
Mme Neiva LINDQVIST, Senior Researcher, *Ericsson*	Examinateur
M. James ROBERTS, Senior Researcher	Examinateur
M. Dario ROSSI, Professeur, *TELECOM ParisTech*	Examinateur
M. Alain SIMONIAN, Research Engineer, *Orange Labs*	Examinateur
M. Thomas BONALD, Professeur, *TELECOM ParisTech*	Directeur de thèse
M. Luca MUSCARIELLO, Principal Engineer, *Cisco Systems*	Co-directeur de thèse

TELECOM ParisTech

Ecole de l’Institut Mines-Télécom - Membre de ParisTech

46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

T H È S E

Acknowledgement / Remerciements

Thank you Prof. Emilio Leonardi and Prof. Fabio Martignon to have consented to evaluating this manuscript. I am deeply honoured by the participation of Dr. Neiva Lindqvist, Dr. James Roberts, Prof. Dario Rossi and Dr. Alain Simonian in my Ph.D. evaluation committee.

I would thank Luca, who offered me this unique opportunity to fulfill a lifetime accomplishment, believed in me and inspired me along this journey. Thanks to Thomas, who honoured a young scientist in accepting to relentlessly oversee his humble work. I also hereby praise Giovanna's brilliance and precious insights.

Deep thanks to Philippe Olivier who kindly reviewed this manuscript. Nabil, Eric, Christian, Mustapha, Adam, Nancy, Yannick, Felipe, Yassine, Deborah, Claudio, Bruno, Orange TRM team as a whole and beyond, I am definitely indebted to your kindness. To all LINCS researchers go my immense admiration and gratitude.

To my loving family and Him for all.

Abstract

This thesis investigates how making content caching and forwarding latency-aware can improve data delivery performance in Information-Centric Networks (ICN). We introduce a new very effective contribution to the existing content caching toolset. The designed mechanism leverages retrieval time observations to decide whether to store an object in a network cache, based on the expected delivery time improvement. We demonstrate that our distributed latency-aware caching mechanism, LAC+, outperforms state of the art proposals and results in a reduction of the content mean delivery time and standard deviation of LRU caches by up to 60%, along with a fast convergence to these figures.

In a second phase, we conjointly optimize the caching function and the multipath request forwarding strategies as both coexist in ICN at network level and should reinforce each other. To this purpose, we introduce the mixed forwarding strategy LB-Perf, directing the most popular content towards the same next hops to foster egress caches convergence, while load-balancing the others.

Third, we address ICN fairness to contents. We show that traditional ICN caching, which favors the most popular objects, does not prevent the network from being globally fair, content-wise. The incidence of our findings comforts the ICN community momentum to improve LFU cache management policy and its approximations. We demonstrate that in-network caching leads to content-wise fair network capacity sharing as long as bandwidth sharing is content-wise fair.

Finally, we contribute to the research effort aiming to help ICN Forwarding Information Base scale when confronted to the huge IoT era's namespace. We propose AFFORD, a novel view on routing in named-data networking that combines machine learning and stochastic forwarding. More specifically, we show that compressing the Forwarding Information Base into bitwise trie-indexed Artificial Neural Networks accelerates next hop lookup and reduces Forwarding Information Base's size by orders of magnitude.

Keywords: Information-Centric Networks; Performance evaluation; Cache management; Transport protocol; Forwarding; Machine learning.

Résumé

Ce travail de thèse s'est tout d'abord attaché à comprendre comment la prise en compte du temps de téléchargement, autrement dit, de la latence, lors de la mise en cache ou de la transmission de données pouvait contribuer aux performances du téléchargement dans les réseaux de caches dont ICN. Nous y introduisons un mécanisme distribué novateur qui décide de l'opportunité de conserver un objet en considérant que plus il a été long à télécharger plus intéressant il semble de le soumettre au cache sous-jacent. Nous montrons que ce nouveau mécanisme surpasse en de nombreux points l'état de l'art, que ce soit du point de vue de la réduction du temps moyen de téléchargement à partir de caches LRU, et de son écart-type (jusqu'à -60%), que de celui de la vitesse de convergence vers ceux-ci. La grande simplicité dudit mécanisme pourrait permettre son intégration immédiate aux sein d'architectures réseaux existantes car ne nécessitant aucun amendement de protocole.

Dans une seconde phase, nous avons optimisé conjointement les fonctions de mises en cache et de distribution multi-chemin de requêtes de contenus, qui d'ailleurs coexistent dorénavant au sein de la couche réseau en ICN, afin qu'elles se renforcent mutuellement.

Troisièmement, nous avons étudié l'équité vis-à-vis des contenus au sein des réseaux de caches et plus particulièrement, d'ICN. Il en ressort qu'ICN tel quel préfigure un réseau α-équitable malgré le recours extensif à des algorithmes de gestion de caches qui favorisent les contenus les plus populaires. Seule suffit une allocation équitable de la bande passante entre les contenus pour que l'équité d'ICN soit complète.

Notre dernière contribution vise à aider au passage à l'échelle d'ICN dans contexte où deviennent réalités l'Internet des Objets et son espace de nommage illimité. Nous avons proposé une approche nouvelle au routage dans les réseaux centrés sur l'information, nommée AFFORD, qui combine apprentissage automatique et diffusion aléatoire. Nous montrons que comprimer la base d'information de transmission (FIB) ICN, en la segmentant en plusieurs réseaux de neurones indexés par un arbre préfixe, accélère la recherche de nom et réduit la taille de la structure FIB de plusieurs ordres de grandeur.

Mots-clés : Réseaux Centrés sur l'Information ; Evaluation de performances ; Gestion de caches ; Protocoles de transport ; Distribution multi-chemin ; Apprentissage automatique.

Contents

Publications

The content of this thesis was published in two workshop [Carofiglio et al., 2015a] [Carofiglio et al., 2015b], two conference [Carofiglio et al., 2015c] [Mekinda and Muscariello, 2016] proceedings and a journal [Carofiglio et al., 2016]. A chapter was submitted to a conference [Bonald et al., 2017]. All are peer-reviewed international venues.

List of publications

Bonald, T., Mekinda, L., and Muscariello, L. (2017). On the Fairness of Information-Centric Networking. In *36th Annual IEEE International Conference on Computer Communications*, IEEE INFOCOM 2017 (Under review). San Francisco, CA, USA.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015a). Analysis of latency-aware caching strategies in information-centric networking. In *ACM SIGCOMM CoNEXT Workshop on Content Caching and Delivery in Wireless Networks*. Heidelberg, Germany.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015b). FOCAL: Forwarding and caching with latency awareness in information-centric networking. In *IEEE Globecom 2015 Workshop on Information Centric Network Solutions for Real-World Applications*, IEEE GLOBECOM 2015 ICNS. San Diego, CA, USA.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015c). LAC: Introducing latency-aware caching in information-centric networks. In *IEEE 40th Conference on Local Computer Networks*, IEEE LCN 2015, pages 422–425. Clearwater Beach, FL, USA.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2016). Joint Forwarding and Caching in Information-Centric Networking. *Computer Networks*.

Mekinda, L. and Muscariello, L. (2016). Supervised Machine Learning-based Routing for Named Data Networking. In *IEEE GLOBECOM 2016*. Washington, DC, USA.

Chapter 1

Introduction

Contents

Information-Centric Networking (ICN) [Jacobson et al., 2009b] is an emerging network communication paradigm intended to alleviate the unsustainable growth of the current Internet load. As key traits, *(i)* it replaces inside the network layer the "where" with the "what", in achieving routing and forwarding on content names; *(ii)* it uses extensively in-network caching. While less and less of the IP traffic remains cacheable due to end-to-end communication channel encryption [Naylor et al., 2014], ICN through, for example, its NDN [Zhang et al., 2014] implementation offers a promising alternative in ignoring user identity (no host address) and authenticating content provider from data packets (digital signature). Therefore, every content item can remain cacheable without sacrificing user privacy and content verification. This is important because inserting cache memories across the communication data path between different processing elements has been demonstrated over time to be a reliable way of improving network performance by localizing - especially popular - content at network edge and so reducing link load and retrieval latency.

Such a latency minimization, or building for virtual zero latency as commonly referred to, is one of the pillars of 5G network architecture design and is currently fostering research work in this space.

This thesis is a contribution to this effort. It tackles various aspects of NDN: *(i)* we propose, analyze and implement two latency-aware caching algorithm LAC and LAC+ that significantly shrink content delivery time; *(ii)* we propose, analyze and implement a joint latency-aware caching and forwarding algorithm, FOCAL, aiming at the performance enhancement of caching algorithms; *(iii)* we analyze the fairness of ICN, given the caching ubiquity it entails; *(iv)* and we propose an artificial neural network-based forwarding approach to help NDN's FIB scale in a global Internet namespace.

The remaining of the chapter will present in more details the Internet as we know it, with its weaknesses; What are the additional challenges 5G introduces; Where ICN can help; Then comes a statement of the problems this thesis addresses, an overview of our contributions and finally a few mathematical prerequisites for understanding the modeling part of our work.

1.1 Today's Internet

1.1.1 An architecture that no longer fits its usage

The Internet basically interconnects machines to transfer data in a client-server communication mode. For this, it relies on the TCP/IP protocol stack, which provides machine addressing and communication primitives. While this architecture has been successful in the earlier Internet stages, for remote access (Telnet, SSH), file transfer (FTP), Internet Relay Chat (IRC) sending and fetching e-mails (SMTP, POP3, IMAP4 protocol) or browsing the textual World Wide Web (HTTP) etc., it no longer fits the Internetwork actual usage. [Cisco, 2016] reports that the Internet is getting overwhelmingly used for multimedia content browsing or video downloading/streaming nowadays (more than 80% of the total traffic), making the host serving content by far less relevant than content itself.

1.1.2 The rampant threat of host-centric Internet collapse

Very few hosts and edge networks would afford the thousands of parallel flows and the resulting congestion most viral videos generate. Moreover, since [Floyd and Fall, 1999] urged to curate TCP-unfriendly, unresponsive or bandwidth greedy flows, no definitive solution has emerged to durably protect the host-centric Internet from congestion collapse in case of competing congestion control algorithms, best-effort protocols or misbehaving senders and receivers [Papadimitriou et al., 2011; Habib et al., 2016]. This weakness has even been the ground for an effective means of cyberwarfare known as Distributed Denial of Service (DDoS) attacks [Ben-Porat et al., 2013]. They is a worrying increase of such attacks, predicted to reach 10^7 a year by 2017. According to [Cisco, 2016], modern DDoS generates in average ~ 1 Gbps of malicious traffic very few organizations can resist.

1.1.3 The dawn of HTTPS-by-default

A host-centric Internet impedes massive *multicast delivery* over reliable transport, as a consequence of resource limitation at content servers but not only. Transport Layer Security (TLS)'s point-to-point cryptosystem prevents secure sessions from being replicated and diffused from the network edge. It is typically a barrier experienced by any ISP who tries to scale-down some YouTube traffic. Actually, half the World Wide Web traffic is

carried over SSL/TLS and major Internet actors like Google, Netflix and Facebook have defaulted to HTTPS [Naylor et al., 2014]. The same paper emphasizes that HTTPS-as-unique-vector nullifies value-added services like in-network caching, thus would yield a minimum 15% traffic increase if a solution is not found. [Paschos et al., 2016] pertinently objected that the current use of content provider representatives such as caching black-boxes to handle encryption matters within operators networks might not be satisfactory. The example of Google Global Cache is illustrative. They leave operators helpless in their legitimate intend to conduct their own network optimization.

Multi-context TLS (mcTLS) [Naylor et al., 2015] has recently been proposed to reconcile TLS with middleboxes, in a way some may not agree with. mcTLS security protocol grants trusted middleboxes access to the traffic, unencrypted. Not prone to that obvious privacy exposure is [Sherry et al., 2015]'s BlindBox, which enables Deep Packet Inspection (DPI) over encrypted traffic. Blindbox HTTPS scans encrypted traffic for encrypted keywords. For this, during a rather slow HTTPS connection setup, it encrypts its ruleset without disclosing neither the rules to the endpoints (thanks to oblivious transfer [Naor and Pinkas, 1999]) nor the private key shared by the communicating endpoints to the middlebox (thanks to Yao garbled circuits [Yao, 1986]). Its slow minute-scale initialization restricts the scope of the protocol to persistent secure tunnels. More important for our study, since BlindBox HTTPS does not provide means for translating the encrypted payload from one secure session to another, it does not enable caching.

1.1.4 Excessive latency

In an extensive survey, [Briscoe et al., 2014] reviewed the roots of Internet latency in a broad sense. They identified (i) structural delays like those caused by sub-optimal routes or name resolution; (ii) processing delays generated by protocol stacks or operating systems; (iii) physical/link layer-related delays due to medium acquisition, signal propagation, scheduling, queuing, bufferbloats; (iv) those resulting from network/transport protocol operations like NAT setup, connection initialization and Head-of-Line locking occurring during packet loss recovery. A few remedies still need to convince a majority of Internet stakeholders.

Standard TCP algorithms like New Reno and Cubic are prone to unnecessary delay. These algorithms only react to packet loss, which often occurs at route bottlenecks due

to tail drops, *i.e.*, when full transmission buffers start dropping incoming packets. Although a wider adoption of Active Queue Management (AQM) and Explicit Congestion Notification (ECN) would have prevented packet drops and subsequent retransmissions, TCP flavours exploiting these techniques, DataCenter TCP (DCTCP) for example, remain underemployed.

Also, TCP Fast Open (TFO) [Radhakrishnan et al., 2011] offers to save a Round Trip Time (RTT) in relaxing the three-way handshake by conveying data to applications in an opening SYN packet, and mitigating security risks with a cryptographic cookie. However, it cannot detect duplicated connections since a TFO cookie encloses the client's IP address the server encrypted and sent during a precedent full handshake.

Quick UDP Internet Connections (QUIC) [Hamilton et al., 2016] seems to gather insights from all previous attempts in turning protocol bootstraps into 0-RTT handshakes by means of a source-address token similar to TFO's cookie. QUIC conveys Forward Error Correction (FEC) packets to avoid as much as possible the retransmission of lost packets. Congestion control may consist of CUBIC (by default) or TCP-Reno or Packet pacing. A few works have demonstrated that the expected content delivery time improvements only show up in very lossy networks [Carlucci et al., 2015; Megyesi et al., 2016]. Moreover, [Lychev et al., 2015] highlighted QUIC's fundamental vulnerability to rather simple DoS attacks exploiting source-address token replay, or off-path byte injection attacks dropping client connections.

Interestingly, [Singla et al., 2014, 2015] pointed out that fiber distance is up to two times longer than road distance, inflating the latency encountered in upper network layers. They examine the possibility of deploying nationwide lightspeed microwave networks. Our work will not address this aspect. We will focus on algorithmic improvements given physical deployments.

1.1.5 Weak multipath support

Reliable *multipath forwarding* is unnatural in the current Internet due to the fact that packets do not mandatorily follow the same paths back and forth. This IP feature, which happens to be a limitation for multipath*ing* reliable transport protocols like TCP, imposes at least some tunneling overhead to guarantee a successful merge of subflows into their original master flow [He and Rexford, 2008; Qadir et al., 2015]. It appears to be efficient only

for long-lasting flows. Such a poor multipath support is rather unfortunate, as multipath forwarding is an interesting means of exploiting multiple routes for bandwidth aggregation, load balancing and failover purposes, especially in challenging environments like those covered by Delay-and-Disruption-Tolerant Networking (DTN) [Cerf et al., 2007].

1.1.6 A non-native relief by CDNs

The precarious situation we portrayed justifies the extensive resort to overlay networks, CDNs. A *Content Delivery Network* or CDN consists in content caches and replica spread closer to the users. They are populated in order to anticipate the users needs and accelerate the delivery of content of interest. In fact, almost half the Internet traffic crosses CDNs and [Cisco, 2016] predicts a steady growth to that trend, reaching two-thirds by 2020. It is thanks to CDNs that the Internet still scale, by the confinement of a significant part of the traffic near the Internetwork edge. CDNs have somewhat paved the way to Information-Centric Networking, this thesis' cornerstone. However, the key difference is that CDNs are overlay, often commercially-operated networks. They often rely on DNS tricks or IP anycast to redirect request towards the closer content replica, so unnatural to the IP world that it inflates network complexity.

1.2 5G, an opportunity

The Fifth Generation Mobile Networks is the next major evolution of mobile systems. It is envisioned to fulfill unprecedented goal in terms of latency, data rate, number of devices and energy consumption. Among key requirements were identified [Maternia and El Ayoubi, 2016]: at least 1 Gbps of user data rate, very low latency : 1-10 ms end-to-end round trip delay, up to a hundred times more connected devices, 90% less energy consumption, up to 10 years of battery life and more than 99% availability. Several research works [Paschos et al., 2016; Augé et al., 2015; Kutscher, 2016] point towards ICN as candidate technology for sustaining such stringent data rate and latency figures.

1.3 Information-Centric Networking

Information-Centric Networking aims at re-engineering the Internet to make it fit its current and future challenges. First introduced by [Jacobson et al., 2009b], this new paradigm advocates the shift from a host-centric layer-3 communication model to a named-content-centered scheme.

This seminal work acknowledged that, decades ago, networking aimed at sharing scarce and expensive resources such as high-speed tape drives or supercomputers. Addressing machines into packet headers to tell the network *where* to convey them to, made sense then. The purpose of networking has fundamentally changed since, as computing power and data storage have been democratized. Nowadays, the Internet shares content/information and must be told *what* to deliver.

Whereas the World Wide Web, peer-to-peer (P2P) and CDNs have paved the way for named-data distribution over the existing Internet, ICN breakthrough dwells in its ubiquity. ICN ambitions to be run either as an overlay infrastructure like its predecessors or as a layer-3 protocol, capable of replacing IP wherever IP operates today.

The ICN research community expects that revamping networking with this updated mindset shall provide intrinsic security, mobility support, as well as enhanced scalability and robustness.

Among the architectures that adopted this approach, there are Data-Oriented Network Architecture (DONA) [Koponen et al., 2007] and Network of Information (NetInf) [Ahlgren and al., 2008]. They differ from the two most prominent architectures, CCN and NDN, in requiring content names to be published and subscribed for into a tree of trusted Resolution Handlers (DONA) or a Distributed Hash Table (NetInf).

1.3.1 Content-Centric / Named-Data Networking

Content-Centric Networking (CCN) [Jacobson et al., 2009b] and Named-Data Networking (NDN) [Zhang and al., 2010; Zhang et al., 2014] are two prominent and very similar Information-Centric Networking architectures. NDN started as a fork of CCN's codebase CCNx, to be almost entirely redeveloped later on. However, from the architectural point of view, CCN and NDN have not diverged. In both architectures, content chunks are identified by unique names, requested via *Interests* packets and retrieved as *Data* pack-

ets through (*inter*)faces. Because every packet encloses its name, it can be persisted at every hop into a *Content Store*, a cache, and delivered from the local node to subsequent requesters.

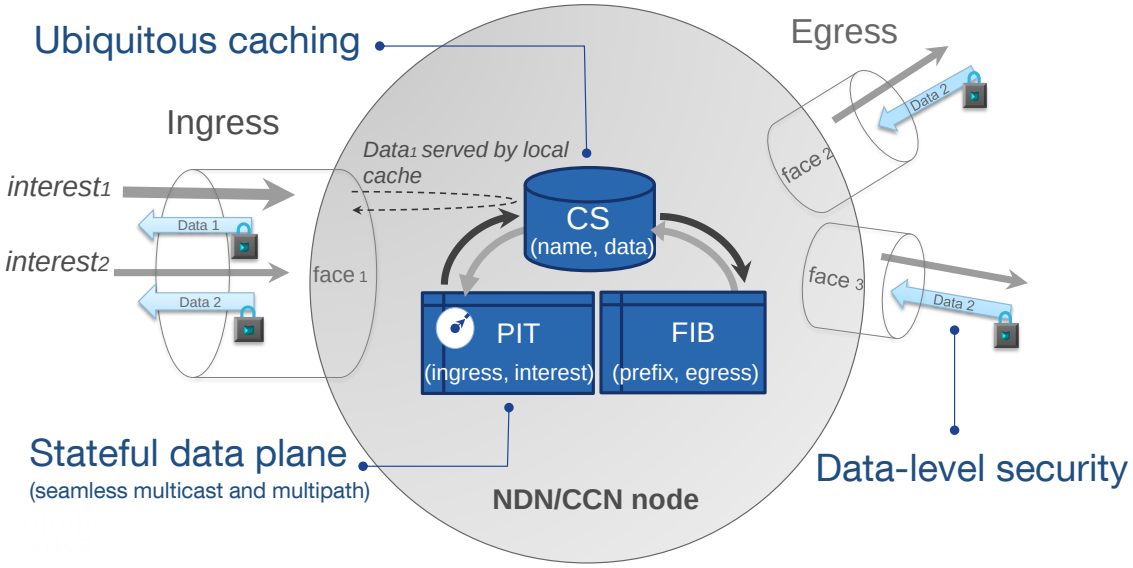


Figure 1.1 – An NDN/CCN node

1.3.2 NDN operations

As depicted in Fig.1.1, every crossed node, NDN/CCN keeps track in a *Pending Interest Table* (PIT) of the faces Interests originate from, the time this occurred and the content name they request. If a content chunk is found into the Content Store, called a *cache hit* event, it is delivered using the state information from the corresponding PIT entry. On the other hand, if that content chunk is not yet or no longer available in the Content Store, this event is denoted as a *cache miss*. Therefore, a Longest Prefix Match of the content name is looked for into the node's *Forwarding Information Base* (FIB). The FIB, populated by a name-based routing protocol, provides for every routable name prefix, egress faces to the missing chunk. Then, the Interest packet gets forwarded according to the configured strategy that might be, for example, Best Route, Broadcast, Load Balancing (LB) or Load Balancing with Persistent Forwarding (LB-Perf) we analyze in this thesis. When the

corresponding Data packet comes back, it is handled by a cache management mechanism such as LAC+ we advocate in this work, and finally sent back to its sequesters while the related PIT entry is deleted. Because of their prominence, common fundamentals and overall operations, both *NDN and CCN will be referred to as NDN* for conciseness throughout the document.

1.3.3 NDN security

A raison d'être of NDN is the need to build next-generation network protocols with security in mind. Still, part of the protection brought by NDN is implicit, being a consequence of its architecture. NDN's explicit security features are the product of decades of research and real-world deployments in public-key cryptography.

Implicit security induced by NDN's architecture

First, NDN flow balance that requires every single Data packets to be requested by a single matching Interest packet. As a consequence, it mitigates those of Internet DoS that amplify attackers probes. PIT aggregation and caching also contribute to DoS mitigation in filtering storms of interests that address the same object. ICN stateful forwarding plane may help alleviating routing information poisoning in allowing distributed Round-Trip Time (RTT) or Pending Interest- based forwarding strategies [Carofiglio et al., 2013c].

However, one of the most harmful DoS attack against an ICN router would be flooding the PIT with unsatisfiable Interests. Since the identity of the attacker is unknown, it may seem hard to find targeted countermeasures to this threat. Fortunately, NDN allows, by bounding the number of pending Interests per face and per prefix, and backpropagate a flooding containment to the faces directly in touch with the attackers [Gasti et al., 2013].

Not carrying source address helps customer's identity privacy, especially in dense networks, but does not seals it. The first hop router still able to unambiguously identify the customer attached to its ingress face.

Explicit security features in NDN

Beyond these features induced by NDN architecture, every data packet's name and payload are cryptographically signed for three major benefits: Data integrity ensuring that the

data has not been altered, Producer authentication verifying the identify of who signed the packet, Data validation ensuring data is the genuine reply to the expressed Interest.

While the current Internet security model (IPsec, SSL/TLS) secures entire sessions, ICN applies its data-centric approach to security in securing the data itself. This fine-grained perspective allows different trustees for different pieces of content. On a Web page, the tier signing a chart is not necessarily the text writer or the video producer. Content-centric security also allows every network stakeholder to check signature, not only communicating endpoints. This, for instance, mitigates the spread of tampered content since every intermediate router can check what it stores and forwards. Every Data packet refers to the certificate that signed it. A data packet conveying a public key might be such a certificate in its own right, being itself signed. The choice of a trust model is flexible. Examples are the classical Public Key Infrastructure (PKI) [Housley et al., 2002], or Certification Authority-free alternatives such as PGP's Web of Trust [Blaze et al., 1996] or the Simple Public-Key Infrastructure/Simple Distributed Security Infrastructure (SPKI/SDSI) [Ellison et al., 1996].

However, it might be argued that per-packet signature verification is impracticable at high wire speed due to its computational complexity [Ghali et al., 2014]. Merkle hash trees [Merkle, 1987] offer the ability to sign and verify signature on a group of packets hashes once, at a parent node. Given that the cost of Public-key algorithms such as RSA-2048 is, at its best, an order of magnitude higher than the cost of hash functions like SHA-1 [Jang et al., 2011], the improvement must be striking.

Data confidentiality is not an NDN feature per-se but is of the responsibility of the applications. However application-led encryption for data confidentiality is to the detriment of content caching.

1.3.4 NDN routing

NDN can reuse current Internet's routing protocols such as OSPF or BGP, differing by advertising name prefixes instead of IP addresses. However, the default NDN routing protocol is *Named-data-based Link State Routing* (NLSR) [Hoque et al., 2013]. In NLSR, every router pulls, via Interest packets, routing updates encapsulated in signed Data packets, in conformance with NDN conversational model. By extending Dijstra' shortest path algorithm, NLSR supports multiple next-hops per prefix while OSPF can only compute

single paths. Moveover, NLSR restricts the synchronization of link state information with neighbours only to minimize the dissemination cost.

While NLSR ultimately makes every router aware of every prefix, [Hemmati and Garcia-Luna-Aceves, 2015] recently proposed for scalability purpose, the Link State Content Routing protocol (LSCR). LSCR's key innovation is to restrict the awareness of each router to the nearest prefix replica.

An interesting alternative being explored is *hyperbolic routing* [Kleinberg, 2007]. The argument is that due to its heterogeneity and strong clustering, NDN namespace has an underlying hyperbolic topology, like the World Wide Web and various complex networks [Krioukov et al., 2010]. Hyperbolic routing maps nodes and prefixes onto an hyperbolic metric space and proposes to route packets towards the neighbour that is the closest to the prefix, only by distance calculation between two hyperbolic (radius, angle) coordinate pairs. Instead of updating every router with the whole network topology (nodes, prefixes and link states), routers just maintain a list of their neighbours coordinates, a list of prefixes coordinates for deducing next hops. The expected benefit is much less routing information updates, since coordinates do not change upon network conditions or topology changes and because of NDN forwarding plane ability to dynamically select alternate hops. A weakness of this approach is that the best hyperbolic distance might be much longer than the shortest path. Moreover, this type of coordinate-based greedy forwarding still not guarantees content delivery as there might be local minima cases where none of a node's neighbours is closer to the destination than the node itself. [Papadopoulos et al., 2010] sees a solution to this last issue in forwarding trapped packets to the closest-to-the-destination node among the least visited neighbours.

1.3.5 NDN forwarding

An IP router has no visibility on the receiver's flow-control window and sender's congestion windows. It can not regulate ongoing TCP sessions that are end-to-end by design. The specificity of NDN forwarding plane is its stateful nature. The innovative PIT structure keeps track, hop-by-hop, of pending Interests and their incoming faces *i.e.*, every ongoing transfer. Every Data packet takes the exact reverse path of the route their Interest counterpart took. Since a nonce field uniquely identifies Interest packets addressing the same object, the PIT structure can detect and drop looping Interests.

Given these architectural assets, every NDN node provides one on more forwarding algorithms within what is called a *strategy layer*.

Two types of algorithms essentially operate in that layer: a *receiver-located* controller, often of the Additive-Increase-Multiplicative-Decrease (AIMD) kind; and *hop-by-hop* Interest shapers or, when multiple egress faces exist for a given prefix: load balancers, broadcasters or best route selectors.

The purpose of a receiver-located controller is to fully use the available bandwidth by managing a global congestion window. It might be complemented by some Active Queue Management to sense forthcoming capacity exhaustion and decrease the congestion window with a per-route probability that is proportional to a smoothed Round-Trip Delay [Carofiglio et al., 2013c].

Thanks to the Interest/Data perfect symmetry, flow and congestion control are feasible at each hop by means of Interest shaping. Hop-by-hop Interest shaping leverage rapid and distributed decision making on bursty traffic [Carofiglio et al., 2012; Wang et al., 2013b]. Observe that it is more convenient to buffer Interest packets than Data considering their incomparable size difference. Moreover, NDN's native multipath support can balance traffic load over several egress faces or adaptively choose different delivery paths on events such as congestion or link failure [Carofiglio et al., 2013c].

1.3.6 Mobility in NDN

User mobility in IP is complex because a wireless link failure may disrupt ongoing TCP sessions. Part of the problem is the lack of native multipath support in TCP, that would have allowed mobiles to connect simultaneously to several networks for a seamless handover [Raiciu et al., 2011]. Conversely, NDN possesses some native multipath support. This owes to its connection-less conversational model, made reliable and symmetric by the hop-by-hop states maintained in the PIT.

Producer mobility is becoming a stringent need in today's Internet where applications like Facebook Live and Periscope have democratized video streaming from mobiles. Whereas content producer mobility in IP relies on tunnels to anchors, in NDN, *anchor-less mobility* has been proven feasible [Augé et al., 2015]. It exploits ICN ubiquitous caching to alleviate producer temporary unavailability, and a Temporary FIB at every router on the path to the producer's former location to reroute traffic.

1.3.7 Caching in NDN

[Imbrenda et al., 2014] demonstrated by means of real traces that a negligible amount of cache space in customer premises (100MB) may reduce the load within Fiber-To-The-Home (FTTH) networks by 25%, whereas a 100GB cache downstream a backhaul link might offload it by 35%. Once questioned, it reinforces on-path in-network caching in its legitimacy. It makes clear that it would be useless to deport all caching memory to the very edge of the network, as suggested in [Fayazbakhsh et al., 2013; Garcia-Luna-Aceves et al., 2014], due to existing-but-insufficient redundancy within the traffic at that stage. The multistage approach of on-path caching combines exploiting intra-customer's traffic redundancy, with inter-customer/ inter-domain redundancy that traffic aggregation unveils. From the architectural viewpoint, NDN matches what [Paschos et al., 2016] foresees for next-generation wireless networks, advocating the penetration of CDN within the Radio Access Network (RAN), spanning mobile devices and base stations, to track any form of traffic redundancy.

Hence, while IP routers are incapable of serving buffered packets to a session other than the one that requested them, cached packets in NDN are entirely reusable since they are uniquely named. Caches are expected to be in every single NDN node. They are a key part of the architecture. Even if caches are supposed to be much bigger than today's routers packet buffers, they remain obviously finite. A management policy must decide about the items to keep in the Content Store and those to evict. First-In-First-Out, (*evict the*) Least Frequently Used (LFU) or various enhancements to the (*evict the*) Least Recently Used policy (p-LRU, LRU+LCD) [Laoutaris et al., 2004] have often been appointed to that office.

The performance of a Named-Data Network bears a clear dependency upon the chosen cache management policy. We hereafter describe a few.

Least Frequently Used policy

LFU consists in evicting the object that has accounted the least downloads during the last time window. This way, only the most popular objects of that period remain into the cache. Assuming stationary popularity distribution *i.e.*, that content popularity does not change over time, this is among the most effective policies for load reduction. However, it suffers from two weaknesses:

(i) Insensitivity to the network state. LFU handles every content it sees only based of its local popularity. It simply ignores the true purpose of caching, improving the user Quality of Experience (QoE).

Example 1.3.7.1. *Consider a Gigabit Ethernet home network, hosting a Network-Attached Storage (NAS). The NAS allows the family to share their favourite music, pictures and videos. From their subnetwork, while half the kids would like to browse the NAS, the other half is captivated by the latest viral videos on the Internet.* ***LFU on the kids NDN router will treat both content categories the exact same way*** *and equally populate its Content Store with material from the NAS, and hard-to-retrieve Internet videos from saturated servers. This is clearly suboptimal.*

(ii) Delayed reaction to popularity variations. LFU needs to collect statistics during a relatively long time frame to hold meaningful popularity figures about every content. For this reason, it performs excellently when there exists a stationary content popularity distribution. In that case, the long-term average of a given content's ratio of downloads is the probability that any given download conveys it. The latter condition characterizes the Independent Reference Model (IRM) for cache performance analysis we will develop later in this chapter. Under the IRM hypothesis, no surprise that LFU is often regarded as the ultimate blueprint [Martina et al., 2013]. However Shot Noise Models (SNM) [Leonardi and Torrisi, 2015; Olmos et al., 2015] where objects are published and end up perishing offer more realistic but less tractable alternatives that do not favor LFU.

We address both shortcomings with the Latency-Aware Caching + policy (LAC+) we introduce in the next section.

First-In-First-Out policy

Here, the oldest content in the cache is evicted first. Requesting an object and finding it in the cache (what is called a hit event) does not change its fate as it leaves the cache unmodified. Such a behaviour does not exploit content popularity to the fullest, since the cache has no means of marking solicited objects by altering its state accordingly. However, a content's hit ratio in a FIFO cache still reflects that content's popularity. Popular objects still have higher hit ratio because of the slow down of the eviction dynamics due to more frequent hits and more frequent insertion after miss events.

A FIFO cache is often implemented as a queue: every new insertion shifts other objects one position to the cache exit. Its simplicity makes it the default cache management policy in NDN. Moreover, from an engineering standpoint, chunk lookup has lower complexity in chunk-oriented queues thanks to less fragmentation: packet retransmissions leave a sequence of chunks, contiguous.

Least Recently Used policy

The Least Recently Used policy evicts the object whose last access is the oldest. A simple implementation of this policy consists in moving an object found in a FIFO cache, to its front. This is referred to as the *Move-To-Front* (MTF) algorithm [Starobinski and Tse, 2001]. The MTF mechanism confers a much better hit ratio on LRU than FIFO's thanks to a more efficient popularity sampling. The most popular objects tend to stay longer in the cache owing to a sort of Time-To-Live reset on object access unveiled by [Choungmo Fofack et al., 2012]. LRU seamless adaptation to popularity changes and its simplicity makes it a much better real-world alternative to LFU, explaining somehow why it has been so often studied [Jelenković and Kang, 2008; Fricker et al., 2012; Leonardi and Torrisi, 2015]. LRU's main drawback remains its lower hit ratio, asymptotically $e^{\gamma} \approx 1.78$ times away from LFU, where γ is Euler's constant. [Jelenković, 1999].

LRU and Leave-a-Copy-Probabilistically policy (p-LRU)

Take an LRU cache and decide, instead of always inserting every newly downloaded object on a miss event, to do so with a fixed probability p. This is the p-LRU policy. As surprising as is may seem, doing so, voluntarily by computationally enforcing that probabilistic insertion with the assistance of a Pseudo-Random Number Generator (PRNG), or due to some involuntary data corruption [Bianchi et al., 2013], increases cache hit ratio. In fact, the closer to zero p, the closer to LFU the p-LRU system [Martina et al., 2013]. This is an important result that drove the design of our policies LAC and LAC+.

However, as p tends to zero, so few items get stored that p-LRU, $p \to 0$ can only work under the stationary popularity assumption.

LRU and Leave-a-Copy-Down policy (LRU+LCD)

Consider that every downloaded objects crosses a sequence of LRU caches. A policy that consists in storing an object only if it does not come from farther than the preceding cache is called Leave-a-Copy-Down. In other words, if the download path of content k traverses caches $1, \cdots, n$, LCD is the slight alteration to LRU that inserts content k in cache n (after a miss event) only if the search for that content in cache $n-1$ resulted in a hit event [Laoutaris et al., 2004]. When gathered inside the same machine, a sequence of k LRU+LCD caches form what is called an LRU-k cache. However, in LRU-k, the first $k-1$ caches just store object identifiers, not actual content.

Theory predicts that LRU+LCD converges to LFU as n grows large [Martina et al., 2013]. In practice, the convergence is quite fast: we observed in most cases that $n = 4$ suffices to persist the most popular items in the n^{th} cache. LCD works as a multistage filter where the most popular objects make it through all stages.

Nevertheless, there are two main weaknesses to the LRU+LCD algorithm:

(i) It does not support traffic split. LCD filtering is jeopardized if upstream caches do not contain the same set of objects. It means that the content popularity distribution of the miss traffic reaching every egress cache must be identical for LCD to work properly. The reason of this limitation is that every path will isolate the most popular subset of a subset of the catalog, but the union of both subsets inside the cache where the split occurred is not guaranteed to be the most popular subset of the whole catalog. The only multipath forwarding strategy that is LCD-compliant is a content-blind Load Balancing. Later in this thesis, it will be observed that LRU+LCD underperforms under our Persistent forwarding. It owes to the present limitation.

(ii) It does not support content split. Consider a cascade of LRU caches operating at content level, whereas the traffic consists of content chunk. LRU+LCD will fail unless we extend the download path by a factor equaling, approximately, the average number of chunks per content. This limitation can be observed when trying to sample content popularity from data packets by means of an LRU-k filter. The solution that consists in identifying some content with one of its chunks, chunk 0, for example, is not satisfactory. Indeed, this does not cope with packet retransmission or partial transfers.

RANDOM policy

RANDOM is probably the simplest eviction policy to implement: choose randomly into the cache the content to replace. All objects in the cache have equal probability to be evicted. [Gallo et al., 2012] demonstrated that the policy still offers some traffic reduction and is even suitable for high speed routers. However, its hit ratio per content is significantly less than LRU's but equal to FIFO's [Gelenbe, 1973] under the Independent Reference Model (IRM) assumption we will describe later in the chapter.

1.4 Problem statement

This thesis aims at providing answers to the following questions:

1. *Can content delivery in ICN be faster ?*
2. *Can content delivery in ICN be fair ?*
3. *Can ICN Forwarding Information Base (FIB) scale ?*

In a nutshell, our answers to these questions are:

1. ***Yes** ICN can even be* 60% *faster*, by using for example our joint caching and forwarding mechanism, FOCAL.
2. ***Yes** ICN can be fair*, just share bandwidth fairly between contents.
3. ***Yes** ICN's FIB can scale*, for example using our AFFORD mechanism that introduces Artificial Neural Networks into the content forwarding realm.

We will elaborate on these problems and propose solutions throughout the document, starting from the next section with an brief introduction on our contributions.

1.5 Our contributions

The contribution of this thesis is fourfold:

1.5.1 LAC/LAC+: Latency-aware caching

We designed a family of randomized dynamic cache management policies leveraging in-network retrieval latency for cache insertion. The locally monitored metric is the time elapsing at a given node between request forwarding and corresponding packet reception. The cache management mechanisms LAC and LAC+ [Carofiglio et al., 2015a,c] we propose consist in the following: every time an object is received from the network, it is stored into the cache with an overall low probability that significantly increases if the content exhibits an exceptional retrieval latency. As such, it is an add-on laying on top of an LRU cache and feeding it at a regulated pace. This way, it implicitly *prioritizes popular and long-to-retrieve objects*, instead of caching every object regardless. The underlying trade-off such a caching mechanism tackles is between a limited cache size and delivery time minimization. As caching intrinsically aims to relieve the fallouts of network distance or traffic congestion, it must be aware of both popularity and delay factors to efficiently handle that cache size / delivery time tradeoff. An overall low probability of insertion samples popular objects whereas data retrieval latency is a simple, locally measurable and consistent metric for revealing either haul distance or traffic congestion.

The applicability of our latency-aware cache management policy is broad: Information-Centric Networking networks, data centers, content-distribution networks or multiprocessor server optimization.

We authored two C++ implementations of LAC and LAC+, one for the CCN Packet Level Simulator (CCNPL-Sim [1]) and another for the NS-3 -based NDN Simulator (NDNSim) that shares code with the NDN Forwarding Daemon.

Chapter 2 substantiates LAC and LAC+ algorithms.

1.5.2 FOCAL: joint Forwarding and Caching with Latency-awareness

Caching needs to be supported by proper packet forwarding [Rossini and Rossi, 2014; Dehghan et al., 2015] as the performance of the first is driven by the request arrival process the second is responsible for. At the same time, the traffic to be forwarded is the miss traffic of the local cache. From this observation it appears clearly that *caching and*

[1] http://systemx.enst.fr/ccnpl-sim.html

forwarding must be jointly optimized.

It is the purpose of FOCAL [Carofiglio et al., 2015b, 2016]. It adopts *LAC+* caches, fed by a novel forwarding strategy *LB-Perf* that persistently directs the most popular objects through the same interfaces, regularly making sure they can afford it, while load balancing the rest of the traffic.

Chapter 3 substantiates FOCAL. It shows how FOCAL is deduced from the optimal solution of a latency minimization problem. Moreover, we analyze FOCAL stability, thoroughly evaluate its CCNPL-Sim C++ implementation on various network topologies and analyze its sensitivity to various settings.

1.5.3 Fairness in Information-Centric Networking

Chapter 4 investigates the fairness of content throughput allocation when caching becomes ubiquitous throughout a network [Bonald et al., 2017]. As caching the most popular objects is the trend our own caching algorithms enforce, will there be a severe distortion to fairness caching algorithms have to be designed to resorb ? It turns out that ensuring the most popular objects occupy storage, as LFU does, is the caching side of the solution to α-fair throughput allocations problems. α-fair content-level packet scheduling is the complementary part of the solution.

In other words, caching policies do not need to be designed for fairness as previous work suggested. Not focusing on cache hit ratio but on the throughput fairness, show that *α-fairness in ICN can be handled in a similar way as in traditional networks, through packet scheduling*.

1.5.4 AFFORD: Ask For Directions, machine learning-based routing

The last contribution of this thesis addresses the prohibitive cost of performing a Longest Prefix Match over a huge FIB. Indeed, considering the unlimited namespace of the anticipated Internet of Everything, it would be hard to forward packets at every node based on an exact knowledge of paths to destinations. With Ask For Directions (AFFORD) [Mekinda and Muscariello, 2016], we propose to *train a trie of small Artificial Neural Networks in the control plane and fast-interrogate them in the data plane for the most*

probable next hops.

Chapter 5 substantiates the novel AFFORD routing approach.

1.6 Mathematical foundations

Little introduction or reminder on a few mathematical concepts should ease the reading of this document.

1.6.1 Elements of probability theory

A stochastic process $\{X(t)\}_{t\geq 0}$ is a sequence of random variables $X(t)$ on the same probability space, indexed by time instants t [Tucker, 2013].

Continuous-Time Markov Chain

Also known as Markov process, a CTMC [Bertsekas et al., 1992] is a stochastic process $\{X(t)\}_{t\in\mathbb{R}_+}$, taking values from a countable state space $S = \{i_n : n \in \mathbb{N}\}$, and such that:

1. The time T_i the process spends in every state $i \in S$ is such that $T_i \sim \text{Exp}(\nu_i)$.
2. The process leaves every state $i \in S$ to enter another state j with probability P_{ij} such that $\sum_j P_{ij} = 1$.

Let $q_{i,j} \equiv \nu_i P_{ij}$ and $p_j = \lim_{t\to\infty} \mathbb{P}[X(t) = j | X(0) = i]$. p_j, in the special case of *regular* CTMC (almost sure finite number of transitions over any finite time interval), exists and is independent of the initial state i.

The *global balance* equations for a CTMC are:

$$p_j \sum_{i=0}^{\infty} q_{ji} = \sum_{i=0}^{\infty} p_i q_{ij}, \forall j.$$

The *detailed balance* equations for a CTMC, which hold for birth-death processes, are:

$$p_j q_{ji} = p_i q_{ij}, \forall i, j.$$

Note that *birth-death processes* are Markov processes where $q_{ij} = 0$ for $|i - j| > 1$. Let $j \equiv i + 1$, q_{ij} is called a birth rate and q_{ji} is called a death rate. If $\forall i, q_{ji} = 0$, the process is called a *pure birth process*.

Counting Process

A counting process $\{X(t)\}_{t\geq 0}$ is a stochastic process whose sample path is a non-decreasing, right-continuous step function taking nonnegative integer values *i.e.*, $X(t) \in \mathbb{N}$ and $(s \leq t \Rightarrow X(s) \leq X(t))$. Also, $X(0) = 0\ a.s.$. Since $X(\cdot)$ increases by jumps only, occurring at specific times called *jump times*, it is a kind of *jump process*.

Renewal Process

A renewal process [Cinlar, 2013] $\{S_n\}_{n\in\mathbb{N}^*}$ is a stochastic process where:

1. S_n is elapsed time up to the n^{th} jump of a companion counting process $\{X(t)\}_{t\geq 0}$. $S_n, \forall n$, are called *renewal times*.
2. the *sojourn or holding times*, *i.e.*, the inter-jump times T_n, of its companion counting process are independent and identically distributed with mean m.

Theorem 1.1 (Elementary Renewal Theorem)**.** *For a renewal process,*

$$\lim_{t\to\infty} \frac{X(t)}{t} = \frac{1}{m}\ a.s.\ \textit{by the Strong Law of Large Numbers and}$$
$$\lim_{t\to\infty} \frac{I\!E[X(t)]}{t} = \frac{1}{m}.$$

Poisson Process

A stochastic process $\{X(t)\}_{t\geq 0}$ is a Poisson process with intensity (or rate) ν if

1. $\{X(t)\}_{t\geq 0}$ is a counting process.
2. The number of arrivals that occur in disjoint time intervals are independent.
3. The number of arrivals in any interval of length τ is said to be Poisson distributed with parameter $\nu\tau$ *i.e.*,

$$\forall n \in \mathbb{N}, \tau > 0, \mathbb{P}[X(t+\tau) - X(t) = n] = \frac{(\nu\tau)^n}{n!} e^{-\nu\tau}.$$

Then, inter-arrival times T_n are exponentially distributed with parameter ν *i.e.*,

$$\forall \tau \geq 0, \mathbb{P}[T_n \leq \tau] = 1 - e^{-\nu\tau}.$$

This distribution is *memoryless*, in the sense that $\mathbb{P}[T_n > \tau + t | T_n > t] = \mathbb{P}[T_n > \tau]$.

Due to its exponential sojourn times T_n and non-decreasing states, a Poisson process is a pure birth process with constant birth rates. The arrival times of a Poisson process *i.e.*, the sums of the $X(t)$ sojourn times form a renewal process.

A Poisson process, defined earlier as a counting process may also be interpreted through its increments during time intervals. As such, it is an *homogeneous* (or *stationary*) Poisson point process defined on the real line. It is said to be stationary because of its constant intensity ν, which implies that the distribution of points does not depend on the interval position on the real line. As such, it is translation-invariant. Otherwise, the process is called an *inhomogeneous* Poisson point process.

1.6.2 Queuing theory fundamentals

Throughout this thesis, our usage of the latency owes to queuing delays. It is the time an object (packet or flow), also referred to as customer, spends in the queuing system [Bertsekas et al., 1992; Médard, 2008].

A queuing system is characterized by a stochastic process $\{N(t)\}_{t\geq 0}$ giving the number of customers in the system at time t.

Parameters

The queue has two main parameters, the number of customer arrivals resp. departure up to time t, $\alpha(t)$, resp. $\beta(t)$. We usually consider known the long-term arrival rate $\lambda = \lim_{t\to\infty} \alpha(t)/t$ and the service time random variable X such that the mean service rate $\mu = (\mathbf{E}[X])^{-1}$.

Properties

We will be interested in knowing:

The long-term average number of customers in the system. We denote it

$$\bar{N} = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau N(t) dt.$$

Provided that the system is assumed *ergodic*, the long-term average number of customers equals the mean number of customers as $t \to \infty : \bar{N} = \lim_{t \to \infty} \mathbb{E}[N(t)]$. Furthermore, the long-term arrival rate equals the long-term departure rate *i.e.*,

$$\lambda = \lim_{t \to \infty} \frac{\beta(t)}{t} < \mu.$$

The long-term average of the time spent waiting in the queue denoted $\bar{W}$.

The long-term average of the time spent in the system. Denoted $\bar{T}$, it is the also known as the long-term average delay encountered by any customer waiting in the queue, then being served. it is also called *mean sojourn time*. Let T_k be the time spent in the system by a given customer k. The cumulative time spent by all the customers in the system divided the total number of arrivals at the infinity gives $\bar{T}$ *i.e.*,

$$\bar{T} = \bar{W} + \mathbb{E}[X] = \lim_{t \to \infty} \frac{1}{\alpha(t)} \sum_{k=1}^{\alpha(t)} T_k.$$

The steady-state probability of finding n customers in the system is assumed to exist and equals

$$p_n = \lim_{t \to \infty} \mathbb{P}[N(t) = n].$$

Theorem 1.2 (Little's theorem)**.** *The long-term average number of customers in the system equals*

$$\bar{N} = \lambda \bar{T}.$$

Thanks to its broad applicability, Little's theorem also holds for the long-term average number of customers waiting in the queue, or those being served. Furthermore, it holds

for a given class of customers.

Kendall's notation

Queues are classified according to the following compound expression:

<Arrival type>/<Service Type>/<Number of servers>[/<Buffer size>][-<Discipline>].

Arrival and service types may be, for example, **M**arkov or Memoryless, which implies exponential inter-arrival times, or **D**eterministic or **G**eneral.
By default, the buffer size is deemed infinite. Among service disciplines, there are First-In-First-Out (FIFO) the default discipline handling customers sequentially, or Processor Sharing (PS) handling all customers simultaneously.

FIFO service discipline

$M/G/1/\infty$-FIFO. In $M/G/1$, customers arrive according to a Poisson process but the service times obey to a law that is not necessarily exponential. The Pollaczek-Khinchin formula is a key result for understanding such a system.

Theorem 1.3 (Pollaczek-Khinchin formula)**.** *Let $\bar{W}$ be the long-term average time spent by a customer waiting in the queue. Define the load of the queue $\rho = \lambda/\mu$. The waiting time of an $M/G/1$ queue is*

$$\bar{W} = \frac{\lambda \mathbb{E}[X^2]}{2(1-\rho)}.$$

The related delay/sojourn time follows by adding the mean service time:

$$\bar{T} = \frac{\lambda \mathbb{E}[X^2]}{2(1-\rho)} + \mathbb{E}[X].$$

By Little's theorem,

$$\bar{N} = \frac{\lambda^2 \mathbb{E}[X^2]}{2(1-\rho)} + \rho.$$

The Pollaczek-Khinchin formula is the general framework for expressing the first-moments of more specific queues.

M/M/1/∞-FIFO. In $M/M/1$, $X \sim$ Exp(μ). The second moment of X is $\mathbb{E}[X^2] = 2/\mu^2$. It follows that

$$\bar{W} = \frac{\rho}{\mu - \lambda}; \qquad \bar{T} = \frac{1}{\mu - \lambda}; \qquad \bar{N} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}.$$

The same results could have been derived by modeling the $M/M/1$ queue as a Continuous-Time Markov Chain, observing from the detailed balance equations that its steady-state measure is a geometric probability mass function. This second way remains the easiest way to obtain the queue's stationary distribution.

$$p_n = \rho^n(1-\rho).$$

M/D/1/∞-FIFO. In $M/D/1$, $X = 1/\mu \; a.s.$ The second moment of X is $\mathbb{E}[X^2] = 1/\mu^2$. It follows that

$$\bar{W} = \frac{\rho}{2(\mu - \lambda)}; \qquad \bar{T} = \frac{1}{2(\mu - \lambda)}; \qquad \bar{N} = \frac{\lambda}{2(\mu - \lambda)}.$$

Processor sharing service discipline

The Egalitarian Processor Sharing discipline (PS) consists in serving the n customers in the system with equal rates μ/n. As such, it is an idealization of the Round-Robin scheduling algorithm. Customers are served immediately upon arrival. The PS discipline is known to be insensitive to the distribution of the service time X beyond the mean. This implies that the system's first-order properties will simply depend on mean values such as $\mathbf{E}[X]$.

PS insensitivity is clear from the following $M/G/1$-PS mean properties, which by the way coincide with those of $M/M/1$-FIFO:

$$\bar{T} = \frac{1}{\mu - \lambda}; \qquad \bar{N} = \frac{\lambda}{\mu - \lambda}.$$

Processor sharing queues have proven to be useful models for performance analysis of fluid approximations of the traffic, under fair bandwidth sharing. TCP sessions whose arrival is assumed to follow a Poisson process have been modeled this way [Fredj et al., 2001].

1.6.3 Lyapunov optimization

As in [Georgiadis et al., 2006], we verify the stability of a queuing network under some optimization policy by use of a *Lyapunov function*. A Lyapunov function is a nonnegative scalar measure on a vector representing the state of the system at a given time. In our context, that state is the vector process of all queue backlogs (or unfinished work). The expected change in the Lyapunov function between two consecutive epochs is called the *Lyapunov drift*. It is aimed to be bounded by a small negative value whenever the sum of queue backlogs grows sufficiently large, to ensure the system stability. The minimization of a quadratic Lyapunov function drift in a queuing network is demonstrated to be throughput optimal and leads to the backpressure algorithm.

1.6.4 Nonlinear optimization

In this thesis, we tackle nonlinear optimization problems, either featuring a nonlinear objective function to be minimized or maximized, or encompassing nonlinear constraints. Such problems are of the following form:

$$\begin{cases} \text{Minimize } f(\mathbf{x}) \\ \text{subject to :} \\ \qquad g_i(\mathbf{x}) \leq 0 \quad i = 1, \cdots, N \\ \qquad g_j(\mathbf{x}) = 0 \quad j = N+1, \cdots, M. \end{cases}$$

where f and $g_i, i = 1, \cdots, M$ are continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$ [Bertsekas, 1999].

Define the function $\mathcal{L}(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \sum_{i=1}^{M} \lambda_i g_i(\mathbf{x})$ called the *Lagrangian* of the problem. $\lambda \equiv (\lambda_i)$ is a vector of the so-called *Lagrangian multipliers*. Define $\mathbf{x}^*$ as a local minimum and $\lambda^* \equiv (\lambda_i^*)$ as the vector of optimal multipliers.

Definition 1.1. *Vector $\mathbf{x}^*$ is said to be regular, if the constraint gradients $\nabla g_i(\mathbf{x}^*), \forall i$ are linearly independent.*

The *Karush-Kuhn-Tucker (KKT) Necessary Conditions* state that, under the assump-

tion that $\mathbf{x}^*$ is regular, there exists a unique vector of multipliers λ^* such that

$$\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*, \lambda^*) = \vec{0}. \quad \textit{(stationarity condition)}$$

Furthermore, the *primal feasibility conditions* hold:

$$\begin{aligned} g_i(\mathbf{x}^*) &\leq 0 \quad i = 1, \cdots, N \\ g_j(\mathbf{x}^*) &= 0 \quad j = N+1, \cdots, M. \end{aligned}$$

The *dual feasibility conditions* also hold:

$$\lambda_i^* \geq 0 \quad i = 1, \cdots, M$$

as well as the *complementary slackness conditions*:

$$\lambda_i^* g_i(\mathbf{x}^*) = 0 \quad i = 1, \cdots, M.$$

The complementary slackness conditions aim at enforcing that at any local minimum, every constraint i is either saturated, (meaning that $g_i(\mathbf{x}^*) = 0$) or inactive (meaning that $\lambda_i^* = 0$).

Definition 1.2 (Convex set)**.** *A convex set within an Euclidean space is a region such that a line segment joining any pair of points lies entirely in that region.*

Definition 1.3 (Convex function)**.** *A twice continuously differentiable function is convex on a convex set iif its Hessian matrix is positive semi-definite on the interior of that convex set.*

If the Lagrangian function is convex on its domain, KKT optimality conditions are also *sufficient*; hence $\mathbf{x}^*$ is also a global minimum.

To tackle a maximization problem, just replace $f(\mathbf{x})$ by $-f(\mathbf{x})$ to convert it into its minimization counterpart where the above KKT conditions pertain.

1.6.5 Cache performance analysis

A substantial part of our work involves modeling caches. In this section, we present the fundamentals of cache performance analysis.

The IRM hypothesis

The *Independent Reference Model* (IRM) assumes an immutable set of independent content-k request processes, $\forall k$, that must all be homogeneous Poisson processes [Martina et al., 2013]. As the superposition of independent Poisson processes, the aggregate request traffic itself follows a Poisson process. This stationary traffic model, though simple and incapable of accounting for any *temporal locality*, has been effective on real ISP traffic at short timescale ($\sim$ 24 hours) [Imbrenda et al., 2014]. Authors found discrete Weibull with shape 0.24 to better fit the empirical popularity distribution than the usual Zipf law with skewness $0.7 \leq \alpha \leq 0.85$.

At a longer scale, documents get published and perish, resulting in catalog dynamics that only more sophisticated models can capture. Thus, Shot Noise Models (SNM) [Traverso et al., 2013; Olmos et al., 2014] have been introduced to render the reality of time-varying content popularity, at the cost of a diminished tractability. In SNM, a Poisson cluster request process aggregates independent content-k request processes, $\forall k$, which are inhomogeneous Poisson processes of rising-and-vanishing intensities.

In this thesis, similarly to recent works in the field, [Garetto et al., 2015, 2016], our models assume stationary traffic. Nevertheless, whereas others use some form of renewal traffic (general or ON-OFF-modulated homogeneous Poisson processes), we assume exponentially distributed inter-request times *i.e.*, IRM, reasonable at short timescale.

Che's approximation

Some exact cache performance analysis exist, based on the stationary distribution of a Markov process [Gelenbe, 1973]; but they are impeded by their intractable computational complexity that grows exponentially with the cache and the catalog size [Dan and Towsley, 1990]. A prominent approach for taming the complexity of cache analysis, introduced in [Che et al., 2006], is referred to as Che's approximation.

Under IRM and according to Che's approximation of the LRU cache hit ratio content k hit probability in a cache of size x is

$$h_k(\tau_x) = 1 - e^{-q_k \tau_x}.$$

Here q_k is content k popularity and τ_x is the cache Characteristic Time. It is deemed

the maximum time before object eviction under normalized content arrival rate. It can be interpreted as the time for the arrival of x distinct content requests.

The approximation consists in considering that τ_x is constant and the solution of the equation:

$$\sum_k h_k(\tau_x) = x,$$

This is actually an approximation for two reasons. First, a miss event for any content k occurs after a duration necessary to witness the arrival of requests for x distinct content items *other than content* k. Hence, τ_x should depend on k and be $\tau_{x,k}$. However a unique τ_x can be assumed as long as $\forall k, q_k \ll \sum_i q_i$. Second, assuming Zipf content popularity distribution and for a sufficiently large catalog, τ_x is a Gaussian variable that tends to its mean almost surely when $x \to \infty$ [Jelenković and Kang, 2008; Fricker et al., 2012].

Che's approximation has showed to hold in case of chunk-oriented caches, not-so-large cache sizes [Fricker et al., 2012], Poisson cluster (Shot Noise) traffic [Leonardi and Torrisi, 2015; Olmos et al., 2015], renewal traffic and even for other eviction policies like FIFO and RANDOM [Fricker et al., 2012; Martina et al., 2013]. It has been a widely adopted approach to cache performance analysis because it decouples distinct contents dynamics, making each content's hit ratio only depend on its own request process and on a unique cache property, namely, its Characteristic Time.

Scale invariance

Under IRM and Che's approximation, the hit ratio of every object in an LRU cache does not depend on the intensity of the *request* arrival process $\lambda > 0$. The request arrival process is the superposition of independent content-k request processes of intensity $\lambda_k = q_k\lambda$. Let $\hat{\tau}_x$ be the Characteristic Time of the cache under this Poisson request arrival. Define T_k, content k's sojourn time in the cache. Content k hit ratio writes

$$\hat{h}_k(\hat{\tau}_x) = \mathbb{P}[T_k \leq \hat{\tau}_x] = 1 - e^{-\lambda_k \hat{\tau}_x}.$$

By Che's approximation, $\hat{\tau}_x$ is the solution of

$$\sum_k \hat{h}_k(\hat{\tau}_x) = \sum_k (1 - e^{-q_k \lambda \hat{\tau}_x}) = x.$$

Observe, this is strictly equivalent to solving

$$\begin{cases} \hat{\tau}_x = \tau_x / \lambda, \\ \sum_k (1 - e^{-q_k \tau_x}) = x. \end{cases}$$

Hence,

$$\hat{h}_k(\hat{\tau}_x) = 1 - e^{-q_k \lambda \tau_x / \lambda} = h_k(\tau_x).$$

This is rather intuitive. Scaling up the arrival rate of all content requests *by the same factor* simply scales down the cache Characteristic Time by that factor, but results in the same hit ratios since the relative dynamics of objects in the cache remains unchanged.

Conversely, scaling down the *data* arrival process (that occurs after cache misses and subsequent data downloads) significantly changes the relative dynamics of the objects in cache. It is the rationale behind the p-LRU algorithm. Less popular contents, more prone to miss events, get recursively deprived, while increasing the hit ratio of the most popular objects. In the next chapter, our LAC and LAC+ algorithms exploit such an asymmetrical filtering, following in that p-LRU footsteps.

1.6.6 Performance analysis of Networks of Caches

Networks of caches are hard to analyze because the IRM assumption no longer holds after the first-level caches [Kurose, 2014]. Indeed, the miss traffic, flowing through a cache's egress faces carries time correlations. The IRM hypothesis allows two consecutive exogenous requests to address the same content, they are deemed totally independent. Conversely, within cache miss streams, two consecutive requests are not likely to address the same content since the first would have triggered a cache hit event for the second. In absence of connectivity issues, *PIT entry aggregation*, which mitigates the forwarding of Pending Interest packet duplicates, makes same-content consecutive misses much less

probable.

In case of LRU caches for example, the Characteristic Time dictates the minimum time interval between two same-content consecutive misses. Thus, the miss traffic essentially carries queries for distinct content per Characteristic time. Clearly, within the miss traffic, the number of requests for any content k does not follow a Poisson process, as their inter-arrival time does not follow an exponential distribution.

[Jelenković and Kang, 2008] demonstrates that inadequacy of the Poisson assumption and suggests to model the miss traffic as a superposition of asymptotically independent (when cache size x tends to infinity) renewal processes instead. Intuitively, observe that for the low popularity objects, those from popularity rank $x, x \to \infty$, which are the main constituents of the miss traffic, the time-spacing effect induced by the cache is indistinguishable from their intrinsic scarcity, leading to independent renewal intervals.

The correlation within the miss traffic makes the analysis of network of caches much harder than the analysis of Jackson queuing networks for example, the latter benefiting, thanks to Jackson's theorem from Poisson arrivals at every queue. Nevertheless, on dense networks, the aggregation of miss streams coming from a large number of caches might recover some Poisson characteristics.

Chapter 2

LAC/LAC+: Latency-Aware Caching Strategies in ICN

Can delivery in ICN be faster ?

Summary. *5G has loudly ambitioned to achieve extremely low latency in mobile networks. To this aim, we have recently introduced two novel latency-aware caching heuristics, LAC and LAC+ and we showed through simulations in Information-Centric Networks their good performance figures. In this chapter, we present an insight on their operations: a mathematical analysis of these caching systems led us to novel results that we validate in simulation. The advantages of these algorithms come (i) on one side from the fact they are distributed and lightweight and (ii) from the ability to quickly adapt to content popularity and network congestion, with no signaling nor explicit coordination between the network nodes. In this chapter we provide analytical bounds of latency aware caching policies and evaluate their performance by network simulations. The proposed mechanisms can halve the mean and standard deviation of content delivery time with respect to approximations of LFU as leave a copy probabilistically.*

Keyword: *Information-Centric Networks; stochastic modeling; caching.*

Contents

2.1 Introduction

Latency reduction objectives, currently emphasized as 5G requirements, imply to solve a number of technical challenges requiring novel solutions in the whole communication network: the physical layer, the MAC as well as the network backhaul and core. In this chapter we focus on one specific aspect of the communication path, *i.e.*, caching systems.

Information-Centric Networking (ICN) is a network architecture that embeds caching functions natively. We focus on the Named-Data Networking architecture (NDN) and we summarize its characteristics here. A detailed description of the system can be found in [Zhang et al., 2014].

Users retrieve named Data using a pull flow control protocol based on subsequent packet queries, triggering Data packets delivery. Name-based routing and forwarding guarantee that queries are properly routed towards a repository, where a permanent copy of the content is stored, following one or multiple paths. Network nodes maintain three major data structures: Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB). The CS caches Data packets received, which can be potentially useful to satisfy future Interest packets. The PIT stores Interests that have been forwarded and waiting for matching Data packets to return. The FIB is similar to IP routing table and is maintained by a name-based routing protocol. A strategy module defines the policy for output interface(s) selection at each FIB entry. For each arriving Data packet, a router finds the entry in the PIT that matches the data name and forwards the data to all downstream interfaces listed in the PIT entry. It then removes that PIT entry, and caches the Data in the CS. Indeed, Data may come from the repository, or from any intermediate cache along the path with a temporary copy of the Data packet. Packets of the same content can therefore be retrieved in a multi-path fashion. This means that data packet follow the reverse path build by the queries. This allows fine-grained monitoring of the response delay at any intermediate node in the communication path.

Intrinsic to ICN design are content authentication, multipath forwarding, affordable multicast and ubiquitous caching. Caches must store some valuable objects and evict some others by optimizing some objective in a scalable and fast way. Several content eviction policies have been designed in many technical areas such as First-In-First-Out (FIFO), Random, *(evict the)* Least-Frequently-Used (LFU) and several *(evict the)* Least-Recently-Used (LRU) derivatives like ARC [Megiddo and Modha, 2003] and CAR

[Bansal and Modha, 2004]. A policy optimizes a specific objective and most maximize the average hit ratio.

In our work, we consider low network latency objectives that can be met with the help of two novel LRU derivatives, LAC and LAC+. LAC, and its improved version LAC+, are designed on the simple idea that upon content arrival, the larger the retrieval latency, the more favorable the caching decision. Latency includes processing, queuing, transmission and propagation delays.

In this chapter, we characterize their miss ratio. Then a quantitative evaluation to corroborate the analysis is provided by means of ICN simulations. Quite impressive results are highlighted in terms of latency reduction under the assumption of a simple, fully distributed approach that self-adapts to varying network conditions. More precisely, we show that our solution outperforms state-of-the-art proposals by achieving significant reduction of content delivery time mean value and standard deviation up to 50%, along with a very fast convergence to these figures.

2.2 Related work

There is a huge literature on caching systems as a means to accelerate the data path of computing systems like ARC [Megiddo and Modha, 2003] and CAR [Bansal and Modha, 2004]. Some more recent literature has considered web caching to reduce access latency and scale content distribution [Laoutaris et al., 2004]. In the present work we focus on ICN systems like Named-Data Networking [Zhang et al., 2014] that embeds caching in the data plane of a network layer where congestion and latency are experienced by the transport protocol managing content retrieval. In particular we focus on algorithms that are distributed and lightweight so as to have a feasible implementation at high speed.

Among the fastest approximations of LFU we cite Leave a Copy Probabilistically (LCP) that keeps an object in a node's cache in a data path with probability p. [Bianchi et al., 2013] analyzes the p-LRU (LCP + LRU) replacement policy under renewal traffic. The probability p, of keeping an object in the cache after retrieval, is a positive constant smaller than 1. Our contribution extends LCP to dynamic cases, which simply refers to algorithms where p follows a stochastic process.

Much closer to our work, [Jelenković and Radovanović, 2004] derives through mathematical arguments a dynamically randomized heuristic for LRU caches. The objective is

to optimize the storage of variable size documents. A common implementation of LRU, referred to as Move-To-Front algorithm (MTF) consists in moving the most recently used object to the front of a FIFO memory. Though they randomized that MTF rule according to the document size and retrieval cost, they kept it *symmetric*, *i.e.*, triggering it with the same probability for both hit and miss events. As they ended up providing mathematical justification for a mechanism previously proposed in [Starobinski and Tse, 2001], we refer to this implementation as Starobinski-Tse-Jelenković-Radovanović's (STJR). On the contrary, our approach may quoted by *asymmetric* as a MTF probability is only considered when a document is freshly inserted into the cache *i.e.*, in case of a miss event. If the document was already in the cache *i.e.*, in case of a hit event, the MTF rule is deterministic as it is applied with probability 1.

2.3 Latency-aware heuristics

In this work, we analyze two distributed algorithms, LAC [Carofiglio et al., 2015c] and LAC+ [Carofiglio et al., 2015b], that aim at minimizing the overall average delivery time in information-centric networks without any coordination among the caches and no signaling overhead. Note that fine grained latency measurements are available in ICN as requests sent across an interface pull down data from the same interface. Network-wide, this enables symmetric routing and latency measurement of the upstream network.

Both algorithms work in the following way: *When a client requests at time t a rank-k object, $k \in \mathcal{K}$, that object is either in a cache along the path and consequently returned to the requester, or that cache will download it, then insert it in its local storage with probability $p_k(t)$ or not, with a probability $1 - p_k(t)$ and finally return that object to the requester.* We refer to p_k as the decision probabilities.

In LAC, the probability of inserting the rank-k object into the cache at time t is:

$$p_k(t) \equiv \min\left(\epsilon \frac{T_k(t)^\beta}{\bar{T}(t)^\gamma}, 1\right). \tag{2.1}$$

In LAC+, the decision probability $p_k^+(t)$ combines two terms:

$$p_k^+(t) \equiv p_k(t) + (1 - p_k(t))\Theta_k(t) \tag{2.2}$$

where $T_k(t)$ refers to the monitored latency for content k up to time t and $\bar{T}_k(t)$, $\bar{T}(t)$ to respectively the temporal average for content k and for all cached contents computed up to time t. Averages are estimated using Exponential Moving Average (EMA) filters. We satisfactory configured the weight of filters past values to 0.9.

ϵ is a small positive real number. β and γ are intensity parameters used in LAC to cleave probabilities between low and high latency retrievals. Higher latency objects will be picked early. Low latencies will get very small decision probabilities but should be eventually picked if the object is popular. For LAC+, since it has a separated latency outlier tracking function $\Theta_k(\cdot)$, we usually set β and γ to 1.

Let μ_t and σ_t be the average and standard deviation of all $\bar{T}_i(t), \forall i \in \mathcal{K}$, at a given node. We define the z^{th} quantile as follows:

$$Q_z(t) = \mu_t + z\sigma_t. \tag{2.3}$$

This allows to unfold p_k^+ second term. $\Theta_k(t)$ is the quantitative indicator at time t that the rank-k object is a latency outlier:

$$\Theta_k(t) \equiv \max\left(\frac{\bar{T}_k(t) - Q_z(t)}{\sup_{i\in\mathcal{K}}\{\bar{T}_i(t)\} - Q_z(t)}, 0\right). \tag{2.4}$$

We satisfactorily use the first quantile ($z = 1$) throughout the rest of the chapter.

To wrap up, LAC+ draws into the cache highly popular objects sampled using $p_k^+(t)$first term or outliers thanks to $p_k^+(t)$ second term.

2.4 Analysis

The dynamics of the network system are complex to capture in a simple model due to the tight coupling between delivery performance and caching functions: the former is certainly affected by network conditions, while clearly the network load is a result of caching performance and vice-versa. This is why we focused on the single cache case in developing analytically some performance bounds expressed in terms of the cache miss ratio. In a nutshell, we contribute in showing that the asymmetric design embodied by LAC and LAC+ outperforms previously known mechanisms, typically STJR. LAC and LAC+ are of the $asym$-LRU kind as opposed to alternative systems where insertion/re-

placement operations are symmetrically driven by the same probability (sym-LRU). LCP is a special case of asymmetric mechanism where the insertion into the cache is determined by a constant probability p. Refer to Table 2.1 for the notation used throughout the chapter. Variables might be later tagged with the current algorithm in superscript.

t	Instant a retrieval occurs.
x	Local cache size in number of objects.
τ_x	Characteristic time threshold for filling a cache of size x.
λ	Total request rate.
λ_k	Request rate of the rank-k object, $k \in \mathcal{K}$.
q_k	Popularity of the rank-k object. $q_k = \lambda_k/\lambda$.
$\varphi_{k,\tau}$	Probability of receiving at least one request for the rank-k object during τ seconds.
M_k	Asymptotic miss ratio for the rank-k object.
$\{\mathrm{VRTT}_{k,t}\}_{t\geq 0}$	Stochastic process modeling the retrieval latency of the rank-k object.
$\{p_{k,t}\}_{t\geq 0}$	Caching decision process of the rank-k object.
p	Random caching probability such that $p \overset{d}{=} p_{k,t}, \forall k, t$ under i.i.d. assumption.
$\{\pi_{k,t}\}_{t\geq 0}$	Miss probability process for the rank-k object.
$\{\mathcal{M}_{k,t}\}_{t\geq 0}$	Miss counting process for the rank-k object. It is expected to increase every $1/m_k$ cycle with $m_k = \mathbb{E}[\pi_{k,t}]$.

Table 2.1 – Notation.

2.4.1 Assumptions

We consider the smallest set of assumptions to have a simple and feasible analytic representation.

- *Zipf-like popularity:* We assume that object popularity follows a generalized Zipf law. Thus $\forall k \in \mathcal{K}, \quad q_k = ck^{-\alpha}$ with $1/c = \sum_{i\in\mathcal{K}} i^{-\alpha}$ and skewness $\alpha > 0$. This assumption is widely accepted in the literature [Breslau et al., 1999; Mitra et al., 2011].

- *Poisson requests:* We assume that clients request objects according to a Poisson process of intensity $\lambda > 0$, similarly to [Carofiglio et al., 2013b; Badov et al., 2014].

- *Independent Reference Model:* Temporal correlation between object requests are neglected here like in [Starobinski and Tse, 2001] and [Fricker et al., 2012]. It is nevertheless foreseen in future extensions of this work.

- *LRU replacement policy:* we focus on the widely adopted LRU replacement policy whose common implementation consists in moving the most recently served object to the front of a list. This allows to study Move-To-Front algorithm as an LRU scheme [Jelenković and Radovanović, 2004].
- *Same object size:* For the sake of simplicity, we assume that, like in [Che et al., 2006], all retrieved objects have the same size. The model will later be improved to encompass more fine-grained features such as variable object size.
- $\text{VRTT}_{k,t}, \forall t \geq 0$ are strictly positive, independent and identically distributed (i.i.d.).
- $p_{k,t} \in]0,1], \forall k, t \geq 0$ are also i.i.d. Hence $\exists p \overset{d}{=} p_{k,t}$ and $\pi_{k,t}, \forall t$ are also i.i.d..
- The characteristic time ("Che's") approximation [Che et al., 2006] as extended by [Fricker et al., 2012] is a key tool in this work. It states that for LRU caches, the object eviction time is well approximated by a unique constant τ_x.

2.4.2 Miss ratio

Let $\pi_{k,t}$ be the rank-k object miss probability at time t and $\varphi_{k,\tau}$ be the probability of receiving at least one request for a rank-k object during τ seconds.

Proposition 2.1. *If we restrict to a countable set of caching decision probabilities, and assuming Che's approximation holds, the miss ratio M_k^{asym} of asymmetric algorithms such as LAC and LAC+ for the rank-k object is:*

$$M_k^{asym} = \sum_u \mathbb{P}[p = u] \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1-u)}, \tag{2.5}$$

$$\tau_x \text{ being the root of } \sum_{k \in \mathcal{K}} (1 - M_k^{asym}) = x. \tag{2.6}$$

Proof. First, we characterize the arrival process to the front of the LRU cache. Given the random insertion probability p, the Move-to-Front probability during the time window τ starting at t, for object k equals:

$$\begin{aligned} F_{k,t}(\tau) &= \left((1 - \pi_{k,t}) + \pi_{k,t} p\right) \varphi_{k,\tau} \\ &= \left(1 - (1-p)\pi_{k,t}\right) \varphi_{k,\tau}. \end{aligned}$$

That Move-to-Front probability leads to the cache miss probability in the following way. Under Che's approximation, the rank-k object miss probability for a cache under stochastic caching decision satisfies:

$$\begin{aligned} F_{k,t}(\tau_x) &= 1 - \mathbb{P}_{k,t}[\sharp MTF > x] \\ &= (1 - (1-p)\pi_{k,t})\varphi_{k,\tau_x}. \end{aligned}$$

$\sharp MTF$ denotes the number of distinct objects moved to the cache front. Also,

$$F_{k,t}(\tau_x) = 1 - \pi_{k,t}.$$

Hence,

$$\pi_{k,t} = \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1-p)}, \tag{2.7}$$

and

$$\begin{aligned} \mathbb{E}[\pi_{k,t}] &= \int_{[0,1]} \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1-u)} d\mathbb{P}[p \leq u] \\ &\equiv m_k. \end{aligned} \tag{2.8}$$

$\mathbb{E}[\pi_{k,t}]$ is the expectation of the random miss probability for content k. However, only the miss ratio M_k *i.e.*, long-term average of a miss counter $\mathcal{M}_{k,t}$ can be effectively measured. Since the cache is always supposed much smaller than the catalog, $1/m_k$ is finite. Thus, the elementary renewal theorem holds and the asymptotic miss ratio for the rank-k object writes:

$$M_k \equiv \lim_{t \to \infty} \frac{1}{t} \mathcal{M}_{k,t} = m_k \ a.s. \tag{2.9}$$

□

Note that $\varphi_{k,\tau_x} \equiv 1 - e^{-\lambda_k \tau_x}$ under Poisson object arrivals.

Accounting for all values of p in Eq.(2.6) might not be computationally tractable. The following proposition shows that all values of p can be effectively replaced by its expected value.

Proposition 2.2. *Assuming Poisson request arrivals, the cache miss ratio* M_k^{asym}

is well approximated using the expected value $\mathbb{E}[p]$ of a unique decision probability p when $\mathbb{E}[p]$ is very small or when the object popularity is very small or the cache is very large:

$$M_k^{asym} \approx \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1 - \mathbb{E}[p])}. \tag{2.10}$$

This result is important as it establishes achievable conditions for asymptotic equivalence between the use of a variable decision probability p and the use of its expected value $\overline{p} = \mathbb{E}[p]$. However, the operational drawback of a constant and small $\overline{p}$ is that it postpones considerably the time popular objects are first stored in the cache. LCP suffers from this phenomenon because the expected time to enter the cache is $\frac{1}{\lambda_k \overline{p}}$. Consequently, LCP overall object delivery time converges slowly. LAC+ brings a solution in adequately varying p in order to cache valuable objects earlier.

Proof. We define the function f_k of a couple of real variables u, v such that

$$f_k(u, v) = \frac{e^{-\lambda_k \tau_x(v)}}{1 - (1 - e^{-\lambda_k \tau_x(v)})(1 - u)}. \tag{2.11}$$

f_k is content k miss probability given a cache characteristic time $\tau_x(v)$ and decision probabilities u and v. It is a convex function of u as

$$\frac{\partial^2}{\partial u^2} f_k(u, v) = \frac{2\left(e^{\lambda_k \tau_x(v)} - 1\right)^2}{\left(1 + \left(e^{\lambda_k \tau_x(v)} - 1\right)u\right)^3} \geq 0. \tag{2.12}$$

Hence, by Jensen's inequality,

$$\mathbb{E}[f_k(p, p)] \geq f_k(\mathbb{E}[p], p). \tag{2.13}$$

Let define $\mathcal{D}_k(x, p) \equiv \mathbb{E}[f_k(p, p)] - f_k(\mathbb{E}[p], p)$.

As p is a strictly positive random variable, Markov's inequality holds and $p \xrightarrow{p} 0$

as $\mathbb{E}[p] \downarrow 0$. Therefore, $\lim_{\mathbb{E}[p]\to 0} \mathcal{D}_k(x,p) = 0$, as

$$\begin{aligned}
\lim_{p \xrightarrow{p} 0} \mathbb{E}[f_k(p,p)] &= \lim_{p \xrightarrow{p} 0} \int_{[0,1]} f_k(u,p) d\mathbb{P}[p \leq u] \\
&= \int_{[0,1]} \lim_{p \xrightarrow{p} 0} f_k(u,p) d\delta_0(u) \text{ by dominated convergence,} \\
&\quad \text{where } \delta_\cdot \text{ is the Dirac measure} \\
&= \lim_{p \xrightarrow{p} 0} f_k(0,p) \\
&= \lim_{\mathbb{E}[p]\to 0} f_k(\mathbb{E}[p],p) \text{ by continuity.}
\end{aligned}$$

To conclude this first part, as $\mathbb{E}[p] \downarrow 0$, Che's approximation yields

$$\sum_k f_k(\mathbb{E}[p],p) = x = \sum_k f_k(\mathbb{E}[p],\mathbb{E}[p]), \tag{2.14}$$

making $\mathbb{E}[f_k(p,p)] = f_k(\mathbb{E}[p],\mathbb{E}[p])$ as $\mathbb{E}[p] \downarrow 0$ follow.

Secondly, as $x \uparrow \infty$ implies that $M_k = 0, \forall k$ and $\tau_x \uparrow \infty$,

$$\lim_{x\to\infty} \mathcal{D}_k(x,p) = 0. \tag{2.15}$$

Also,

$$\lim_{\lambda_k \to 0} D_k(x,p) = 0. \tag{2.16}$$

Using the fact that $\mathcal{D}_k$ is differentiable on its domain and non-negative, it gets minimal as either $\mathbb{E}[p] \downarrow 0$ or $x \uparrow \infty$ or $\lambda_k \downarrow 0$. It means that Eq.(2.10)'s underestimation of rank-k content miss ratio shrinks under these conditions. □

2.4.3 Lower bound

Providing a closed-form approximation for $asym$-LRU miss ratio and its characteristic time τ_x^{asym} is hard. Instead, we demonstrate its superiority over the analytically tractable sym-LRU mechanism. With some loss of generality, α, the exponent of the Zipf law, also referred to as its skewness parameter, is assumed greater than one. Let us consider

the symmetric mechanism sym-LRU where the MTF rules are conditioned by the same probability in both hit and miss cases. By contrast in $asym$-LRU the MTF decision is taken in case of miss only.

Proposition 2.3. *Assuming $VRTT_{k,t}, \forall k, t$ are i.i.d. and large catalog and cache, the steady-state miss probability of symmetric LRU algorithms, for the rank-k object, approximates to:*

$$M_k^{sym} = \exp\left\{-\frac{x^\alpha}{k^\alpha \Gamma(1-\frac{1}{\alpha})^\alpha}\right\}, \tag{2.17}$$

where $\Gamma(\cdot)$ is the Gamma function.

Proof. Let $\overline{p} = \mathbb{E}[p_{k,t}], \forall k \in \mathcal{K}$ at steady state.

Let $\sharp k$ denote the number of times a rank-k object is moved to the cache front during a time interval. The mean number of distinct objects moved to the front of the LRU cache during τ, as $|\mathcal{K}| \uparrow \infty$ and $\tau \uparrow \infty$, is:

$$\sum_k \mathbb{E}\left[\mathbb{1}_{\{\sharp k>0\}}\right] = \sum_k (1 - e^{-\lambda_k \tau \overline{p}}) \sim (\lambda \tau c \overline{p})^{\frac{1}{\alpha}} \Gamma\left(1 - \frac{1}{\alpha}\right) \tag{2.18}$$

in virtue of Lemma 5 of [Jelenković and Radovanović, 2004]. Hence, the power of α-magnified mean number of distinct objects moved to the front of the LRU cache during characteristic time τ_x^{sym}:

$$x^\alpha = \lambda \tau_x^{sym} c \overline{p} \Gamma\left(1 - \frac{1}{\alpha}\right)^\alpha \Rightarrow \tau_x^{sym} = x^\alpha (\lambda c \overline{p})^{-1} \Gamma\left(1 - \frac{1}{\alpha}\right)^{-\alpha}. \tag{2.19}$$

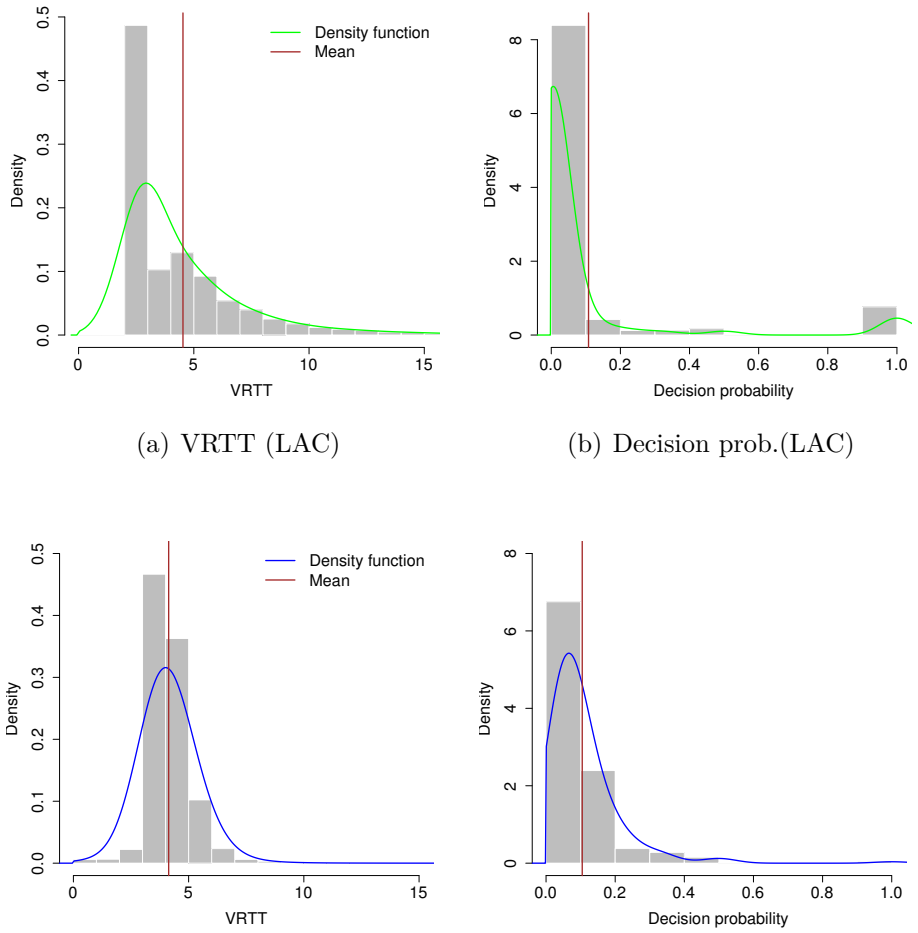
Recall that c is the normalization constant of the Zipf distribution. The rest follows by using the exponential inter-arrival distribution for an object with rank k. □

The closed-form expression of Proposition 2.3 is intrinsically the same as LRU's in [Carofiglio et al., 2013b]. This observation yields the next corollary.

Corollary 2.3.1. *Assuming $VRTT_{k,t}, \forall k, t$ are i.i.d.,*

$$M_k^{sym} = M_k^{LRU}$$

i.e., *sym-LRU behaves in stationary regime like LRU.*



(a) VRTT (LAC) (b) Decision prob.(LAC)

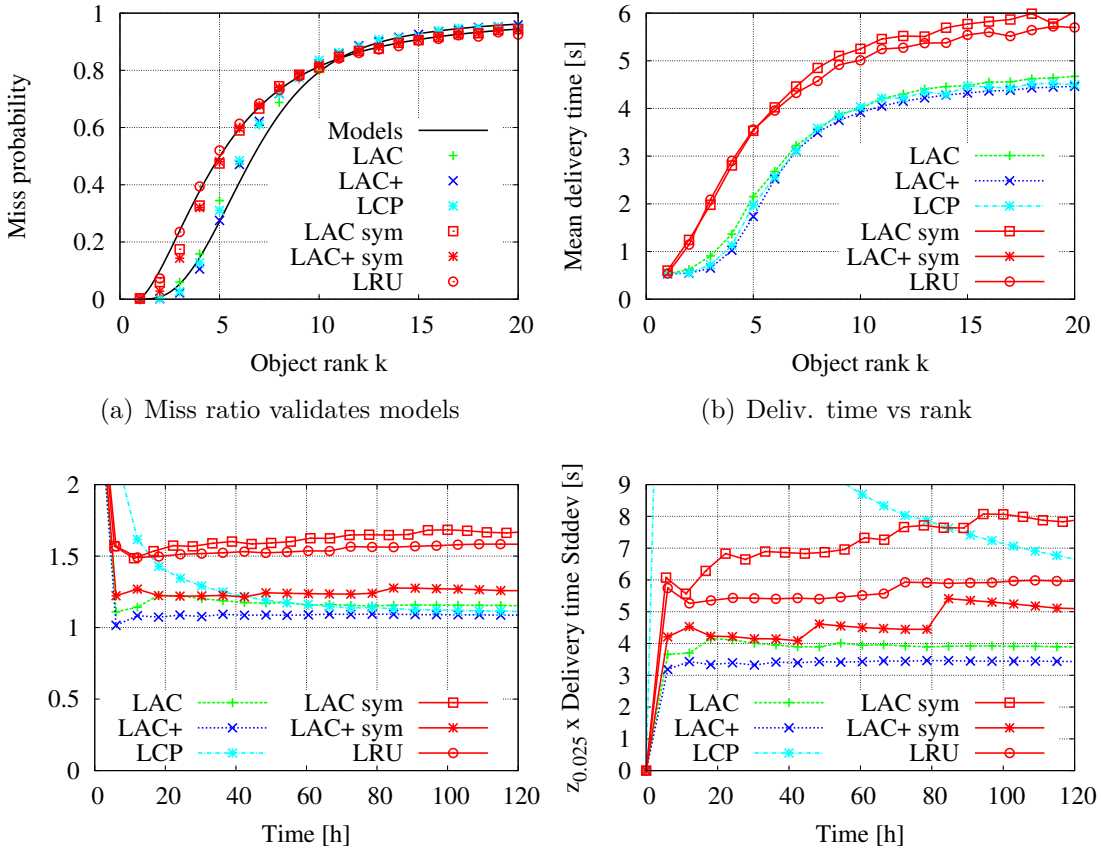
(c) VRTT (LAC+) (d) Decision prob.(LAC+)

Figure 2.1 – Single cache: latency and decision probability distributions.

asym-LRU consequently outperforms *sym*-LRU thanks to its convergence to the Least Frequently Used replacement policy [Martina et al., 2013]. This leads to Proposition 2.4 which relies on ϵ-permanent accommodation, a notion to be introduced first.

Definition 2.1. *An object is ϵ-permanently accommodated if its miss ratio is less than a small value ϵ.*

In that context, let $\eta_{mechanism}$ be the number of most popular objects ϵ- permanently accommodated thanks to a caching mechanism.



(a) Miss ratio validates models

(b) Deliv. time vs rank

(c) Mean deliv.time vs time

(d) Deliv.time stddev vs time

Figure 2.2 – Single cache: LAC and LAC+ decrease LRU delivery time by 30% and outperform LCP on convergence.

Proposition 2.4. *As decision probability's expected value goes small, asym-LRU allows to accommodate ϵ- permanently more of the most popular objects than sym-LRU* i.e.*,*

$$\eta_{asym} \geq \eta_{sym}.$$

Proof. Let the miss ratio of all permanently stored objects admit a sufficiently small

value ϵ as upper bound. Then:

$$\eta_{asym} = \left[\frac{\lambda c \tau_x^{asym}}{\log\left(1 + \frac{1}{\mathbb{E}[p]}\left(\frac{1}{\epsilon} - 1\right)\right)} \right]^{\frac{1}{\alpha}} \text{ and} \tag{2.20}$$

$$\eta_{sym} = \frac{x}{\Gamma(1 - \frac{1}{\alpha})(-\log\epsilon)^{\frac{1}{\alpha}}}. \tag{2.21}$$

Since a first-order Taylor series expansion of ϵ for $asym$-LRU, when $\mathbb{E}[p] \downarrow 0$, yields:

$$\eta_{asym} \underset{\mathbb{E}[p]\to 0}{\sim} x \left[\Gamma\left(1 - \frac{1}{\alpha}\right) \mathbb{E}[p] \log\left(1 + \frac{1}{\mathbb{E}[p]}\left(\frac{1}{\epsilon} - 1\right)\right)^{\frac{1}{\alpha}} \right]^{-1}, \tag{2.22}$$

we have

$$\lim_{\mathbb{E}[p]\to 0} \frac{\eta_{asym}}{\eta_{sym}} \geq (-\log\epsilon)^{\frac{1}{\alpha}} > 1. \tag{2.23}$$

□

Let $LAasym$ denote LRU equipped for asymmetric latency-aware stochastic caching decision (LAC and LAC+) and let $LAsym$ denote LRU modified for symmetric latency-aware stochastic MTF decision (STJR).

Corollary 2.4.1. *As decision probability's expected value goes small,*

$$\exists \kappa \geq 1 : \eta_{LAasym} \geq \kappa \eta_{LAsym}. \tag{2.24}$$

This typically means that the performance of LRU caches equipped with LAC or LAC+ can exceed beyond a given factor κ that of sym-LRU, then LRU studied analytically and extensively in previous works [Carofiglio et al., 2013b]. Numerous simulations backed these mathematical results, where often $\kappa > 2$ unleashes tremendous content delivery time decreases.

2.5 Simulation

We evaluate LAC and LAC+ against three state-of-the-art caching management mechanisms: LRU + Leave-Copy-Everywhere (LRU), LRU + Leave-Copy-Probabilistically (LCP) and LRU + Leave-Copy-Down (LCD)[Laoutaris et al., 2004]. This is carried out by means of the packet-level NDN simulator CCNPL-Sim (the code of the simulator as well as the input files to run the scenarios presented in this chapter can be found at `http://systemx.enst.fr/ccnpl-sim`) (i) on a single cache topology, (ii) then on a complex network where core caches are located along a ring. While in (i) the workload is IRM and Zipf skewness $\alpha > 1$, in (ii) we injected some time locality and set $\alpha < 1$ to investigate situations closer to the real world.

Single cache topology

The following results are achieved in a simulated ICN with a single caching node between the object consumers and the publishing server. The whole simulation setup is available online. Here are the main configuration parameters. Cache sizes are equal to 80kB. The Poisson process for generating content requests is characterized by a rate of 1 object/s. Objects are requested over a catalog of $20,000$ items, according to a Zipf-like popularity distribution of parameter $\alpha = 1.7$. This value of α is still realistic [Mitra et al., 2011]. The two FIFO links from the consumers up to the content publisher have a capacity of 200Kbps and of 30Kbps, respectively. The size of every object conveyed through these links is 10kB, that we also take as fixed packet size. About LAC parameters, $\epsilon = 1$ while $\beta = \gamma = 4.5$ to pick latency outliers and leave quickly delivered objects to popularity sampling. The function f is the mean latency of all ever-cached objects. LAC+ is configured with $\epsilon = 0.05$ and $\beta = \gamma = 1$, relying on its adjunct outlier tracking function $\Theta_k(\cdot)$. We report the simulation results in Fig.2.2.

First, it appears clearly in Fig.2.1 that the decision probability values are predominantly small. The mean decision probability equals 0.1 for both LAC and LAC+. This is what drives their joint popularity sampling / latency screening capabilities. Secondly, we can observe from the plots in Fig.2.2 that LAC and LAC+ converge to the same steady state as LCP, which approximates the optimal LFU behavior. LCP, LAC and LAC+ miss probabilities coincide even though to the exception of the former, they are based on temporal measurements of residual latency, so adapting over time based on the sensed varia-

tions in terms of experienced latency. Thirdly, we observe how much LAC/LAC+ latency-aware technique reduces both delivery time mean and standard deviation. It is striking to see how quickly they converge, compared to classical LCP. Observe that LAC+ is so efficient that, even in its symmetric implementation, it captured early the highest popular content and made the delivery time drop. Conversely, the constant decision probability used in LCP is the average of all latency-aware decision probabilities ($p = 0.1$) and this negatively impacts the convergence and the system reactivity to temporal variations of latency, as opposed to our LAC and LAC+ proposals. Finally, we observe that $LAsym$ and LRU miss ratio curves coincide in steady state as predicted in [Jelenković and Radovanović, 2004]. A symmetric filtering of objects to put in and to move to the cache front has the only effect of slowing down convergence while not modifying the dynamics of the underlying Markov chain.

Line topology network

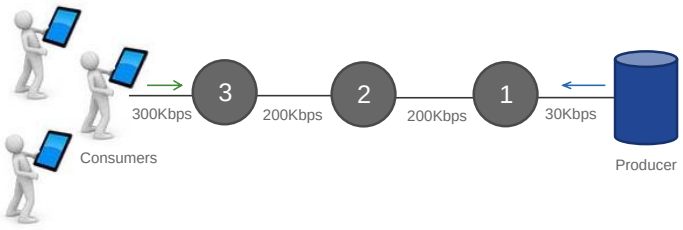
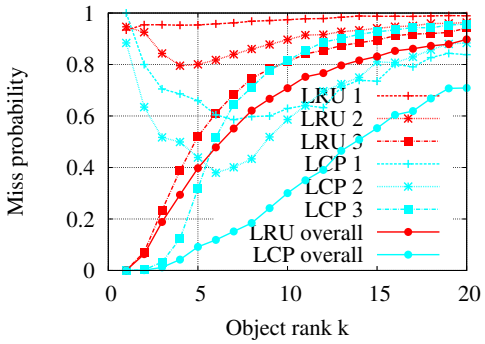


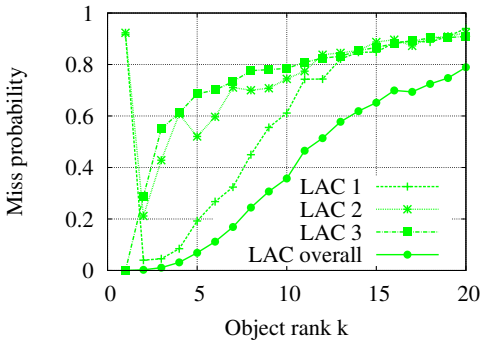
Figure 2.3 – Simulated line topology.

First, we consider the setting in Fig. 2.3, with three caching nodes in-line between the users and the publishing server. The four links from the consumers up to the publisher have capacities equal to 300Kbps, 200Kbps, 200Kbps and 30Kbps respectively. Cache sizes are equal to 80kB. Each object has an average size of 10kB, that we also take as fixed packet size. LCP is parametrized with the probability $p = 0.1$ and corresponds to the lowest mean latency-aware caching decision probability, Cache 3's. We configure LAC with $\beta = \gamma = 5$ to stress the rejection of quickly delivered objects. Related results are reported in Fig. 2.4. The resulting link load ρ on downlinks from the repository to the

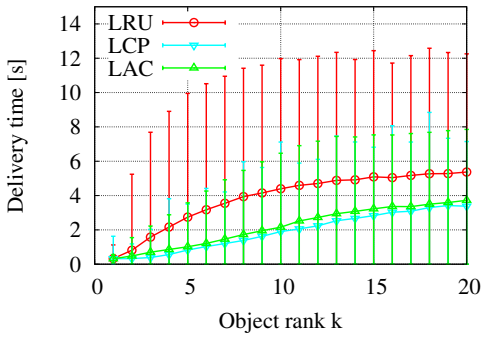
users is respectively : (0.5, 0.01, 0.03, 0.27) under LRU, (0.27, 0.02, 0.02, 0.27) under LCP and (0.22, 0.04, 0.06, 0.27) under LAC.



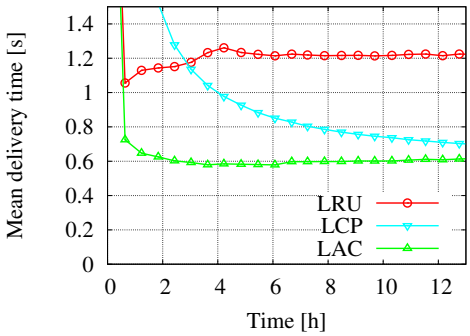
(a) LRU and LCP miss probability



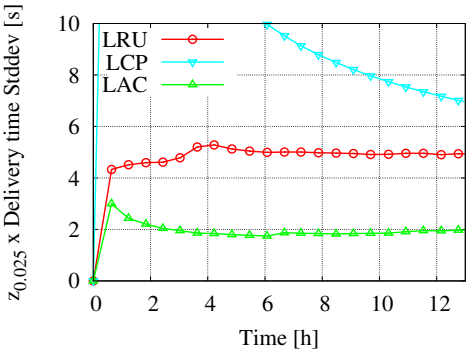
(b) LAC miss probability



(c) Delivery time vs content rank



(d) Evolution of the mean delivery time



(e) Evolution of the delivery time standard deviation

Figure 2.4 – Line topology simulation: LAC decreases LRU delivery time by 50% and outperforms LCP on convergence.

Clearly, the expensive traffic to the publisher decreases significantly with LAC, while

very little increase can be observed on the other links. The tremendous gain in delivery time (50% of LRU's) can be appreciated in both its first and second moments. Such a delivery time standard deviation decrease plays a central role in stabilizing customers quality of experience.

Tree topology network

The next results are those achieved in the ICN setting in Fig. 2.5, spanning a binary tree topology whose seven caching nodes are spread over three network levels, between the users and the repository (publishing server). In this configuration, cache sizes are 8MB. Object size is taken equal to 1MB. Downlink capacities from the users up to the repository are 30Mbps-capable, except the last one toward the repository, which is 9Mbps. Each packet has an average size of 10kB, making every object equal to 100 packets in size. Caches are equipped for LAC decision, with $\beta = \gamma = 3$. Cache 4 is on the first layer (the closest to the consumers), Cache 8 on the second layer and Cache 10 on the third (the farthest to the users). LCP's $p = 0.03$. That corresponds to LAC's mean latency-aware caching decision probability. We report the related charts in Fig. 2.6.

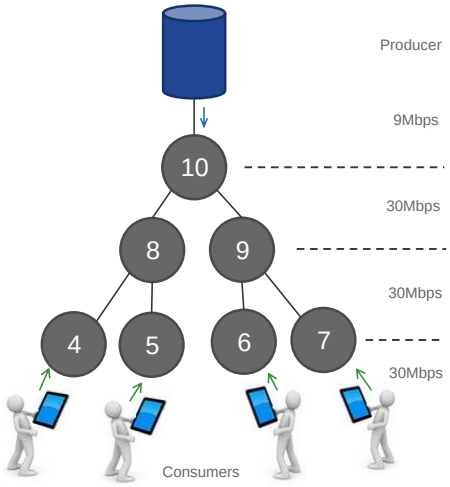
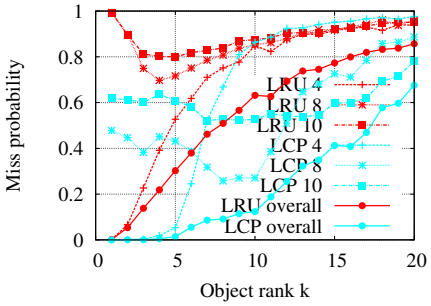
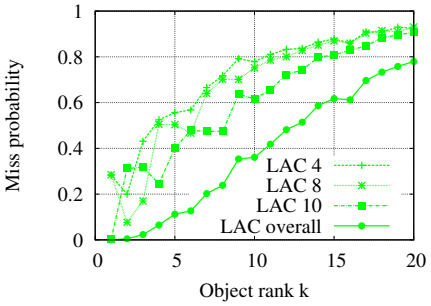


Figure 2.5 – Simulated tree topology

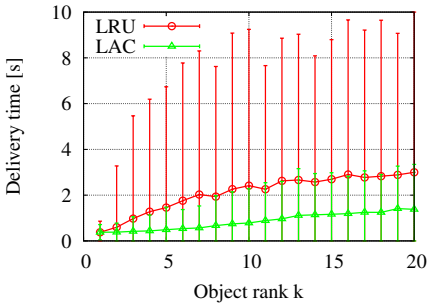
The observed link load ρ on downlinks from the repository to the users is respectively: (0.7, 0.31, 0.18, 0.6) under LRU, 0.7, 0.07, 0.33, 0.6) under LCP and (0.7, 0.12, 0.23, 0.6) under LAC. Again, our LAC mechanism allows to lower maximum and average link load



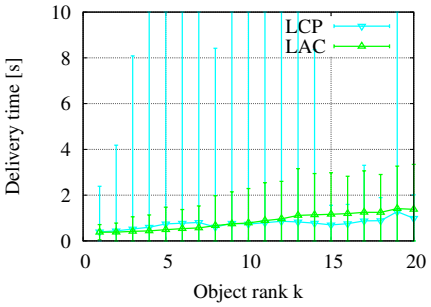
(a) LRU and LCP miss probability



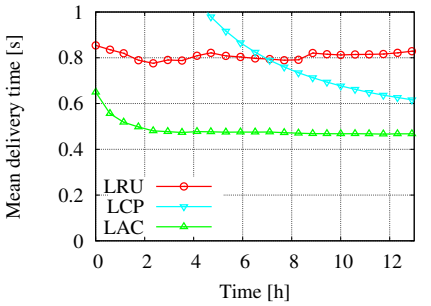
(b) LAC miss probability



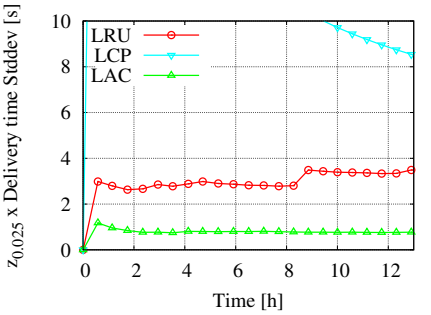
(c) Delivery time vs content rank (vs LRU)



(d) Delivery time vs content rank (vs LCP)



(e) Evolution of the mean delivery time



(f) Evolution of the delivery time standard deviation

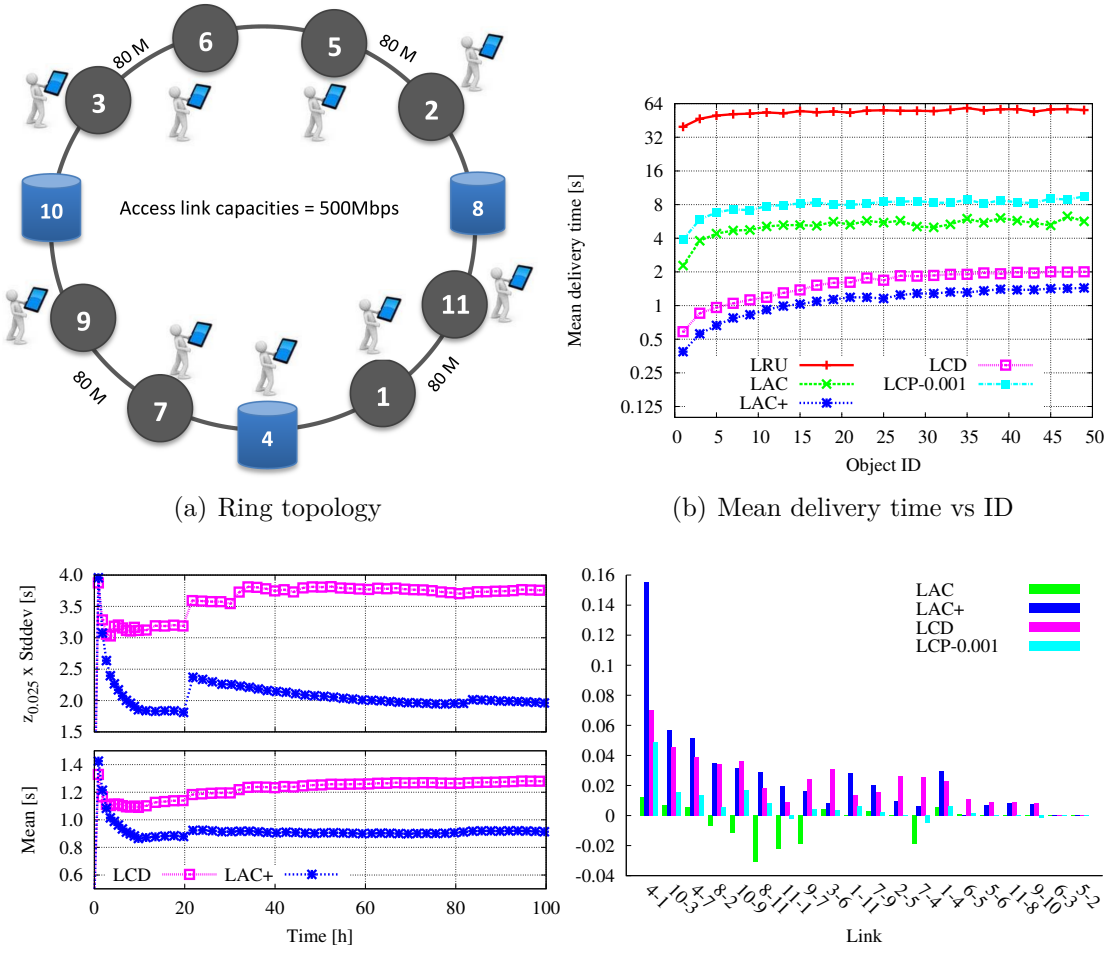
Figure 2.6 – Tree topology simulation: LAC decreases LRU delivery time by 30% and outperforms LCP on convergence.

over the network. So, even though LAC reduces by half LRU's load between layer 3 and layer 2 caches, it still relies on caching delegation, which implies some inter-cache traffic.

Finally, we observe as a general rule that implementing LAC decreases the overall cache miss probability *i.e.*, the probability that all solicited caches fail to serve the re-

quested object. It also decreases and stabilizes the overall object delivery time. Indeed, the mean delivery time and the 95% confidence interval around the average, both decrease by up to 50%. Note also that this overall improvement is not achieved to the detriment of the convergence speed, unlike LCP. The latter, indeed, exhibits tremendously slow convergence and extremely high delivery time standard deviation.

Ring topology



(a) Ring topology

(b) Mean delivery time vs ID

(c) Delivery time vs Time

(d) Load reduction × load_{LRU}

Figure 2.7 – Ring topology with non-stationary workload

In this section, we evaluate the consistency of our algorithms in a network scenario where eleven interconnected core nodes form a ring. Every link at the network core offers

a 80Mbps capacity. Among the core nodes, three are content producers. Each offers $20,000$ Zipf-ranked objects from its own catalog. Objects from the producer at Node 4 are given the prefix */Netflix/*, */Orange/* for those from Node 8 and */Youtube/* for those originating from Node 10. Each object is conveyed in chunks of 3kB and has a total size of 2MB. Each node cache can accomodate up to 40 objects. While the skewness of the Zipf-like popularity distribution, α, remains 0.9 for the whole simulation, we inject some time locality in shuffling every object rank every ten hours. Clients connect to their closest core node to send interests and retrieve data over dedicated 500Mbps links. Client requests reaching every core node follow a Poisson process with intensity $\lambda = 2$ objects/s. Clients are equally interested in every catalog, so that any of them addresses every catalog with probability $1/3$. Routing is single path. Fig.2.7(a) depicts the network setup. LAC+ and ϵ-LCP share a common value of $\epsilon = 0.001$. LAC exploits a different $\epsilon = 1$ but keeps intensity parameters β and γ equal to 1.

The striking results in Fig.2.7(b) show LAC+ decreasing LRU mean delivery time by up to two orders of magnitude. LAC+ clearly outperforms LAC and LCP in ensuring a content delivery at least twice faster. Moreover, as witnessed by Fig.2.2(c) and Fig.2.2(d), LRU equipped with the Leave-Copy-Down algorithm denoted by LCD is surpassed. LAC+ heuristic minimizes the highest link load in a dynamic way (Fig.2.7(d)). Thanks to its reactivity to congestion, LAC+ reduces by 30% the mean delivery time inducted by LCD and by 50% the related standard deviation.

2.6 Conclusion and future work

Throughout the chapter, we characterized, bounded and evaluated the performance of latency-aware LRU caches. The theoretical contribution extends the state-of-the-art of probabilistic caching analysis. The novel idea behind ubiquitous latency-awareness is simple, fully distributed and demonstrated powerful by means of extensive simulations. By fully distributed, we highlight the fact that latency-awareness blasting performance is free of any form of signaling. Actually, making early caching decisions based on the latency of retrieved objects will sound increasingly intuitive, especially in the forthcoming 5G era. The task of accurately modeling networks of such caches in order to capture their dynamics is still ongoing.

Chapter 3

FOCAL: Joint Forwarding and Caching with Latency-awareness in ICN

Can delivery in ICN be much faster ?

Summary. *Research on 5G has recently promoted latency minimization from a critical network optimization criterion to an architectural cornerstone. Information-Centric Networking (ICN) appears a promising candidate technology for building an agile communication model that reduces latency via a fully distributed and adaptive delivery approach coupling in-network caching and forwarding. In the chapter, we theoretically and empirically investigate the role of latency awareness on multi-path ICN delivery performance and analyze FOCAL, an approach combining novel caching and forwarding strategies to reduce end-user experienced latency without any network signaling nor coordination between routers.*

FOCAL gathers a latency-proportional probabilistic caching policy, with a load-aware dynamic forwarding strategy, that preferentially routes popular content requests through a single path (set of caches), while globally achieving minimum network load and user content delivery time, thus delay minimization. By means of ICN simulation, we assess the advantages of FOCAL over existing alternatives given by the combinations of known caching policies and forwarding strategies. FOCAL dras-

tically improves end-user delivery performance as it reduces by up to 60% *the mean and variance of content delivery time. It also results in a faster convergence to these figures, even under the varying network conditions induced by non-stationary content popularity distributions.*

Keywords: *Information-Centric Networking; Caching; Adaptative Forwarding; Network Performance Analysis.*

Contents

3.1 Introduction

If latency minimization is already an important traffic engineering criterion in current networks, it is anticipated to be a founding principle for the architectural design of 5G networks. An efficient orchestration of edge caching, wire-speed packet processing and traffic load-balancing appears essential to relieve congestion and to accommodate QoE (Quality of Experience) requirements of latency-sensitive applications.

Research on ICN has recently highlighted the benefits of content-centric over host-centric communication in terms efficient data delivery [Katsaros et al., 2014], but also of optimized use of network resources [Llorca et al., 2013] [Imbrenda et al., 2014], simplified management of mobility [Augé et al., 2015] and embedded security [Ion et al., 2013].

In ICN a tight coupling exists by definition between distributed forwarding and caching operations: the use of hop-by-hop dynamic forwarding determines the arrival process at in-network caches, where the persistence of content can be locally optimized. To close the loop, the resulting hit/miss performance affects link loads and forwarding decisions to achieve overall performance optimization. Therefore, latency reduction can be achieved, in ICN, via both in-network caching and hop-by-hop distributed forwarding. The goal of this chapter is to further investigate the impact of latency-awareness on caching decisions alone, then to study the interaction with different dynamic forwarding strategies and to propose a combined approach. Previous studies focused on the definition of ICN caching and forwarding strategies with the aim of minimizing a network cost function, very few of them considered latency. This chapter is, to the best of our knowledge, the first theoretical and empirical study of latency awareness on multipath ICN data delivery performance under dynamic bandwidth sharing and network congestion. Herein, latency encompasses processing, queuing, transmission and propagation delays.

The key elements to understand how latency-awareness improves caching are that: (i) it makes the cache focus on a smaller catalog subset, the few objects retrieved along a path whose bottleneck is upstream the node; (ii) it implicitly accelerates convergence towards a dynamic LFU as latency-aware caches would store new objects until congestion drop, which indicates that latency-generating objects, potentially the most popular, have just been cached. This is particularly interesting in case of dynamic catalog, non-stationary content popularity distribution or mobility. Schematically, latency-awareness improves

forwarding as it prevents from overloading congested routes.

Concretely, we recapitulate and extend the work started in an initial proposal of Latency-Aware Caching (LAC)[Carofiglio et al., 2015c], enhanced later to strengthen latency dependency in probabilistic caching decisions (LAC+)[Carofiglio et al., 2015a]. If the benefits brought by LAC+ were clear on a single cache or a system of caches working under random request forwarding, the interaction with smart forwarding strategies was the second step for the definition of FOCAL we introduced in [Carofiglio et al., 2015b]. We consider as starting point the optimal load-balancing (LB) solution derived in [Carofiglio et al., 2013c] to achieve load minimization by distributed and dynamic monitoring of the residual round trip time behind output interfaces. Intuitively, such a content-agnostic fine-granularity forwarding strategy does not help differentiate the arrival process at caches along different paths and hence realizes implicit latency-aware cache coordination. To this aim, we present and analyze a novel load-balancing strategy, *LB-Perf* that locally monitors the most popular content requests and persists in routing them through a single path, while applying the agnostic LB approach to the aggregate of less popular requests. In this chapter, we provide mathematical grounds for FOCAL, read the combination of LAC+ and LB-Perf. Its performance is evaluated by means of ICN simulation experiments in various parameter settings and network scenarios to show its robustness and the consistent benefits w.r.t existing alternatives. These consist in combinations of known caching policies (LRU, probabilistic caching, Leave-a-Copy-Down) and forwarding strategies (random, load-balancing, load-balancing with persistent forwarding). Promising results are obtained in terms of reduced end-user delivery time average and variance by up to 60%. Furthermore, we observed faster convergence to these figures, even under the varying network conditions induced by non-stationary content popularity distributions.

Our approach to Information-Centric Networking definitely owes to the Named-Data Networking (NDN) architecture, a fork of Content-Centric Networking (CCN) , in terms of data structures and work-flow. However, the genericity of our models and algorithms makes them pertain beyond the NDN/CCN realm, wherever caches operate and forwarding decisions must be made.

The remainder of the chapter is organized as follows.

Sec.3.2 describes related work. Sec.3.3 formalizes the problem. Sec.3.4 provides guidelines for optimal joint caching and forwarding algorithm design. They are grounded in a mathematical analysis of the optimum, complemented by the solutions a state-of-the-

art solver computed. In Sec.3.5, we derive the FOCAL heuristics and present latency-aware caching and forwarding strategies. In Sec.3.6 we provide a mathematical proof of the algorithm stability. The evaluation results are gathered in Sec.3.7. They essentially highlight, given the Remote Adaptive Active Queue Management (RAAQM) congestion/rate controller [Carofiglio et al., 2013c], the improvements in content delivery time mean and standard deviation w.r.t. popularity ranks and elapsed simulation time. Finally, Sec.3.8 concludes the chapter, complemented by number of appendices that provide proofs to lemma and mathematical propositions.

3.2 Related work

In the context of ICN research, we identify two categories of related work: proposals introducing enhancements of classical cache management policies for a given cost function and studies focusing on forwarding strategies to optimize request-to-cache routing.

Latency-aware caching

Within the panoply of cache management proposals, some leverage content placement (e.g. [Yu et al., 2015; Li and Simon, 2011]) while others deal with caching mechanisms based on selective insertion and replacement in cache (e.g. [Psaras et al., 2012; Badov et al., 2014; Ioannou and Weber, 2014; Ming et al., 2012]). The first class of approaches is appropriate for small-scale controlled environments like a CDN (Content Delivery Network), where topology and content catalog are known a priori. Either [Yu et al., 2015] and [Li and Simon, 2011] deals with video streaming in ICN and orchestrate caching and scheduling of requests to caches in order to create a cluster of caches with a number of guaranteed replicas. Unlike these approaches, our previous ([Carofiglio et al., 2015c]) and current work on latency-aware caching belong to the second class of caching solutions by defining a decentralized solution that automatically adapts to changes in content popularity, network variations etc. by leveraging content insertion in cache. We share the same objective as in [Badov et al., 2014], where authors propose a congestion-aware caching mechanism for ICN, based on estimation of local congestion, of popularity and of bottleneck position. Differently from our work, their congestion estimate does not differentiate content items in terms of latency. More recently, [Nguyen et al., 2015] proposed

a new caching policy that consists in solving a knapsack of congestion prices. That study assumed single-path content delivery. Similar considerations hold for other related approaches: the ProbCache work in [Psaras et al., 2012], using the same cache probability for every content item at a given node and the cooperative caching mechanism in [Ioannou and Weber, 2014; Ming et al., 2012] exploiting overall popularity and distance-to-server.

Caching and Forwarding Interaction

The search for an optimal interplay between in-network caching and forwarding has drained effort in ICN research, as driven by different user or network performance objectives. The optimal cache placement and forwarding problem applied to ICN was tackled in [Wang et al., 2013a]. Unlike our approach to forwarding, bandwidth sharing is not taken into account by such formulation, leading to results that are more appropriate for network dimensioning purposes than for end-user latency minimization. Among the contributions that focus on cache-aware forwarding strategies, [Eum et al., 2012] proposes an Interests-to-neighbor forwarding aiming at maximizing a difference of potentials, whose strength decreases with the distance to the content location. Similarly, in [Sourlas et al., 2014], authors suggest to forward Interests to the neighboring node that advertised the highest hit probability for the requested content object. A closer work to ours is [Yeh et al., 2014], where an optimization framework is defined to jointly handle backpressure-based forwarding and LFU-like content placement. The work designs a control plane that feeds the actual chunk-level data plane with flow rates and queue sizes in order to operate optimal content placement and request forwarding. Nodes must advertise their own queue states to their neighbors. Our solution differs in that it does not require signaling and does fully-distributed and dynamic cache insertion/replacement without requiring optimal content placement a priori. The latter aspect is important to guarantee self-adaptiveness to varying network/traffic conditions.

3.3 Problem statement

The problem of improving end-user delivery performance can be formulated as the minimization of the overall average delivery time for all users in the network and over all requested objects.

Take $q_{k,u}$ as the normalized request rate of object k from user u (namely, the popularity function at user u), and $\mathbf{w}_{u,r} \equiv (w_{k,u,r})_{k\in\mathcal{K}}$ as a vector whose every component is the probability to download object k from route r. In conformity with the BASS model [Carofiglio et al., 2013a], we assume that the significant latency originates from the route's bottleneck link. Hence, $\mathbf{h}_{u,r} \equiv (h_{k,u,r})_{k\in\mathcal{K}}$ denotes a vector whose every component is the probability that object k is served by a cache along route r but downstream its bottleneck (between the user and the bottleneck). The total cache budget along route r is limited to $x_{u,r}$ objects.

$\mathbf{E}[\mathrm{T}_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r})]$ is the average latency i.e., the round-trip time to retrieve object k on route r. We assume that bottleneck links are crossed by a single route. The set of routes available to user u is identified by $\mathcal{R}_u$. Content items can follow any of the available routes. Read "user" as a group of content consumers that share caches and routes. Another assumption is that such groups can not share neither caches nor routes.

Put together, we obtain the formulation:

$$
\begin{cases}
\underset{\substack{w_{k,u,r}\\ h_{k,u,r}}}{\text{Minimize}} \displaystyle\sum_{u\in\mathcal{U},k\in\mathcal{K},r\in\mathcal{R}_u} q_{k,u} w_{k,u,r}(1-h_{k,u,r})\mathbf{E}[\mathrm{T}_{u,r}(\mathbf{w}_{u,r},\mathbf{h}_{u,r})] & (3.1)\\
\text{subject to:} & \\
\displaystyle\sum_{k} h_{k,u,r} = x_{u,r}, \qquad \forall u, r & (3.2)\\
\displaystyle\sum_{r} w_{k,u,r} = 1, \qquad \forall k, u & (3.3)\\
0 \le w_{k,u,r} \le 1, \qquad \forall k, u, r & (3.4)\\
0 \le h_{k,u,r} \le 1 \qquad \forall k, u, r. & (3.5)
\end{cases}
$$

Similarly to [Carofiglio et al., 2013a], we model bottleneck links as $M/G/1$-PS queues, making the mean latency depend on the number of content downloads in progress. Hence,

$$
\mathbf{E}[\mathrm{T}_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r})] \approx \Big[\mu_{u,r} - \lambda \sum_{k} q_{k,u} w_{k,u,r}(1-h_{k,u,r})\Big]^{-1}, \tag{3.6}
$$

where λ is the content demand rate at the user and $\mu_{u,r}$ is the mean service rate at route

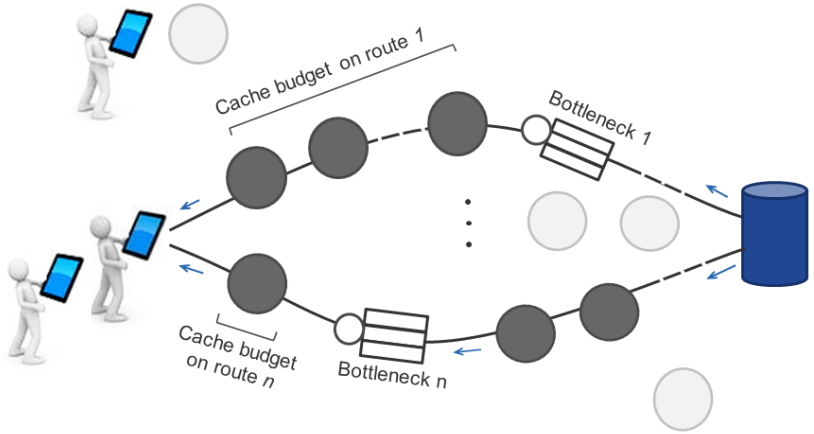


Figure 3.1 – The problem is depicted: find the optimal content hit ratios and route weights minimizing the mean content delivery latency.

r's bottleneck. The objective in Eq.3.1 can be simplified to:

$$\underset{\substack{w_{k,u,r} \\ h_{k,u,r}}}{\text{Minimize}} \sum_{u \in \mathcal{U}, r \in \mathcal{R}_u} \left[\rho_{u,r}^{-1}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r}) - 1 \right]^{-1} \tag{3.7}$$

$$\text{with the additional constraint that } \rho_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r}) < 1. \tag{3.8}$$

$\rho_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r}) \equiv \lambda \mu_{u,r}^{-1} \sum_k q_{k,u} w_{k,u,r}(1 - h_{k,u,r})$ denotes the load at route r's bottleneck. We thereby reformulated the mean latency minimization problem, by Little's theorem, as minimizing the sum of mean numbers of active flows. Fig.3.1 depicts the network system the optimization problem formalizes.

3.4 Optimal algorithm design

This section articulates, via mathematical analysis, guidelines shaping optimal caching and forwarding algorithms. They are later confirmed by the numerical experiments led using the SCIP [Achterberg, 2009] nonlinear optimization suite. In addition, as a sub-problem, we draw attention to the fact that the optimal performance of popularity-aware egress caches requires an optimal split of the forwarded traffic.

3.4.1 Optimal algorithm design guidelines through analytic insight

We indicate below that *i)* maximizing the hit ratio of a unique copy of the most popular objects downstream the bottleneck, and *ii)* single-path forwarding of the most popular object while load-balancing with latency-awareness the others are optimal traits. Both guidelines are cornerstones to the distributed *FOCAL* algorithm we introduce in the next section.

Store downstream bottlenecks a unique copy of the most popular objects

Maximizing the hit ratio of the most popular content items is optimal. Moreover, it is not suitable to maintain multiple instances of the same object across paths. The following proposition is an argument to this claim.

Proposition 3.1 (Optimal caching)**.** *Assume a catalog of size K, a cache budget x spread over R routes downstream their respective bottleneck and a Zipf content popularity distribution q with parameter $\alpha = 1$. For minimizing the miss traffic, the optimal cache management policy satisfying the following conditions consists in ensuring the maximum hit ratio h^{max}, $0 < h^{max} \leq 1$, to the $({}^{x}/_{h^{max}})$-most popular items as:*

$$\forall h : [1, K] \times [1, R] \rightarrow [0, h^{max}] \text{ a caching policy's hit ratio such that}$$
$$H(K) \equiv h^{max} + \int_{1}^{K} \sum_{r} h(v, r) dv = x \text{ and } \sum_{r} h(v, r) \leq 1, \forall v,$$

$$\int_{1}^{K} \frac{1 - \prod_{r}(1 - h(v, r))}{v} dv \leq h^{max} \log \frac{x}{h^{max}}. \tag{3.9}$$

Proof. We aim to prove that the miss traffic whatever the policy characterized by a hit ratio h, is always larger than the result of maximizing the hit ratio of the $({}^{x}/_{h^{max}})$-most popular content objects. Let λ the content arrival rate, formally, it

writes as follows:

$$\lambda \left[(1 - h^{max}) \int_1^{x/h^{max}} q(v)dv + \int_{x/h^{max}}^{K} q(v)dv \right] \leq \lambda \int_1^K \prod_r (1 - h(v,r))q(v)dv$$

$$i.e., \int_1^K \frac{1 - \prod_r (1 - h(v,r))}{v} dv \leq h^{max} \log \frac{x}{h^{max}}.$$

Holding from the definition of any hit function h:

$$H(K) = x \Leftrightarrow h^{max} \log \frac{H(K)}{h^{max}} = h^{max} \int_1^{H(K)/h^{max}} \frac{dv}{v} = h^{max} \log \frac{x}{h^{max}}.$$

Define $H(u)/h^{max} = v$. By u-substitution,

$$h^{max} \int_1^K \frac{\sum_r h(u,r)}{H(u)} du = h^{max} \log \frac{x}{h^{max}}.$$

Hence, as $H(u) \leq u$ and $1 - \prod_r (1 - h(u,r)) \leq \sum_r h(u,r)$,

$$\int_1^K \frac{1 - \prod_r (1 - h(u,r))}{u} du \leq h^{max} \log \frac{x}{h^{max}}.$$

□

Single-path forwarding of the most popular but load-balancing the others

We adopt the framework of non-linear optimization to capture additional traits optimal algorithms must comply with. They are, at least, necessary to solution optimality. The following lemma strengthens them.

Lemma 3.1 (Convex regime)**.** *Assuming that $|\mathcal{K}| = 1$, the latency minimization objective is convex under the condition on the content demand rate that*

$$\frac{1}{2} \sup \left\{ \frac{\mu_{u,r}}{w_{1,u,r}(1 - h_{1,u,r})} \right\} + \frac{1}{2} < \lambda < \inf \left\{ \frac{\mu_{u,r}}{w_{1,u,r}(1 - h_{1,u,r})} \right\}.$$

Whenever for a given value of $|\mathcal{K}|$ such a condition exists and is fulfilled, it has a

radical impact on the problem. As constraints are either affine or convex, the KKT conditions we invoke below are both necessary *and* sufficient in that regime, leading to strong optimal algorithm design guidelines.

Proof. With $|\mathcal{K}| = 1$, the objective reformulated in Eq.3.7 is a sum of terms whose Hessian matrix is:

$$\mathcal{H}_{u,r}^{(1)} \equiv \left(\mu_{u,r} - \lambda w_{1,u,r}(1 - h_{1,u,r})\right)^{-3} \begin{bmatrix} A & B \\ B & C \end{bmatrix}, \tag{3.10}$$

with

$$A = 2\lambda^2 \mu_{u,r} w_{1,u,r}^2, \tag{3.11}$$

$$B = -\lambda \mu_{u,r} \left(\mu_{1,u} + w_{1,u,r}(1 - h_{1,u,r})\right), \tag{3.12}$$

$$C = 2\lambda^2 \mu_{u,r}(1 - h_{1,u,r})^2. \tag{3.13}$$

The Hessian i.e., the determinant of $\mathcal{H}_{u,r}^{(1)}$, is positive *iif* $AC > B^2$

$\Leftrightarrow 4\lambda^4 \mu_{u,r}^2 w_{1,u,r}^2 (1 - h_{1,u,r})^2 > \lambda^2 \mu_{u,r}^2 \left(\mu_{1,u} + w_{1,u,r}(1 - h_{1,u,r})\right)^2$

$\Leftrightarrow \left(2\lambda w_{1,u,r}(1 - h_{1,u,r})\right)^2 > \left(\mu_{u,r} + w_{1,u,r}(1 - h_{1,u,r})\right)^2$. It follows that:

$$\frac{1}{2} \sup \left\{ \frac{\mu_{u,r}}{w_{1,u,r}(1 - h_{1,u,r})} \right\} + \frac{1}{2} < \lambda < \inf \left\{ \frac{\mu_{u,r}}{w_{1,u,r}(1 - h_{1,u,r})} \right\}. \tag{3.14}$$

If Eq.3.14 is verified at the candidate-optimum vector $(w_{1,u,r}^*, h_{1,u,r}^*)_{u,r}$, as the leading principal minor A is also positive, the Hessian matrix $\mathcal{H}_{u,r}^{(1)*}$ at that vector is definite positive in virtue of Sylvester's criterion. It further implies that the objective function is convex. □

The observation below highlights the importance of capacity (probed using retrieval latency) and popularity awareness in optimal caching and forwarding algorithms.

Observation 3.1 (Capacity and popularity awareness)**.** *Assuming a content object k is forwarded through multiple routes, always cached but not permanently, caching and forwarding algorithms that are characterized by ratios of the route weight over content k miss probability made proportional to the product of content popularity and the route's bottleneck link capacity, comply with necessary optimality conditions.*

Their solution vector verifies:

$$\frac{w^*_{k,u,r}}{1-h^*_{k,u,r}} \propto q_{k,u}\mu_{u,r}. \tag{3.15}$$

This analytic insight indicates that the most popular objects are affected near-one hit ratios, whereas the least popular content items end up with near-zero hit ratios and predominantly higher path weights on higher capacity routes.

Context. Define two vectors of decision variables $w \equiv (w_{k,u,r})$ and $h \equiv (h_{k,u,r})$. The Lagrangian of the problem, $\mathcal{L}(w, h, \vartheta)$, is:

$$\sum_{u,r}\Big[\left(\rho^{-1}_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r}) - 1\right)^{-1} - \vartheta^{(6)}_{u,r}\left(1 - \rho_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r})\right)\Big] \tag{3.16}$$

$$- \sum_{u,r}\vartheta^{(2)}_{u,r}x_{u,r} + \sum_{u,k,r}\vartheta^{(2)}_{u,r}h_{k,u,r} \tag{3.17}$$

$$- \sum_{u,k}\Big[\sum_{r}\vartheta^{(3)}_{k,u}w_{k,u,r} - \vartheta^{(3)}_{k,u}\Big] \tag{3.18}$$

$$- \sum_{u,k,r}\vartheta^{(4)}_{k,u,r}w_{k,u,r}(1 - w_{k,u,r}) \tag{3.19}$$

$$- \sum_{u,k,r}\vartheta^{(5)}_{k,u,r}h_{k,u,r}(1 - h_{k,u,r}) \tag{3.20}$$

with $\vartheta^{(2)}_{u,r}, \cdots, \vartheta^{(3)}_{k,u}, \cdots, \vartheta^{(6)}_{k,u,r} \in \mathbb{R}_+, \forall u \in \mathcal{U}$, $\forall k \in \mathcal{K}$ and $\forall r \in \mathcal{R}_u$, the related multipliers and ϑ their vector. According to KKT optimality conditions, the gradient of the Lagrangian at optimal vectors $w^* \equiv (w^*_{k,u,r})$ and $h^* \equiv (h^*_{k,u,r})$ is zero given the vector of optimal multipliers ϑ^*. It writes:

$$\nabla_{w,h}\mathcal{L}(w^*, h^*, \vartheta^*) = \vec{0}.$$

Thus,

$$\frac{\lambda\mu_{u,r}q_{k,u}(1-h^*_{k,u,r})}{\left(\mu_{u,r}-\lambda\sum_j q_{j,u}w^*_{j,u,r}(1-h^*_{j,u,r})\right)^2} = \vartheta^{(3)*}_{k,u} - \vartheta^{(4)*}_{k,u,r}(2w^*_{k,u,r}-1), \forall k,u,r \quad (3.21)$$

$$\frac{\lambda\mu_{u,r}q_{k,u}w^*_{k,u,r}}{\left(\mu_{u,r}-\lambda\sum_j q_{j,u}w^*_{j,u,r}(1-h^*_{j,u,r})\right)^2} = \vartheta^{(2)*}_{u,r} + \vartheta^{(5)*}_{k,u,r}(2h^*_{k,u,r}-1), \forall k,u,r. \quad (3.22)$$

Furthermore, the complementary slackness conditions impose that:

$$x_{u,r} - \sum_k h^*_{k,u,r} = 0, \quad \forall k,u,r \quad (3.23)$$

$$\sum_r w^*_{k,u,r} - 1 = 0, \quad \forall k,u,r \quad (3.24)$$

$$\vartheta^{(4)*}_{k,u,r} w^*_{k,u,r}(1-w^*_{k,u,r}) = 0, \quad \vartheta^{(4)*}_{k,u,r} \geq 0, \forall k,u,r \quad (3.25)$$

$$\vartheta^{(5)*}_{k,u,r} h^*_{k,u,r}(1-h^*_{k,u,r}) = 0, \quad \vartheta^{(5)*}_{k,u,r} \geq 0, \forall k,u,r \quad (3.26)$$

$$\text{and by stability condition, } \vartheta^{(6)*}_{u,r} = 0, \quad \forall u,r. \quad (3.27)$$

It follows that:

$$(1-h^*_{k,u,r})\left(\vartheta^{(2)*}_{u,r} + \vartheta^{(5)*}_{k,u,r}(2h^*_{k,u,r}-1)\right) = w^*_{k,u,r}\left(\vartheta^{(3)*}_{k,u} - \vartheta^{(4)*}_{k,u,r}(2w^*_{k,u,r}-1)\right)$$

Assume that Constraints (3.4) and (3.5) are not saturated. Content objects this applies to are multipath-forwarded and not perfectly cached. Previous equation simplifies to:

$$\frac{w^*_{k,u,r}}{1-h^*_{k,u,r}} = \frac{\vartheta^{(2)*}_{u,r}}{\vartheta^{(3)*}_{k,u}}. \quad (3.28)$$

At this point, we learn that the content miss ratio on a route divided by the route weight for that content equals to $\vartheta^{(3)*}_{k,u}$, a property of that content divided by $\vartheta^{(2)*}_{u,r}$, a property of the route. The following relation also hold:

$$\sum_r w^*_{k,u,r} = \sum_r \frac{(1-h^*_{k,u,r})\vartheta^{(2)*}_{u,r}}{\vartheta^{(3)*}_{k,u}} = 1 \Leftrightarrow \vartheta^{(3)*}_{k,u} = \sum_r (1-h^*_{k,u,r})\vartheta^{(2)*}_{u,r} \quad (3.29)$$

It suffices to choose $\vartheta^{(2)*}_{u,r}$ proportional to route r's route's bottleneck link capacity and $\vartheta^{(3)*}_{k,u}$ inversely proportional to content k's popularity.

As a consequence to this substitution, in Eq.3.29 a content having, whatever route capacities, a low hit ratio should be of low popularity. □

3.4.2 Numerical solutions

We numerically solved problem (3.1) using SCIP 3.2.1 [Achterberg, 2009] + IPOPT 3.12.4 [Wächter and Biegler, 2006] + PARDISO 5.0.0 [Kuzmin et al., 2013; Schenk et al., 2007, 2008], a Mixed Integer Non-Linear Program (MINLP) optimization suite. A.1 is the corresponding mathematical program tuned to cope with the ZIMPL modeling language [Koch, 2004] restrictions. An easy way to run it without installing the suite is via the NEOS Server[2] free internet service. We investigated two schemes related to whether traffic characteristics are perfectly known a priori or not:

Static caching (or replica placement) Assume an oracle reveals the content popularity distribution. Then, optimal caching consists in ensuring a perfect hit ratio of 1 to the $(\sum_r x_{u,r})$-most popular content items. The optimal forwarding for these most popular objects is single-path whereas the remaining traffic can be load balanced across available routes.

Example 3.4.2.1. *Consider the topology in Fig.3.1 with a* 10*-objects-per-route cache budget. Some content might be permanently stored, meaning that their hit ratio might be 1. We intend to optimize the sum of the mean numbers of active flows across the* 3 *routes separating the user to a repository of* 1000 *objects. Objects popularity follows a Zipf distribution with skewness parameter* $\alpha = 1$*. The object demand rate is* 10 *objects/s whereas route bottlenecks can respectively achieve* 5*,* 3 *and* 2 *objects/s downlink service rates. These are also the default parameter settings of the mathematical program in A.1 we submitted to the SCIP solver.*

Here is the optimal solution: as depicted in Fig.3.2, up to rank 30*, objects are cached with a hit ratio of 1 and single-path forwarded. From rank* 30*, none is cached since there is no more slot available. At this point, forwarding balance objects across routes, splitting flows like content 32 if needed. Optimal link loads end up at* 0.55

[2] http://www.neos-server.org/neos/solvers/go:scip/MPS.html

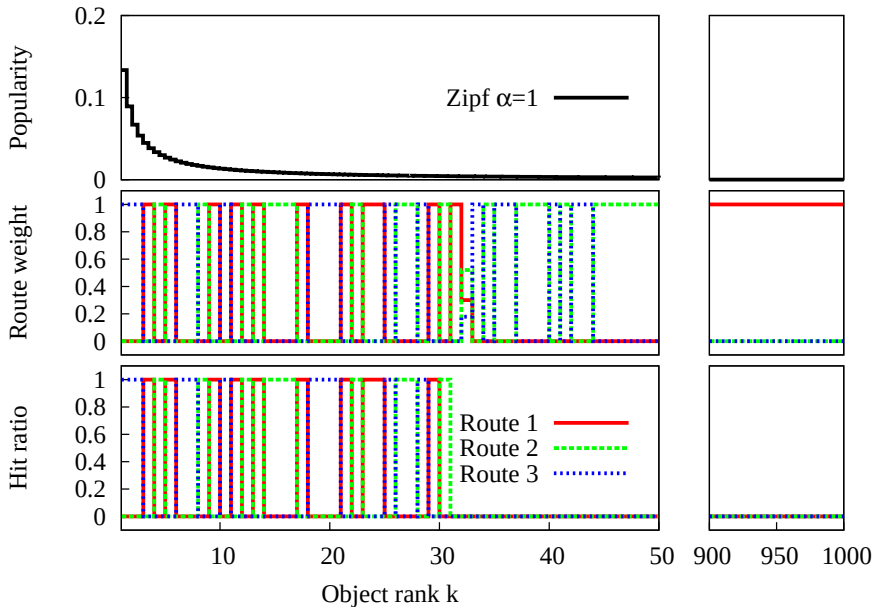


Figure 3.2 – Static caching optimum per popularity rank: 3 routes, 10-objects-per-route cache budget and a catalog of 1000 objects. Observe load balancing and the end of caching right after rank 30.

for the highest-capacity bottleneck, 0.42 *for the medium-capacity one and* 0.29 *for the lowest-capacity link. The exercise results in a sum of mean numbers of active flows equal to* 2.42 *at the optimum.*

In the vast majority of applications, content popularity distribution has to be inferred from ongoing traffic. Caching policies dynamically react to non-stationary content arrival and infer decision parameters such as content popularity on-the-fly. However, state-of-the-art traffic models are so approximate that opportunistic caching can not guarantee perfect hit ratios of 1 [Leonardi and Torrisi, 2015]. In this mode, a portion of the most popular traffic will always leak through the bottleneck links. We take into account real-world cache performance by defining for Constraint (3.5) a hit ratio's upper bound $h^{max} < 1$. Surprisingly, as reported by Fig.3.3, the previous algorithm is still optimal in this case. Even with $h^{max} = 1/2$, optimal caching does not consist in duplicating objects over several routes. On the contrary, at the optimum, the number of popular objects cached 50% of the time equals twice the cache budget and single-path forwarding still prevails.

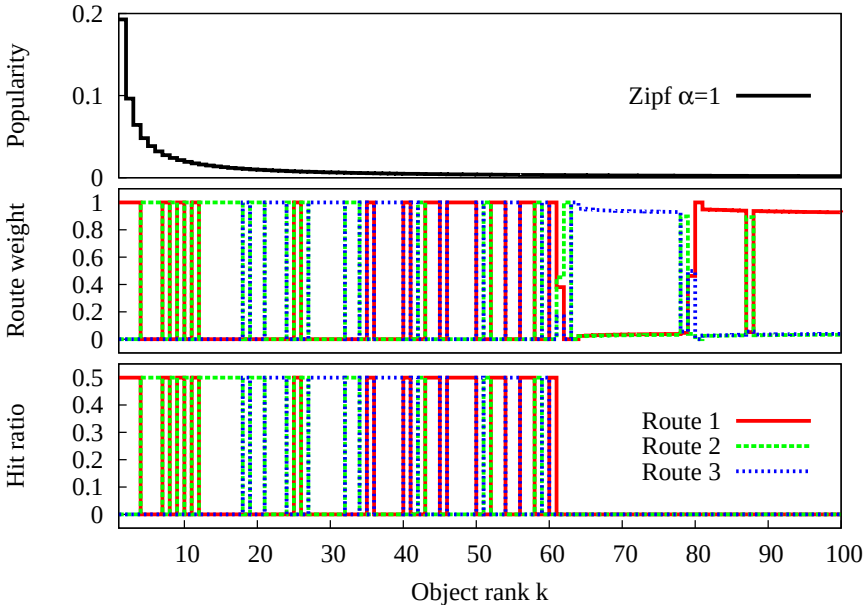


Figure 3.3 – Opportunistic caching optimum per popularity rank: $h^{max} = {}^1/_2$, 3 routes, 10-objects-per-route cache budget and a catalog of 100 objects. Observe load balancing and the end of caching right after rank 60.

Opportunistic caching

Example 3.4.2.2. *Consider the same setup as in the previous example, but with a 100-object catalog, a demand rate of* 15 *objects/s and cache performance upper bounded by* $h^{max} = {}^1/_2$. *In conformity with Proposition 3.1, SCIP solver finds the optimal solution depicted in Fig.3.3 where up to rank 60, objects are cached with hit ratio* $^1/_2$ *and single-path forwarded. Note that the objects concerned are those whose rank is less than* $h^{max}\times$ *total cache budget. Beyond rank* 60, *typically for content* 61 *and* 79, *we experienced load balancing even though number of flows get just single-path forwarded due to their scarcity. Finally, optimal link loads end up at* 0.85 *for the highest-capacity bottleneck,* 0.81 *for the medium-capacity one and* 0.76 *for the lowest-capacity one. At the optimum, the sum of the mean numbers of active flows reaches* 13.4, *its minimal value.*

Up to now, the optimal content hit ratios have been set into the caching systems, as fixed targets to achieve. However, number of caching systems are autonomously governed by an insertion/eviction policy aiming to cope with the fact that their capacity is much

smaller than the size of catalog they address. Hence, a content hit ratio becomes an outcome to the policy operation, then no longer a given. Below, we take the single-path forwarding guideline further. We advocate that there exists some selection of the flows single-path forwarded towards a common egress cache, referred to as bundling, that maximizes dynamic caches hit ratio.

3.4.3 Maximizing the hit ratio of dynamic caches through optimal bundling

Take for instance caches ruled by the *(evict the)* Least Recently Used (LRU) policy. They are not just passive storage handling content objects independently. Their dynamics and performance depend on the relative popularity of content objects they deal with. This has to be taken into consideration when the most popular flows are single-path forwarded in accordance with guideline 3.4.1.0. A bundle denotes the set of flows sent in a single-path way to the same egress cache whereas bundling is composing that set. That bundle-dependent performance claim holds from solving, at every node n, the following optimization sub-problem.

Definition 3.1. *Let $\Gamma^+(n,k)$ be the set of egress caches for content k at node n. We assume homogeneous routing in the sense that every content can be forwarded to any of the egress caches. Therefore, it is enough to define for each node n a content agnostic set of egress caches $\Gamma^+(n) \equiv \Gamma^+(n,k), \forall k \in \mathcal{K}$.*

The sub-problem, sketched in Fig.3.4, consists in finding the best combination $\{i_{k,b} : b \in \Gamma^+(n), k \in \mathcal{K}\}$ of content objects for each cache in order to maximize the overall hit probability. The egress traffic is split so that the number of distinct objects sent to every cache b can not exceed a real factor η_b times the cache size x_b.

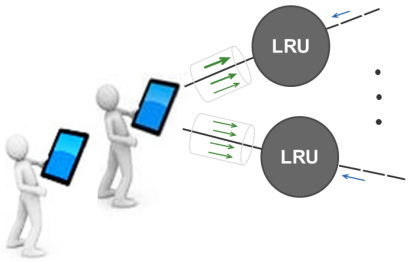


Figure 3.4 – The sub-problem is depicted: find the optimal flow bundling maximizing egress LRU caches hit ratio.

$$\begin{cases} \underset{i_b}{\text{Maximize}} \sum_{b\in\Gamma^+(n),k\in\mathcal{K}} q_k i_{k,b} h_{k,b}(i_b \equiv (i_{1,b},\cdots,i_{|\mathcal{K}|,b})) & (3.30) \\ \text{subject to:} \\ i_{k,b} \in \{0,1\}, \qquad \forall k,b & (3.31) \\ \sum_{b\in\Gamma^+(n)} i_{k,b} \leq 1, \qquad \forall k & (3.32) \\ \sum_{k\in\mathcal{K}} i_{k,b} \leq \eta_b x_b, \qquad \forall b. & (3.33) \end{cases}$$

The *Independent Reference Model* (IRM) assumes that the content request process is a sequence of independent random variables with a common probability distribution [Roberts and Sbihi, 2013]. So, under IRM and according to Che's approximation of the LRU cache hit ratio [Che et al., 2006], content k hit probability in cache b is $h_{k,b}(i_b) = 1 - e^{-q_k t_C(i_b,b)}$. Here, $t_C(i_b,b)$ shortened as $t_C(i_b)$, is egress cache b Characteristic Time. It is deemed the time before object eviction.

Analytic insight

The proposition below gives a necessary criterion for the hit probability to be maximal assuming LRU caches. It is clarified by an illustrative corollary that tackles the case of same-size LRU caches and flow bundles. In a nutshell, we state that it complies with necessary optimality conditions to forward the union of a suitably small compact subset of the highest popularity ranks and a subset of the lowest popularity ranks to the same

egress LRU cache, as long as the cumulative popularities of the resulting bundles follow the same order as those of the popular subsets.

Consider this preliminary lemma:

Lemma 3.2 (Characteristic time w.r.t. content activation)**.** *Under IRM , forwarding a new content to an LRU cache decreases its Characteristic Time. That negative slope is the ratio of the content hit probability over the cache miss probability.*

Proof. We inherit from Che's approximation the relation:

$$F(i_b, t_C(i_b)) = \sum_{k \in \mathcal{K}} i_{k,b} \left(1 - e^{-q_k t_C(i_b)}\right) - x_b = 0, \quad \forall b \in \Gamma^+(n). \tag{3.34}$$

We relax Eq.3.34 by assuming that $i_{k,b} \in]0, 1]$. Then, we apply the Implicit Function Theorem to Eq.3.34. It gives:

$$\begin{aligned}
\frac{\partial t_C(i_b)}{\partial i_{k,b}} &= -\frac{\partial F(i_b, t_C(i_b))}{\partial i_{k,b}} \left(\frac{\partial F(i_b, t_C(i_b))}{\partial t_C(i_b)}\right)^{-1}, \quad \forall k \in \mathcal{K}, \forall b \in \Gamma^+(n) \\
&= -\left(1 - e^{-q_k t_C(i_b)}\right) \left(\sum_{l \in \mathcal{K}} q_l i_{l,b} e^{-q_l t_C(i_b)}\right)^{-1} \\
&= -\frac{h_{k,b}(i_b)}{M_b(i_b)} < 0,
\end{aligned}$$

where $h_{k,b}(\cdot)$ denotes content k hit probability in cache b and $M_b(\cdot)$ the cache overall miss probability. □

We exploit this result to derive an optimal egress flow bundling.

Proposition 3.2 (Optimal forwarding towards LRU caches)**.** *We assume that the content objects k that can be harmlessly discarded from caches are those for which k is large. Assuming that IRM holds and that node n's egress nodes $a \in \Gamma^+(n)$ are equipped with x_a-size LRU caches, any optimal flow bundling, which maximizes cache hit ratios, is such that any permutation of some converged cache characteristics would be detrimental to the hit ratios. Formally, $\forall a \in \Gamma^+(n), \exists \epsilon^*_{k,a,b}$:*

$$x_a \leq \sum_{k\in\mathcal{K}} \frac{i^*_{k,a} \inf_{b\in\Gamma^+(n)\setminus\{a\}} \left\{\left(1-(1-h_{k,b}(i^*_b))\,\gamma^*_b + \epsilon^*_{k,a,b}\right) h_{k,b}(i^*_b)\right\}}{1-(1-h_{k,a}(i^*_a))\,\gamma^*_a}, \tag{3.35}$$

$$with\ \gamma^*_b = (M_b(i^*_b))^{-1} \sum_{l\in\mathcal{K}} q_l i^*_{l,b}\ \ and\ \lim_{k\to 1^+} \epsilon^*_{k,a,b} = 0. \tag{3.36}$$

We demonstrate it below, as well as this illustrative corollary.

- Define x_b as the capacity of an egress LRU cache $b \in \Gamma^+(n)$.
- Define B^j_b as the flow bundle forwarded to cache b at iteration step j.
- $B^0_b \equiv \varnothing$. Let $\mathcal{S} \subset \mathcal{K} \setminus \bigcup_b B^j_b$. At iteration step $j+1$, $B^{j+1}_b \equiv B^j_b \cup \mathcal{S}$.
- Let q_k be content k popularity.
- Define $P^j_b \equiv \sum_{k\in B^j_b} q_k$ as the cumulative popularity of bundle b at step j.

Corollary 3.2.1. *Merging into the same bundle a suitably small compact subset of the highest popularity ranks with lower popularity ranks, such that bundles cumulative popularity strict ordering is preserved, complies with Proposition 3.2's optimality criterion.*

By *suitably small compact subsets* of the highest popularity ranks, we mean $\lceil \eta_b x_b \rceil$-size subsets of the most popular contents, $\eta_b \leq 1$, such that each subset is an arithmetic sequence with common difference 1, *e.g.*,$\{1,2,3\}, \{4,5,6,7,8\}$.

Preserving the cumulative popularity strict ordering means ensuring that, $\forall a,b \in \Gamma^+(n), P^j_b < P^j_a \Rightarrow P^{j+1}_b < P^{j+1}_a$. For example, assume a Zipf popularity distribution with skewness $\alpha = 1$. Since $\{1,2,3\} \cup \{9\}$ has strictly bigger cumulative popularity than $\{4,5,6,7,8\} \cup \{10\}$, such an iteration is said to preserve prior cumulative popularity strict ordering.

Proof. The Lagrangian of that sub-problem is:

$$\mathcal{L}_c(i_b, \phi) = \sum_{b \in \Gamma^+(n), k \in \mathcal{K}} q_k i_{k,b} \left(1 - e^{-q_k t_C(i_b)}\right) + \sum_{b,k} \phi_{k,b}^{(1)} i_{k,b} \left(i_{k,b} - 1\right) - \sum_k \phi_k^{(2)} \left(\sum_b i_{k,b} - 1\right) - \sum_b \phi_b^{(3)} \left(\sum_k i_{k,b} - \eta_b x_b\right)$$

with $\phi_{k,b}^{(1)}, \phi_k^{(2)}, \phi_b^{(3)} \in \mathbb{R}_+, \forall k \in \mathcal{K}, \forall b \in \Gamma^+(n)$, the related multipliers and ϕ their vector. $\Gamma^+(n)$ stands for the set of node n egress nodes.

Karush-Kuhn-Tucker optimality condition $\nabla_{i_b} \mathcal{L}_c(i_b^*, \phi^*) = \vec{0}$, yields:

$$q_k \left[1 + e^{-q_k t_C(i_b^*)} \left(\frac{\partial t_C(i_b^*)}{\partial i_{k,b}^*} \sum_{l \in \mathcal{K}} q_l i_{l,b}^* - 1\right)\right] + \phi_{k,b}^{(1)*}(2i_{k,b}^* - 1) - \phi_k^{(2)*} - \phi_b^{(3)*} = 0, \forall k, b. \tag{3.37}$$

Lemma 3.2's argument makes this equation equivalent to:

$$q_k h_{k,b}(i_b^*) \left(1 - \left(1 - h_{k,b}(i_b^*)\right) \gamma_b^*\right) + \phi_{k,b}^{(1)*}(2i_{k,b}^* - 1) - \phi_k^{(2)*} - \phi_b^{(3)*} = 0, \quad \forall k, b \tag{3.38}$$

where $\gamma_b^* = \left(M_b(i_b^*)\right)^{-1} \sum_{l \in \mathcal{K}} q_l i_{l,b}^*$ is a constant typical to the optimal setup of every cache b.

Then, we characterize the content objects k that are not cached at all at the optimum, those for which $\forall b \in \Gamma^+(n), i_{k,b}^* = 0$. This means that $\phi_k^{(2)*} = 0$ since the matching constraint is not saturated, and

$$q_k h_{k,b}(i_b^*) \left(1 - \left(1 - h_{k,b}(i_b^*)\right) \gamma_b^*\right) - \phi_{k,b}^{(1)*} - \phi_b^{(3)*} = 0, \quad \forall b. \tag{3.39}$$

Therefore, as content objects that can be ignored are those for which $k \uparrow \infty$, valid values for multipliers in Eq.3.39 are such that $\left(\phi_{k,b}^{(1)*}, \phi_b^{(3)*}\right) q_k^{-1} \downarrow (0,0)$ as $k \downarrow 1, \forall b$.

Let us characterize the contents k that are cached at the optimum, those for which $\exists a \in \Gamma^+(n) : i_{k,a}^* = 1$ and $\forall b \in \Gamma^+(n) \setminus \{a\}, i_{k,b}^* = 0$:

$$q_k h_{k,a}(i_a^*) \left(1 - \left(1 - h_{k,a}(i_a^*)\right) \gamma_a^*\right) + \phi_{k,a}^{(1)*} - \phi_k^{(2)*} - \phi_a^{(3)*} = 0,$$
$$q_k h_{k,b}(i_b^*) \left(1 - \left(1 - h_{k,b}(i_b^*)\right) \gamma_b^*\right) - \phi_{k,b}^{(1)*} - \phi_k^{(2)*} - \phi_b^{(3)*} = 0.$$

Removing $\phi_{k,\cdot}^{(1)*}$ positive terms yields the following inequalities:

$$\begin{aligned} -q_k h_{k,a}(i_a^*)\left(1-\left(1-h_{k,a}(i_a^*)\right)\gamma_a^*\right)+\phi_k^{(2)*}+\phi_a^{(3)*}\geq 0,\\ q_k h_{k,b}(i_b^*)\left(1-\left(1-h_{k,b}(i_b^*)\right)\gamma_b^*\right)-\phi_k^{(2)*}-\phi_b^{(3)*}\geq 0. \end{aligned}$$

Finally, the sum of both lines gives:

$$\frac{h_{k,a}(i_a^*)}{h_{k,b}(i_b^*)}\leq \frac{1-\left(1-h_{k,b}(i_b^*)\right)\gamma_b^*+\epsilon_{k,a,b}^*}{1-\left(1-h_{k,a}(i_a^*)\right)\gamma_a^*}, \tag{3.40}$$

with $\epsilon_{k,a,b}^*\equiv\left(\phi_a^{(3)*}-\phi_b^{(3)*}\right)q_k^{-1}$ and $\lim_{k\to 1^+}\epsilon_{k,a,b}^*=0$. By summing over all the content objects that share cache a, it follows that:

$$\sum_k i_{k,a}^* h_{k,a}(i_a^*)=x_a\leq\sum_k\frac{i_{k,a}^*\inf_b\left\{\left(1-\left(1-h_{k,b}(i_b^*)\right)\gamma_b^*+\epsilon_{k,a,b}^*\right)h_{k,b}(i_b^*)\right\}}{1-\left(1-h_{k,a}(i_a^*)\right)\gamma_a^*}.$$

This concludes Proposition 3.2's proof.

Corollary 3.2.1 claims that inserting into the same bundle a suitably small compact subset of the highest popularity ranks , then adding lower popularity ranks such that bundles cumulative popularity strict ordering is preserved, obeys to the optimality condition stated by Eq.3.35. Indeed, upon such a bundling, except for one cache a, it exists a cache $l\equiv\arg\inf_b\{\cdot\}:\gamma_a^*>\gamma_l^*$ and $h_{k,l}(i_l^*)>h_{k,a}(i_a^*)\ \forall k$ in cache a due to a lower cumulative content popularity in l and Lemma 3.2. Thus, $1-\left(1-h_{k,l}(i_l^*)\right)\gamma_l^*>1-\left(1-h_{k,a}(i_a^*)\right)\gamma_a^*$. As l may not exist when a is the cache accommodating the lowest popularity items, choose multipliers $\phi_b^{(3)*}=0$ except for that cache. □

Numerical solutions

We computationally solved this MINLP, formulated as the mathematical program in A.2, under a local SCIP installation.

Default formulation The original version of the problem possesses an intuitive optimum that creates bundles of the same size as the target cache. Optimal bundles contain the

most popular content objects. Whether they are conveyed towards one cache or another does not matter. Indeed, the LRU caches never evict anything, ensuring an overall hit ratio of 1. Unfortunately, the solution is not practical since implementing such a forwarder assumes a perfect a priori knowledge of content popularity and egress cache sizes.

Formulation with fixed and over-sized bundles We change Constraint (3.33) to an equality to investigate the more realistic case where bundle sizes are arbitrarily fixed and made bigger than the target cache. Such a MINLP, as a nonlinear 0-1 equality multiple knapsack problem, is harder than its inequality counterpart, which is already harder than the **NP**-hard nonlinear 0-1 knapsack problem [Hochbaum, 2007]. Moreover, Che's approximation defines the characteristic time as an implicit function expensive to evaluate. Hence, the few successfully solved instances had tiny number of objects, tiny and different cache sizes and fixed bundle size equaling cache size + 1. Still, the optimal solution consists in bundles merging a suitably small compact subset of the highest popularity ranks with lower popularity ranks, such that bundles cumulative popularity strict ordering is preserved.

Example 3.4.3.1. *Consider a tiny catalog of 14 objects ranked according to a Zipf law. The popularity skewness is 1. We aim to split the traffic into two bundles of 6 and 7 flows for feeding two egress LRU caches. The caches can accommodate up to 5 and 6 objects. These are the default parameter settings of A.2. The optimal solution, which gives an overall hit ratio of 0.91, places content ranks* $B_1 = \{1, 2, 3, 10, 11, 12\}$ *in bundle 1 and content ranks* $B_2 = \{4, 5, 6, 7, 8, 9, 13\}$ *in bundle 2. Observe that* $B_1 = \{1, 2, 3\} \cup \{10, 11, 12\}$ *and* $B_2 = \{4, 5, 6, 7, 8, 9\} \cup \{13\}$ *and cumulative popularity* $1 + 1/2 + 1/3 + 1/10 + 1/11 + 1/12 = 2.107 > 1.072 = 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + 1/9 + 1/13$ *in full compliance with Corollary 3.2.1.*

In the next section, we derive distributed algorithms from these guidelines that try to minimize the objective function Eq.(3.1) by obtaining $w_{k,u,r}$ and $h_{k,u,r}$ without any coordination among the nodes and no signaling. That optimal objective can be heuristically generalized to every node n in the network by substituting $\mathbf{E}[\mathrm{T}_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r})]$ with $\mathbf{E}[\mathrm{T}_{u,r}(\mathbf{w}_{u,r}, \mathbf{h}_{u,r})](n)$ the local virtual residual round-trip time (VRTT) on route r.

Hence we set the probability to store an object k at a given node n, proportional to the popularity and latency locally observed at node n. A globally optimal strategy

performed in each node would heuristically prefer to cache locally popular content having high retrieval latency.

Forwarding will be single-path for the most popular content items and multipath for the others, with a route weight inversely proportional to the number of pending interests per content and per route. As shown in [Carofiglio et al., 2013c], such a weight estimation reflects link capacity.

3.5 FOCAL

3.5.1 Latency-aware caching strategies

Taking into account latency awareness into cache management, can improve alone the delivery time of latency-sensitive applications and on average the global delivery time perceived by user, as prescribed by Proposition 3.1 and shown in [Carofiglio et al., 2015a]. Before considering the joint effect of latency-aware caching and forwarding, we present a novel stochastic caching mechanism, exploiting monitored latency information for cache insertion decisions and not involving cache coordination. The novel latency-aware caching policy is named *LAC+* and builds upon the LAC proposal in [Carofiglio et al., 2015c] that we summarize below. The enhancement of LAC+ consists in strengthening latency-dependency in probabilistic cache decisions w.r.t. LAC, based on an online monitoring and estimation of the second-order moment of latency distribution.

LAC

In ICN, when a requested content object is not available in cache, a cache miss event occurs and the Interest is forwarded up to the first hitting cache where the corresponding Data is retrieved and sent downstream to the client. On the reverse path to the client, every cache decides whether or not storing the object, at the cost of triggering the eviction of another object due to finite storage space constraints.

According to LAC, at time t, the decision to store object k is positive with a given probability $p_k(t)$ and negative otherwise (with probability $1 - p_k(t)$). As a special case of

a prior proposal from [Carofiglio et al., 2015c], we characterize $p_k(t)$ as

$$p_k(t) \equiv \min\left(\epsilon\frac{T_k(t)}{\bar{T}(t)}, 1\right) \tag{3.41}$$

where $T_k(t)$ refers to content k monitored latency at time t, $\bar{T}_k(t)$, $\bar{T}(t)$ respectively to the temporal averages for content k and for all cached contents computed up to time t. $\bar{T}(t)$ and $\bar{T}_k(t)$ are estimated using Exponentially Weighted Moving Averages (EWMA), with a weight associated to the historical value of average latency set to $\alpha = 0.9$. The cache insertion probability, $p_k(t)$ results from the product of a small factor, ϵ, modulated by the ratio of its retrieval latency over the average latency of cached objects. A first assessment of LAC performance suggested that further benefits may result from strengthening the latency-awareness contribution by highlighting second order moment characteristics of monitored latency. This is the rationale behind LAC+ proposal.

LAC+

LAC may suffer from the slow convergence of any other probabilistic approach (see e.g. [Psaras et al., 2012]), due to the small ϵ factor. In simulations, we observe a non negligible time for even very popular objects to be persistently cached. We recall that the ideal behavior of a cache should be to capture the most valuable objects (according to a defined cost function), while avoiding unnecessary replication across network of caches. Unnecessary replication is typically object replication below a bottleneck or the lack of implicit coordination between neighboring caches. LAC+ achieves such objective by supplementing LAC with an outlier tracking function, meant to estimate second order moment of observed latency distribution. The outlier tracking function, denoted as Θ_k, significantly increases cache insertion probability for those exhibiting an exceptionally high deviation in comparison the other content objects. Such outliers correspond to significantly higher-than-average latency items, that is important to cache even when not very popular. According to LAC+, at time t, the decision to store object k is positive with a given probability $p_k^+(t)$ and negative otherwise (with probability $1 - p_k^+(t)$). We define $p_k^+(t)$ as the linear combination of two terms:

$$p_k^+(t) \equiv p_k(t) + (1 - p_k(t))\Theta_k(t). \tag{3.42}$$

Let μ_t and σ_t be the average and standard deviation of all $\bar{T}_i(t), \forall i \in \mathcal{K}$, at a given node. The z^{th} quantile being $Q_z(t) = \mu_t + z\sigma_t$, it follows that

$$\Theta_k(t) \equiv \max\left(\frac{\bar{T}_k(t) - Q_z(t)}{\sup_{i\in\mathcal{K}}\{\bar{T}_i(t)\} - Q_z(t)}, 0\right). \tag{3.43}$$

$p_k^+(t)$ inherits its first term from LAC. Its added value dwells in the second term, that allows to account for objects with a sensibly higher-than-average latency, in order to cache them even when not very popular (namely, when not selected by the filtering embedded in the first term). Its purpose is to strengthen the latency dependency of the caching decision to favor, by means of Θ_k, a positive caching decision for those objects whose retrieval can be very costly: e.g. long distance to the hitting cache, upstream congestion or severe bandwidth limitations. $\Theta_k(t)$ is defined as the probability that object k average latency at time t is an outlier.

3.5.2 Latency-aware forwarding strategies

Latency reduction can be also achieved via smart hop-by-hop request forwarding strategies trying to minimize *i)* distance to the first hitting cache, *ii)* congestion status of the network. To such extent, the presence of multiple paths is clearly essential. Using as baseline for comparison the *uniform random forwarding* approach that blindly selects with equal probability output interfaces in FIB, our focus is on the family of distributed, dynamic load-balancing approaches whose objective is to split content requests over time and through the available output interfaces such as to minimize *i)-ii)* on average. In [Carofiglio et al., 2013c], a load balancing scheme is derived from a joint optimization of end-user rate/congestion control and multipath forwarding under the objective of minimizing the maximum link load network-wide. The minimization of the maximum link load implicitly leads to a significant reduction of the overall average latency as it can be appreciated in the simulated scenarii. Hereinafter we refer to such an approach simply as *Load balancing (LB)*. LB selects available output interfaces per FIB entry randomly according to computed weights. At the beginning, each interface has the same weight equal to one and the randomized forwarding process is uniform over available output interfaces. This allows to probe all available interfaces and to monitor the average number of outstanding Pending Interests (PI) per FIB entry and per interface. Such a metric reflects

the residual latency due to first hitting cache distance and congestion status. After this initial phase, the computation of the weights driving interface selection simply consists in taking the average number of PI per FIB entry and per output interface normalized to the total average number of PI per FIB entry (so that weights are comprised between 0 and 1). Ideally, LB works on per-content FIB entries enabling a fine granular load-balancing at flow scale. However, a feasible approximation that keeps limited FIB state replaces per-content with per-prefix entries aggregating all content names behind the same prefix (FIB lookup is assumed to be Longest-Prefix Match). Note that in our simulations we adopt a per-content LB approach with the objective to quantify its best performance.
LB may achieve significant improvement of overall end-user throughput/latency over uniform random forwarding via load-aware utilization of multiple paths. However, it is not capable of realizing implicit cache coordination for caches along different paths, as a consequence of its randomized weighted split, that load-balances Interest for the same content over all output interfaces according to the weights. To understand this issue, let us consider the case of three output interfaces available for a given content k with associated weights, w_1, w_2, w_3. At each incoming request for content k, LB splits the Interest arrival process over time into three output processes, with rate respectively w_1, w_2, w_3 of the total, without selecting the same output interface for a given chunk request. As a result, the arrival process at caches along the three paths has the same characteristics (except for the rate) of the original one, leading to caches operating independently and storing the same items.

Intuitively such a behavior advantages most popular objects cached with high probability over all available paths, but reduces overall caching benefits due to lack of cache coordination. Instead, splitting Interests in a way to persist the selection of one or few single output interfaces over time on a per-content basis, (while keeping per-prefix load-balancing according to LB weights) would differentiate the arrival process at caches along the three paths, so realizing implicit cache coordination and better overall performance. This idea, grounded in Proposition 3.1, Proposition 3.2 and Corollary 3.2.1, inspires our proposal for an enhanced load balancing scheme, that we name *LB-Perf* (Load Balancing with Persistent Forwarding).

In an initial phase, LB-Perf computes per-prefix weights to associate to available output interfaces as in LB case. FOCAL is also equipped by a popularity sampler which continuously monitors the most popular objects and store their name locally. The method

Algorithm 1: The most popular content items are sampled in PopularFiles. Create flow bundles, one per FIB entry, based on observed interest volume and associate persistent faces to popular content items.

```
At update time (every ΔT);
Faces are ranked every ΔT_f ≫ ΔT;
T ← T + ΔT ;
IsPersistentDisabled = FALSE;
foreach FileName in PopularFiles do
    prefix ← GetFIBPrefix(FileName) ;
    OuputFaces ← GetOutputFaces(prefix) ;
    FlowBundle ← GetFlowBundle(prefix) ;
    Face ← OuputFaces.Begin();
    Sort(FlowBundle by InterestCounter) ;
    if ( T ≥ ΔT_f ) then
        Sort(FaceRecord by weight) ;
        T ← 0 ;
    end
    CumSum ← 0 ;
    foreach ( FlowRecord in FlowBundle ) do
        CumSum ← CumSum + FlowRecord.Popularity() ;
        while ( Face ≠ OuputFaces.End() ) do
            if ( CumSum < Face.weight ) then
                FlowRecord.SetFace(Face);
                break ;
            else
                CumSum ← CumSum + FlowRecord.Popularity() ;
                Face ← OuputFaces.Next();
            end
        end
    end
end
```

Algorithm 2: Popularity based persistent face selection.

```
At Interest I arrival with name /p/file_name/chunk_name ;
I matches name prefix /p in the FIB ;
OuputFaces ← GetOutputFaces(prefix = /p) ;
if ( IsPersistentDisabled ) then
    LoadBalancing.Update(OuputFaces.weights) ;
    FaceID ← LoadBalancing.SelectFrom(OuputFaces) ;
else
    PopularitySampler.Insert(I) ;
    if (PopularitySampler.Find(FileName(I))) then
        PopularFiles.ManageHit(I) ;
    end
    if (PopularFiles.IsPopular(FileName(I))) then
        FlowRecord ← FlowBundle.Find(FileName(I)) ;
        FlowRecord.InterestCounter++ ;
        FlowBundle.Norm++ ;
        FaceID ← FlowRecord.GetFace() ;
        if ( FaceID isEmpty ) then
            FaceID ← LoadBalancing.GetFace(OuputFaces) ;
        end
    else
        FaceID ← LoadBalancing.GetFace(OuputFaces) ;
    end
end
DoSendInterest(I, FaceID) ;

function ManageHit(Interest = I)
    prefix ← GetFIBPrefix(FileName(I)) ;
    FlowBundle ← GetFlowBundle(prefix) ;
    if PopularitySampler.IsPopular(Filename(I)) then
        FlowRecord ← FlowBundle.Find(FileName(I)) ;
        if FlowRecord isEmpty then
            FlowBundle.Insert(FileName(I)) ;
        end
    end
end
```

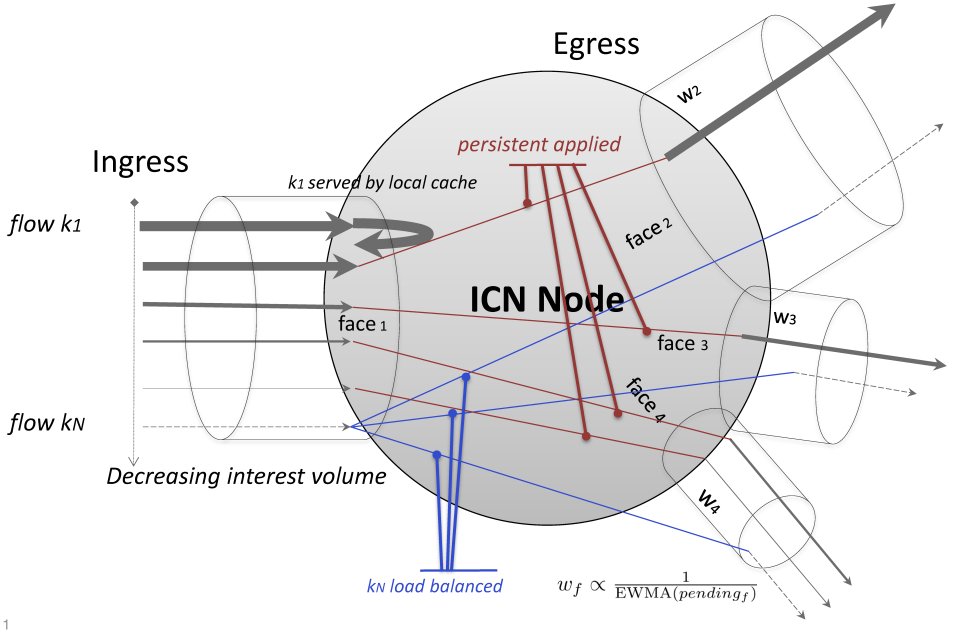


Figure 3.5 – The face selection algorithm is depicted: Load balancing with persistent face selection for popular content.

to perform online popularity estimation is out of scope of this chapter, but in our simulation we have used the a k-LRU filter [Martina et al., 2013]. In our simulations, we set each sub k-LRU cache equal to 50 objects (in the first simple scenario) or to 160 (in the other scenarii). Objects found in the last sub-cache are considered high popularity and get every of their chunk hit counted for precise flow sizing. By essence flow sizing is volatile. To stabilize it, we assume normalized flow sizes follow an unknown power law. To infer the parameters of that law, we compute the logarithm of the normalized flow sizes and fit the obtained linear model using ordinary least squares. In a more general implementation of our mechanism, we do not suggest to use k-LRU which, while being simple, requires k to be very large when popularity is measured in terms of observed traffic and not in terms of number of content item requests. However a content item (object or file) request is difficult to be identified in practice, as different clients can request the same object using distinct permutations of the chunks sequence numbers.

For a given prefix, the sample most popular items are grouped into flow bundles as reported in Algorithm 1. Each bundle contains items with consecutive popularity up to the face weight, hence the size of each bundle depends on the face weight as reported in Algorithm 1. Thus, for more popular items a single output interface is persistently selected by selecting less congested interfaces (with higher weights) first. For all other items, face selection obeys to standard LB rule, see Algorithm 2. Every ΔT seconds, the most popular items are reassigned to flow bundles according to face weights which are, on the other hand, updated independently.

3.6 Performance analysis

$n \in \mathcal{N}$	ICN node identifier. $\mathcal{N} \subseteq \mathbb{N}$.
$t \in \mathbb{R}_+$	Instant a content retrieval occurs.
$k \in \mathcal{K}$	Content popularity rank. The one ranking first is the most popular, while rank $\lvert\mathcal{K}\rvert$ indicates the least popular object.
$\Gamma^-(n)$	Set of node n's ingress nodes.
$\Gamma^+(n)$	Set of node n's egress nodes.
$Q_{k,n,b}(t)$	Size of the Pending Interest Queue for content k on link (n,b) at time t.
$\mu_{k,n,b}(t)$	Data rate for content k on link (b,n) at time t.
$\lambda_{n,k}(t)$	Exogenous interest rate for content k at node n at time t.
$\mathbb{1}_{\{.\}}$	Indicator function.
$h_{n,k}(t)$	$\mathbb{1}_{\{\text{content } k \text{ is in cache } n \text{ at } t\}}$.
$\bar{v}$	Long-term average of v. v might be either Q, μ, λ or h.
$C_{b,n}$	Link (b,n) capacity in chunks/s.

Table 3.1 – Notation.

In this section, we establish bounds to the stability region of FOCAL. Then we prove its throughput-optimal i.e., that FOCAL stabilizes an ICN network whose exogenous interest rate vector is within this region. To this aim, we follow a methodology that was first introduced in [Tassiulas and Ephremides, 1992] and previously applied to ICN in [Yeh et al., 2014]. Refer to Table 3.1 for the notation used hereinafter. Let define a Pending Interest Queue (PIQ) size as the number of pending interests per content and per face. An interest queued in a PIQ is served when the matching data packet comes back. The time evolution upper bound of the PIQ size of content k for egress nodes $b \in \Gamma^+(n)$

at node n follows:

$$Q_{k,n,b}(0) = 0, \forall b \in \Gamma^+(n) \text{ and} \tag{3.44}$$

$$\sum_{b\in\Gamma^+(n)} Q_{k,n,b}(t+\delta t) \leq \max\left\{ \sum_{b\in\Gamma^+(n)} \left(Q_{k,n,b}(t) - \mu_{k,n,b}(t)\delta t\right), 0 \right\} + \lambda_{n,k}(t)\delta t + (1 - h_{n,k}(t)) \sum_{a\in\Gamma^-(n)} \mu_{k,a,n}(t)\delta t. \tag{3.45}$$

The service rate $\mu_{k,a,b}(t)$ of the PIQ is the data rate for content k on link (b, a). $\lambda_{n,k}(t)$ is the exogenous interest rate for content k at node n.

Model assumptions

- The stochastic processes $\{\lambda_{k,n,b}(t)\}_{0\leq t\leq T}$, $\{\mu_{k,n,b}(t)\}_{0\leq t\leq T}$ and $\{h_{n,k}(t)\}_{0\leq t\leq T}$ are independent.
- The network routes a single prefix.
- Same chunk size.

First of all, we need a fundamental building block defining what it means for a PIQ to be strongly stable.

Lemma 3.3 (PIQ Stability)**.** *Let the PIQ size at time t be a sequence of i.i.d. random variables. We name $\{Q(t)\}_{0\leq t\leq T}$ this stochastic process. A PIQ is strongly stable if:*

$$\lim_{T\to\infty} \frac{1}{T^2} \int_0^T Q(t)dt \equiv \lim_{T\to\infty} \frac{\bar{Q}(T)}{T} = 0 \quad a.s. \tag{3.46}$$

In other words, a PIQ is strongly stable iif its size's time average $\bar{Q}(\cdot)$ is asymptotically dominated by the elapsed time.

Proof. Recall the strong queue stability definition in [Georgiadis et al., 2006]:

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[Q(t)] < \infty. \tag{3.47}$$

Eq.3.47 holds iif it exists a finite non-negative constant M:

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[Q(t)] = \limsup_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1} Q(t)\right] = M$$
$$\Leftrightarrow \lim_{T\to\infty} \frac{1}{T^2}\mathbb{E}\left[\sum_{t=0}^{T-1} Q(t)\right] = 0 = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T^2}\int_{\mathcal{S}} Q d\sigma\right],$$

given any measure σ on any subset $\mathcal{S} \subseteq [0,T]$ such that $\sigma(\mathcal{S}) = T$. The counting measure on the discrete subset $[0,T]\cap\mathbb{N}$ belongs here. So does the Lebesgue measure on $[0,T]$. By Lebesgue's Dominated Convergence Theorem, as

$$\frac{1}{T^2}\int_{\mathcal{S}} Q d\sigma \leq \frac{TO(T)}{T^2} < \infty,$$
$$\lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T^2}\int_{\mathcal{S}} Q d\sigma\right] = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T^2}\int_{\mathcal{S}} Q d\sigma\right] = 0.$$

Because the term inside the expectation is non-negative, we conclude that:

$$\lim_{T\to\infty} \frac{1}{T^2}\int_{\mathcal{S}} Q d\sigma = 0 \quad a.s.$$

□

We can deduce a definition of the network-wide stability. It does not differ from previous work view of the same concept.

Definition 3.2 (ICN Network Stability)**.** *An ICN network is called stable if every PIQ within is stable.*

FOCAL stability region indicates the rate at which a Pending Interest Queue can admit packets and still ensures service. Beyond this region we establish upper bounds of *i.e.*, its closure, PIQs grow infinitely until (parts of) the network collapse(s).

Proposition 3.3 (FOCAL Stability Region)**.** *The stability region for the FOCAL algorithm is characterized by the closure* Λ *of the set of all long-term average interest rates* $(\bar{\lambda}_{n,k})_{n\in\mathcal{N},k\in\mathcal{K}}$ *where:*

$$\bar{\lambda}_{n,k} \leq \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} - \sum_{a \in \Gamma^-(n)} (1 - \bar{h}^+_{n,k}) \bar{\mu}_{k,a,n}, \tag{3.48}$$

$$C_{b,n} \geq \sum_{k \in \mathcal{K}} \bar{\mu}_{k,n,b}, \quad \forall b \in \Gamma^+(n). \tag{3.49}$$

$\bar{h}^+_{n,k}$ *is the long-term cache hit ratio at node* n *for content* k *under LB-Perf,*
$\bar{\mu}_{k,a,b}$ *is content* k*'s long-term download rate on link* (b, a)*,*
$C_{b,n}$ *is link* (b, n) *capacity.*

Proof. We first reorganize terms in the queue evolution described by Eq.4.1:

$$\sum_{b \in \Gamma^+(n)} \underbrace{\frac{Q_{k,n,b}(t + \delta t) - Q_{k,n,b}(t)}{\delta t}}_{\dot{Q}_{k,n,b}(t)} \leq \lambda_{n,k}(t) - \sum_{b \in \Gamma^+(n)} \mu_{k,n,b}(t) + (1 - h_{n,k}(t)) \sum_{a \in \Gamma^-(n)} \mu_{k,a,n}(t).$$

Through the invocation of the Fundamental Theorem of Calculus, we have:

$$\begin{aligned}
\sum_{b \in \Gamma^+(n)} \lim_{T \to \infty} \frac{1}{T^2} \int_0^T Q_{k,n,b}(t) dt \leq & \lim_{T \to \infty} \frac{1}{T^2} \int_0^T \int_0^t \lambda_{n,k}(u) du dt \\
& - \sum_{b \in \Gamma^+(n)} \lim_{T \to \infty} \frac{1}{T^2} \int_0^T \int_0^t \mu_{k,n,b}(u) du dt \\
& + \sum_{a \in \Gamma^-(n)} \lim_{T \to \infty} \frac{1}{T^2} \int_0^T \int_0^t (1 - h_{n,k}(u)) \, \mu_{k,a,n}(u) du dt \\
\Leftrightarrow \sum_{b \in \Gamma^+(n)} \lim_{T \to \infty} \frac{\bar{Q}_{k,n,b}(T)}{T} \leq & \bar{\lambda}_{n,k} - \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} + \sum_{a \in \Gamma^-(n)} (1 - \bar{h}_{n,k}) \bar{\mu}_{k,a,n}.
\end{aligned}$$

From Lemma 3.3, PIQ instability occurs if the above ratio of the long-term average of PIQ size over the elapsed time is strictly positive, meaning that:

$$\bar{\lambda}_{n,k} - \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} + \sum_{a \in \Gamma^-(n)} (1 - \bar{h}_{n,k}) \bar{\mu}_{k,a,n} > 0.$$

Conversely and consequently, PIQ stability requires that:

$$\bar{\lambda}_{n,k} \leq \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} - \sum_{a \in \Gamma^-(n)} (1 - \bar{h}_{n,k}) \bar{\mu}_{k,a,n}. \tag{3.50}$$

However, the proof is not complete since FOCAL forwards the most popular contents in a single-path manner at long-term average rate $\bar{\mu}^{\text{perf}}_{\cdot,\cdot,\cdot}$. In formal terms, $\forall n \in \mathcal{N}, \exists K \in \mathcal{K} : \forall k \leq K$,

$$\exists b \in \Gamma^+(n) : \bar{\lambda}_{n,k} \leq \bar{\mu}^{\text{perf}}_{k,n,b} - \sum_{a \in \Gamma^-(n)} (1 - \bar{h}_{n,k}) \bar{\mu}_{k,a,n}. \tag{3.51}$$

Thus, it remains to show that FOCAL's Eq.3.51 preserves the stability region defined in Eq.3.50:

$\sum_{b \in \Gamma^+(n)} \sum_{k} \bar{\mu}_{k,n,b} \leq \sum_{b \in \Gamma^+(n)} C_{b,n}$ is the total data rate satisfying node n's egress traffic. Simply by the associative and commutative properties of addition in $\mathbb{R}$, we rearrange the l.h.s into bundles B_b such that $\exists K \in \mathcal{K} : \{B_b, \forall b \in \Gamma^+(n)\}$ forms a partition of $\{1, \cdots, K\}$ i.e., $\bigcup_{b \in \Gamma^+(n)} B_b = \{1, \cdots, K\}$ and $\bigcap_{b \in \Gamma^+(n)} B_b = \varnothing$ while $\forall b \in \Gamma^+(n)$, $\sum_{k \in B_b} \bar{\mu}^{\text{perf}}_{k,n,b} \leq C_{b,n}$, as follows:

$$\begin{aligned}
\sum_{b \in \Gamma^+(n)} \sum_{k} \bar{\mu}_{k,n,b} = \sum_{k} \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} &= \sum_{k \leq K} \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} + \sum_{k > K} \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} \\
&= \sum_{b \in \Gamma^+(n)} \sum_{k \in B_b} \sum_{l \in \Gamma^+(n)} \bar{\mu}_{k,n,l} + \sum_{k > K} \sum_{b \in \Gamma^+(n)} \bar{\mu}_{k,n,b} \\
&= \underbrace{\sum_{b \in \Gamma^+(n)} \sum_{k \in B_b} \bar{\mu}^{\text{perf}}_{k,n,b}}_{\text{Perf}} + \underbrace{\sum_{k > K} \sum_{b \in \Gamma^+(n)} \bar{\mu}^{\text{LB}}_{k,n,b}}_{\text{LB}}.
\end{aligned} \tag{3.52}$$

We see that FOCAL's forwarding strategy, denoted LB-Perf, rearranges a few egress flows within bundles and load-balances others by achieving optimal rates $\mu^{\text{LB}}_{k,n,b}(t)$ in a way that fully preserves the flow conservation constraints [Carofiglio et al., 2013c].

Furthermore, as stated in Lemma 3.2, any cache's hit ratio $\bar{h}_{n,k}$ increases when the size of the catalog seen by node n decreases. Precisely, Persistent Forwarding

reduces the size of the catalog seen by every node, increasing the hit ratio to $\bar{h}^{+}_{n,k} = \bar{h}_{n,k} + \epsilon_{n,k}$, $\epsilon_{n,k} \geq 0$. It follows that,

$$\exists \epsilon'_{n,k} \geq 0 : \bar{\lambda}_{n,k} + \epsilon'_{n,k} \leq \sum_{b \in \Gamma^{+}(n)} \bar{\mu}_{k,n,b} - \sum_{a \in \Gamma^{-}(n)} (1 - \bar{h}^{+}_{n,k}) \bar{\mu}_{k,a,n}, \tag{3.53}$$

which completes the proof of FOCAL stability. □

In practice, FOCAL's persistent forwarding does not know link capacities $C_{n,b}$. It rather exploits egress face weights $w_{n,b}$ estimated from related counters of pending interests, and considers it equals to normalized link capacities $\tilde{C}_{n,b}$. However, $w_{n,b}$ get overestimated for higher capacity links as the egress load increases. This is because these weights reflect average round-trip times [Carofiglio et al., 2013c]. To better understand this bias, observe that:

$$w_{n,b} \equiv \frac{\sum_{l \in \Gamma^{+}(n)} (\sum_{k} \bar{Q}_{k,n,l})^{-1}}{\sum_{k} \bar{Q}_{k,n,b}} \approx \frac{C_{b,n} - M_n}{\sum_{l \in \Gamma^{+}(n)} (C_{l,n} - M_n)}$$

after approximating per-face sets of PIQs with $M/G/1$-PS queues (like a per-face PIT), where $M_n \leq \inf_{l \in \Gamma^{+}(n)} \{C_{l,n}\}$ is the egress (miss) traffic rate on node n. It is clear that $w_{n,b} \sim \tilde{C}_{n,b}$ as $M_n \downarrow 0$. On the other hand, since

$$\frac{\partial w_{n,b}}{\partial M_n} = \frac{|\Gamma^{+}(n)| C_{b,n} + \sum_{l \in \Gamma^{+}(n)} C_{l,n}}{\left(\sum_{l \in \Gamma^{+}(n)} (C_{l,n} - M_n) \right)^2} M_n,$$

the bigger M_n and the link capacity the bigger the weight inflation. Such a bias results in over-provisioned flow bundles on the most capacited links and congestion. As FOCAL's caching mechanism, LAC+, is capable by design of alleviating congestion by driving the oversize bundle's items into the cache, so it does. Nevertheless, this somewhat tactical solution uses caching (LAC+) to fix a forwarding issue (LB-Perf weight estimation). It is by far better to design the weight estimation such that LB-Perf conforms autonomously to the model in Proposition 3.3. Again, assuming the $M/G/1$-PS queuing model holds, we can remove the bias on the weight estimation made from the number of pending interests just by adding 1 to the latter's inverse. No knowledge of the egress traffic rate M_n is necessary. Hence, while load balancing weights remain $w_{n,b}$ as estimates of the available

throughput, the unbiased persistent forwarding weights become:

$$w_{n,b}^{\text{perf}} = \frac{C_{b,n}}{\sum_{l\in\Gamma^+(n)} C_{l,n}} \approx \frac{\left(\sum_k \bar{Q}_{k,n,b}\right)^{-1} + 1}{\sum_{l\in\Gamma^+(n)} \left(\left(\sum_k \bar{Q}_{k,n,l}\right)^{-1} + 1\right)}. \tag{3.54}$$

The following proposition states that, as long as the exogenous interest rates are within the closure Λ, FOCAL dynamically stabilizes the whole ICN network whenever some PIQ grows too large by offloading it. The argument lies in the Lyapunov drift analysis technique.

Proposition 3.4 (FOCAL Throughput Optimality)**.** *Under FOCAL, assuming ergodic data rates and hit ratios over time, and within the stability region defined in Proposition 3.3, there exists a couple of strictly positive constants B and ϵ, such that the Lyapunov drift over the vector process of PIQ sizes $\boldsymbol{Q}(t) \equiv (Q_{k,n,b}(t))_{k\in\mathcal{K},n\in\mathcal{N},b\in\Gamma^+(n)}$,*

$$\Delta(\boldsymbol{Q}(t)) \leq B - \epsilon \sum_{n\in\mathcal{N},k\in\mathcal{K},b\in\Gamma^+(n)} Q_{k,n,b}(t). \tag{3.55}$$

This means that the above drift becomes negative beyond a given cumulative PIQ size, leading to overall stability.

Proof. Choose the Lyapunov function

$$\mathcal{L}(\mathbf{Q}(t)) = \frac{1}{2} \sum_{n\in\mathcal{N},k\in\mathcal{K}} \left[\sum_{b\in\Gamma^+(n)} |Q_{k,n,b}(t)|\right]^2. \tag{3.56}$$

By following the steps sketched in [Georgiadis et al., 2006], the Lyapunov drift is:

$$\Delta(\mathbf{Q}(t)) \equiv \mathbb{E}[\mathcal{L}(\mathbf{Q}(t+1)) - \mathcal{L}(\mathbf{Q}(t)) \mid \mathbf{Q}(t)] \tag{3.57}$$

$$\leq B + \sum_{n\in\mathcal{N},k\in\mathcal{K}} \bar{\lambda}_{n,k} \sum_{b\in\Gamma^+(n)} Q_{k,n,b}(t) - \sum_{n\in\mathcal{N},k\in\mathcal{K},b\in\Gamma^+(n)} Q_{k,n,b}(t)$$

$$\times \mathbb{E}\left[\sum_{j\in\Gamma^+(n)} \mu_{k,n,j}(t) - (1 - h_{n,k}^+(t)) \sum_{i\in\Gamma^-(n)} \mu_{k,i,n}(t) \mid \mathbf{Q}(t)\right] \tag{3.58}$$

$$\text{with } B = \frac{1}{2}\sum_{n}\Bigg[\Big(\sum_{b\in\Gamma^{+}(n)} C_{b,n}\Big)^{2} + \Big(\sum_{k\in\mathcal{K}}\bar{\lambda}_{n,k} + 2\sum_{a\in\Gamma^{-}(n)} C_{n,a}\Big)^{2} + 2\sum_{a\in\Gamma^{-}(n)} C_{n,a}\sum_{b\in\Gamma^{+}(n)} C_{b,n}\Bigg]. \tag{3.59}$$

Since the random variables within the conditional expectation's brackets are from ergodic processes and independent of PIQ sizes, we can replace the conditional expectation term in Eq.3.58 with Eq.3.53 , which defines FOCAL's stability region. To complete the proof, we choose $\epsilon = \inf_{n\in\mathcal{N},k\in\mathcal{K}}\{\epsilon'_{n,k}\}$. □

Corollary 3.4.1. *Proposition 3.4 holding, mean PIQ sizes are upper bounded as follows:*

$$\limsup_{T\to\infty} \sum_{n\in\mathcal{N},k\in\mathcal{K},b\in\Gamma^{+}(n)} \frac{1}{T}\int_{0}^{T} \mathbb{E}[Q_{k,n,b}(t)]dt \leq \frac{B}{\epsilon}. \tag{3.60}$$

Proof.

$$\begin{aligned}
\epsilon \sum_{n\in\mathcal{N},k\in\mathcal{K},b\in\Gamma^{+}(n)} \int_{0}^{T} \mathbb{E}[Q_{k,n,b}(t)]dt &\leq BT - \int_{0}^{T}(\mathbb{E}[\mathcal{L}(\mathbf{Q}(t+1))] - \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))])dt \\
&= BT - \int_{1}^{T+1} \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))]dt + \int_{0}^{T} \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))]dt \\
&= BT - \int_{T}^{T+1} \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))]dt + \int_{0}^{1} \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))]dt \\
&\leq BT + \int_{0}^{1} \mathbb{E}[\mathcal{L}(\mathbf{Q}(t))]dt.
\end{aligned}$$

Since the Lyapunov function $\mathcal{L}$ is zero at $\mathbf{Q}(0) = \vec{0}$ and its increase is bounded in a finite time slot,

$$\limsup_{T\to\infty} \sum_{n\in\mathcal{N},k\in\mathcal{K},b\in\Gamma^{+}(n)} \frac{1}{T}\int_{0}^{T} \mathbb{E}[Q_{k,n,b}(t)]dt \leq \frac{B}{\epsilon}.$$

□

Observation 3.2 (FOCAL Complexity)**.** *FOCAL average complexity is*

$$O(PopularFiles.Size()).$$

Find a wider landscape in Table 3.2. To reduce LAC+ space complexity in a worst case

Algorithm	Complexity
Uniform forwarding	Constant.
LB forwarding	Constant using a hash table.
LB-Perf forwarding	$O(PopularFiles.Size())$.
LCE, LCP, LCD, LAC	Constant in average using a hash table.
LAC+	Constant in average using a hash table. Catalog size in worst case, when the biggest outlier's mean latency decreases.

Table 3.2 – Algorithmic complexity per packet.

scenario, the number of content items whose average latency is tracked might be upper bounded by $k^{max} < |\mathcal{K}|$. In that way, when the hash table of average latencies reaches its maximum size, items k whose mean latencies $\bar{T}_k(t)$ are less than the z^{th} quantile $Q_z(t)$ might be deleted. Their mean latency will be assumed to equal the average μ_t.

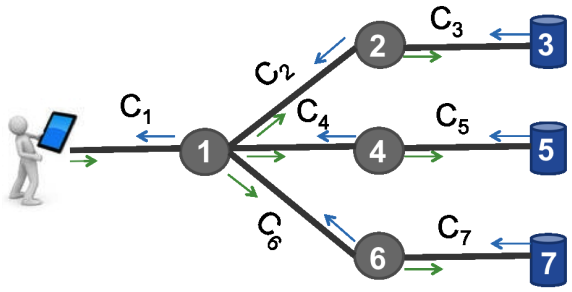
3.7 Simulation

In this section we assess the performance of FOCAL by means of ICN simulations in three different scenarii and against existing forwarding and caching alternatives given by the combination of the following known caching policies:

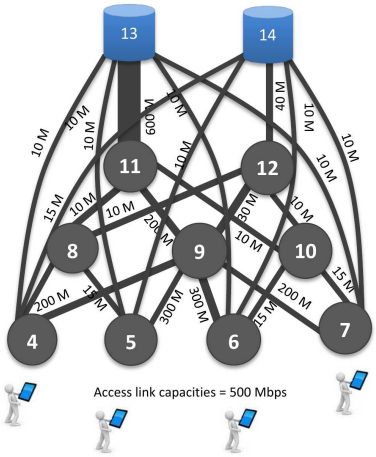
- *LRU*, Least Recently Used: deterministic cache insertion of every item arriving at the cache coupled with LRU replacement,
- ϵ-*LCP*, Leave a Copy Probabilistically: a probabilistic cache insertion with probability ϵ ($\epsilon = 10^{-3}$ where not specified) coupled with LRU replacement,
- *LCD*, Leave a Copy Down: a content object retrieved at $l-th$ cache along a path is cached at $(l-1)$-th cache only, rather that in all caches from 1 to $l-1$ ([Laoutaris et al., 2006]),
- *LAC*, Latency-Aware Caching: the approach proposed in [Carofiglio et al., 2015c],
- *LAC+*, enhanced LAC: our approach presented in Sec.3.5.1.0. The default quantile z is 1.

and forwarding strategies:

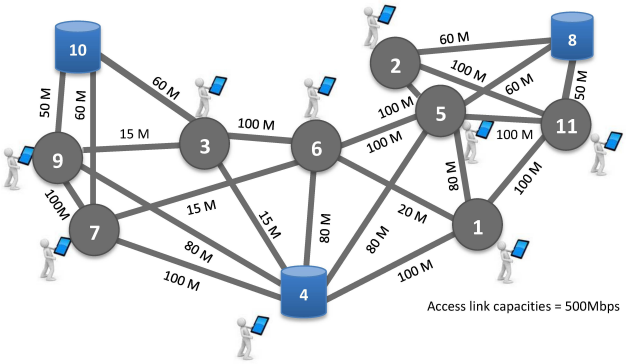
- *Uniform*: uniform selection performed on per-packet basis of output interfaces stored in FIB entries aggregated per-prefix;
- *LB*, Load-Balancing: load-aware selection performed on nearly per-packet basis of output interfaces stored in FIB entries as designed in [Carofiglio et al., 2013c].
- *LB-Perf*, Load-Balancing with Persistent forwarding: our approach presented in Sec.3.5.2. The approach combining LB-Perf with LAC+ caching is denoted as FOCAL.



(a) Linear topology with forwarding branches.



(b) Fat tree with direct links to repositories.



(c) Abilene-like topology.

Figure 3.6 – Network topologies used in the evaluation.

To this purpose, we implement FOCAL and its alternatives in the packet-level NDN simulator CCNPL-Sim (`http://systemx.enst.fr/ccnpl-sim`). It benefits for congestion/rate control from Remote Adaptive Active Queue Management (RAAQM) thoroughly described in [Carofiglio et al., 2013c]. Three topologies are considered: a linear topology with forwarding branches, a hierarchical fat tree with direct access to content repositories and a non-hierarchical meshed topology (Abilene-like).

3.7.1 Linear topology with forwarding branches

Forwarding / Caching	Uniform			LB			LB-Perf		
	Avg	StdDev	T_{stat}	Avg	StdDev	T_{stat}	Avg	StdDev	T_{stat}
LRU	0.98	1.53	20h	0.78	0.77	3h	0.57	0.57	10h
p-LCP	0.43	0.84	>55h	<0.4	0.6	>55h	0.32	0.43	>55h
LCD	0.5	1	4h	0.48	0.82	3h	0.4	0.7	3h
LAC	0.47	0.92	>55h	<0.4	0.65	>55h	0.31	0.42	>55h
LAC+	0.51	0.93	12h	0.47	0.65	9h	0.31	0.37	9h

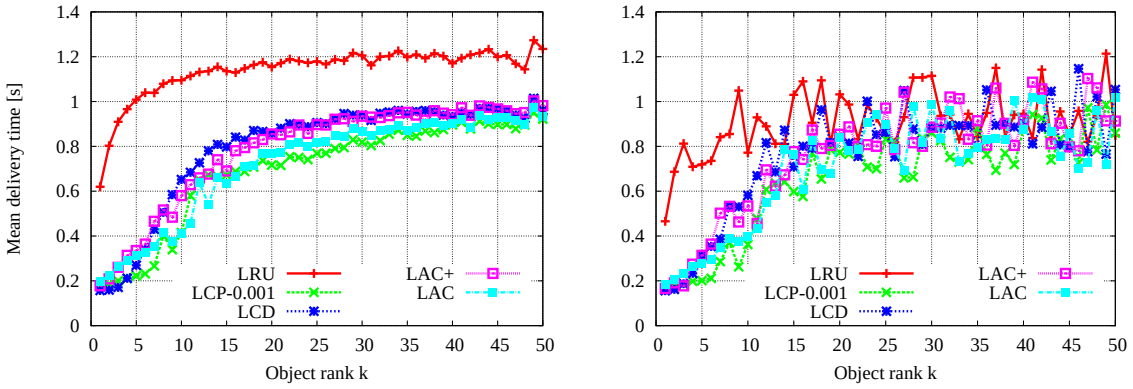
FOCAL

Figure 3.7 – Linear topology with forwarding branches, cache cap. = 10: Steady state values.

We simulate the simple topology in Fig.3.6(a) to show: *i)* the improvement of latency-aware caching policies in presence of random forwarding, *i.e.*, without latency-awareness in link selection; *ii)* the interaction with forwarding strategies and overall superiority of FOCAL. The tests consist in a branched network of ICN nodes capable of storing up to 10 content items per cache. We simulate a 55-hour traffic involving a single content producer that serves a catalog of 20,000 objects. Popularity is Zipf-like distributed with parameter $\alpha = 0.9$. It implies that the ten most popular items weight 19% of the traffic. Each content item is conveyed in chunks of 3kB and has a total size of 2MB (the same size is

used in all scenarios presented in the chapter). In all simulations reported in the chapter, data retrieval is managed by an implementation of the transport protocol presented in [Carofiglio et al., 2013c].

Link capacities are limited to C_1=600Mbps, C_2=60Mbps, C_3=20Mbps, C_4 = 100Mbps, C_5=30Mbps, C_6=300Mbps, C_7=50Mbps. The object request process feeding node 1 is assumed to be Poisson with rate parameter $\lambda = 3$ objects/s. The maximum long term link load in the network does not exceed 50% of utilization. In Fig.3.8(a), we neglect the impact of forwarding, by considering a uniform selection of the three output interfaces at node 1. All caching policies aim in a more or less effective way at caching the most popular items on the first cache, the following ones in terms of more popular items at the second level of caches (2,4,6).



(a) Uniform: cache capacity = 10. (b) Load Balanced (LB): cache cap. = 10.

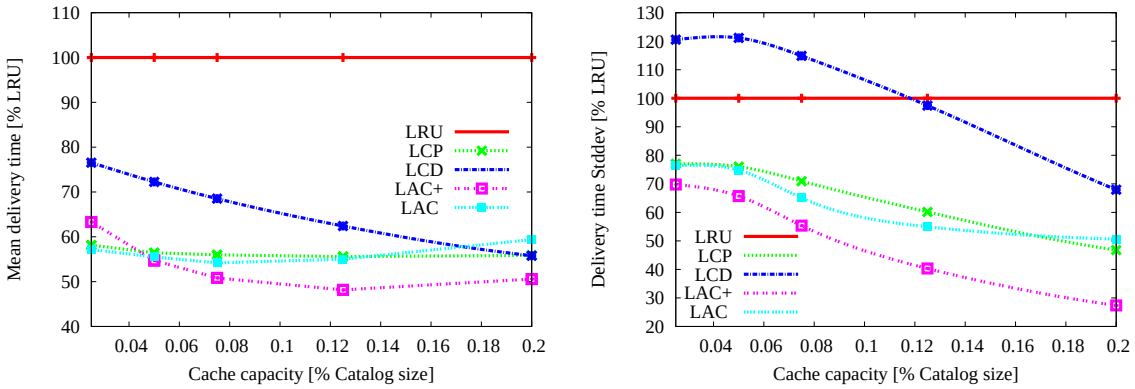
Figure 3.8 – Linear topology with forwarding branches: (a)-(b) Mean delivery time w.r.t content rank.

Except for LRU which under-performs, the other policies show similar results. Clearly, the second level caches work independently under uniform forwarding, because they receive the same arrival process sampled at $1/3$ of the total request rate. This still happens in presence of LB forwarding (Fig.3.8(b)) due to the blind load-aware forwarding of packets over the three interfaces. The benefits in terms of latency reduction deriving from load distribution across the three paths appear to be negligible compared to the lack of implicit cache coordination. A significant reduction up to 40% in average delivery time over the entire catalog can be observed under LB-Perf w.r.t. uniform forwarding and particularly for FOCAL (LAC+ with LB-Perf) as shown in Fig.3.15(c). When we dig a little more

into FOCAL internals at Node 1, we observe, as depicted in Fig.3.15(d), that the logarithm of normalized forwarded interest count decreases linearly with the logarithm of the rank. The slope is the local popularity skewness. In the top of that figure, we see that the resulting popularity estimation and face weight estimations are stable enough for FOCAL to persistently forward a majority of flows through the same faces.

Overall, the values reported in Fig.3.7 allow to quantify in 48% the gains brought by caching alone (LAC+) in terms of average delivery time reduction and in another 40% reduction via LB-Perf. FOCAL also achieves the lowest variance with good convergence time when compared to other approaches (absolute values of convergence are due to the slow request rate considered in simulations).

Impact of cache capacity on LB-Perf



(a) LB-Perf: Mean vs cache capacity.

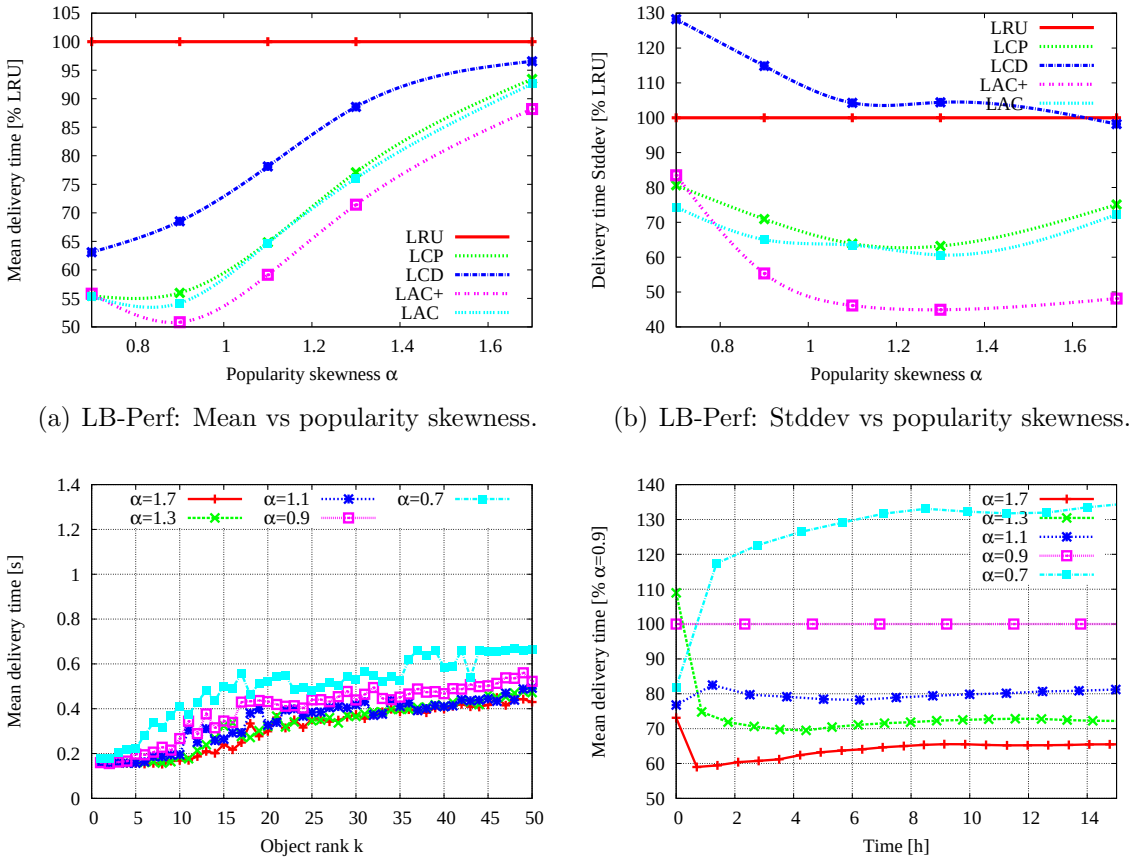
(b) LB-Perf: Stddev vs cache capacity.

Figure 3.9 – Linear topology with forwarding branches: Normalized delivery time vs cache capacity.

We evaluate LB-Perf under various cache capacities: 5, 10, 15, 25, 40 objects. Per-popularity ranks results can be found in Fig.3.15. Obviously, the larger the cache, the smaller the mean delivery time thanks to less evictions from caches. Ultimately, when the cache capacity reaches 40 objects, which is 0.2% of the catalog size, the egress traffic from the closest cache to the client node becomes so undifferentiated that it gets merely load balanced. Overall results in Fig.3.9 complement the picture. They are normalized to help infer more general knowledge from the observations. On the x-axis, the cache capacity is expressed as a percentage of the whole catalog ($20,000$ objects), ranging from 0.025% to

0.2% of the catalog. The mean delivery time and standard deviation are expressed on y-axes as percentages of those achieved by the basic LRU algorithm. It is remarkable that up to 0.2% of the catalog size, FOCAL increasingly reduce LB-Perf+LRU's mean delivery time by 35% to 50%. In the meantime the larger the cache, the smaller the normalized delivery time standard deviation (up to 72% reduction). It instills the idea that designing a proper tandem of caching and forwarding algorithms like FOCAL is extremely rewarding unless one can afford prodigal cache budgets.

Impact of the popularity skewness on LB-Perf



(a) LB-Perf: Mean vs popularity skewness.

(b) LB-Perf: Stddev vs popularity skewness.

(c) FOCAL: Mean vs Rank.

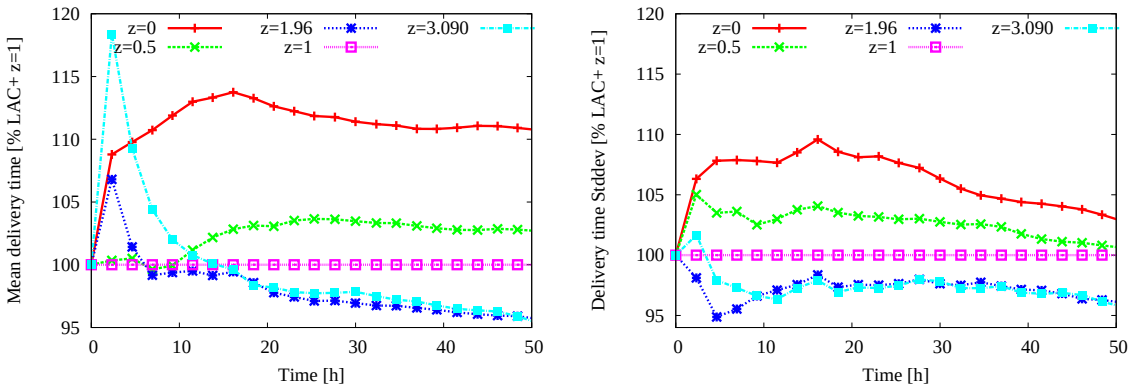
(d) FOCAL: Mean vs Time.

Figure 3.10 – Linear topology with forwarding branches: Impact of popularity skewness on content delivery time.

We evaluated LB-Perf under the following Zipf-like α parameter values: 0.7, 0.9, 1.1,

1.3 and 1.7. This parameter characterizes the skewness of the content popularity distribution. Obviously, content mean delivery time decreases as popularity skewness increases: see FOCAL's case in Fig.3.10(d). This is because the weight of the most popular contents, which are quickly available since often in cache, increases with the popularity skewness. While comparing caching algorithms in Fig.3.10(a), it appears clearly that LAC+ outperforms others, particularly at $0.9 \leq \alpha < 1.1$ where FOCAL cuts by 50% LB-Perf+LRU's mean delivery time. As the popularity skewness increases, the competitive advantage of FOCAL becomes its ability to stabilize the delivery time, quantified in Fig.3.10(b) by its minimization of the delivery time standard deviation.

Impact of the quantile z on FOCAL



(a) FOCAL: Mean vs Time.

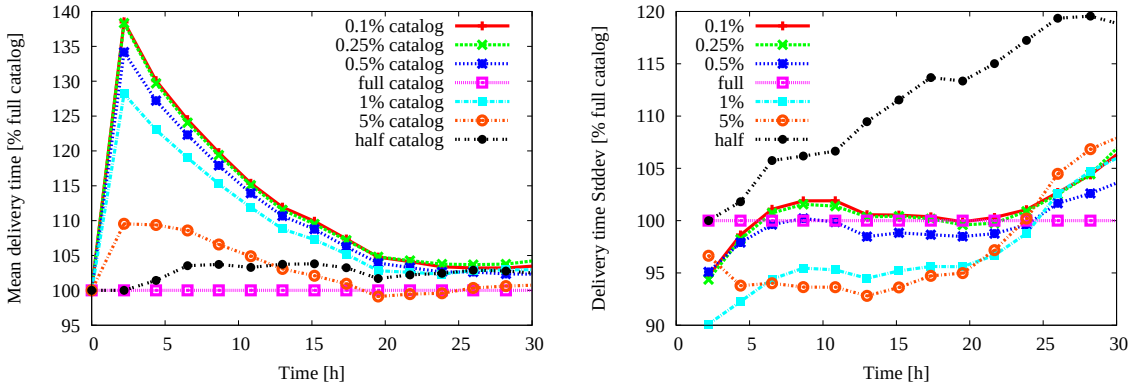
(b) FOCAL: Stddev vs Time.

Figure 3.11 – Linear topology with forwarding branches: Impact of LAC+ quantile z on FOCAL normalized delivery time evolution.

In LAC+, the probability to store an object into the cache relies on how far the object latency is from the z^{th} quantile of the distribution of average content latencies. We led a sensitivity analysis on this parameter z. Five z-values have been evaluated: 0, 0.5, 1 (the default), 1.96 and 3.090. The non-zero z-values assuming a Gaussian distribution of average content latencies represent 38%, 68%, 95% and 99.8% of the content items. Setting $z = 0$ means to calculate the probability that an object is an outlier from its distance to distribution mean. The results depicted in Fig.3.11 show that $z = 0$ is by far the worst choice as it fails to capture exceptional events. It degrades reference figures, $z = 1$, by 10%. Conversely $z = 1$ is the best compromise between delivery time reduction

and convergence speed. The latter is actually critical in case of time locality. Otherwise, when the workload is stationary, $z = 1.96$ might be considered, as it improves LAC+ by 5%.

Impact of the VRTT hash table size on FOCAL



(a) FOCAL: Mean vs Time.

(b) FOCAL: Stddev vs Time.

Figure 3.12 – Linear topology with forwarding branches: Impact of LAC+ mean VRTT hash table size on FOCAL.

Observation 3.2 deals with the complexity of the caching and forwarding algorithms we study in this chapter. As LAC+'s mean VRTT container may be implemented using an hash table, it implies a constant complexity (in average) of accessing mean VRTTs. However, the algorithm requires the search for the supremum of mean VRTTs, which is of constant complexity as long as the current supremum's mean value does not decrease. Indeed, the simplest way to catch it is to compare the most recently updated mean VRTT with the current supremum and to make it the new supremum if greater. On the other hand, when an update to that supremum decreases it, it might no longer be the biggest outlier. Then a full search into the whole hash table becomes mandatory. Hence, the size of the hash table becomes critical as it drives LAC+ worst case complexity.

This test evaluates several hash table sizes to appreciate FOCAL sensitivity to truncated hash tables of mean VRTTs. Results are depicted in Fig.3.12. They show that implementing LAC+ over a hash table of mean VRTTs whose maximum size is 5% of the whole catalog initially degrades FOCAL mean delivery time by only 10%, then converges to the full catalog performance. In general, it is not suitable to size hash tables to less than

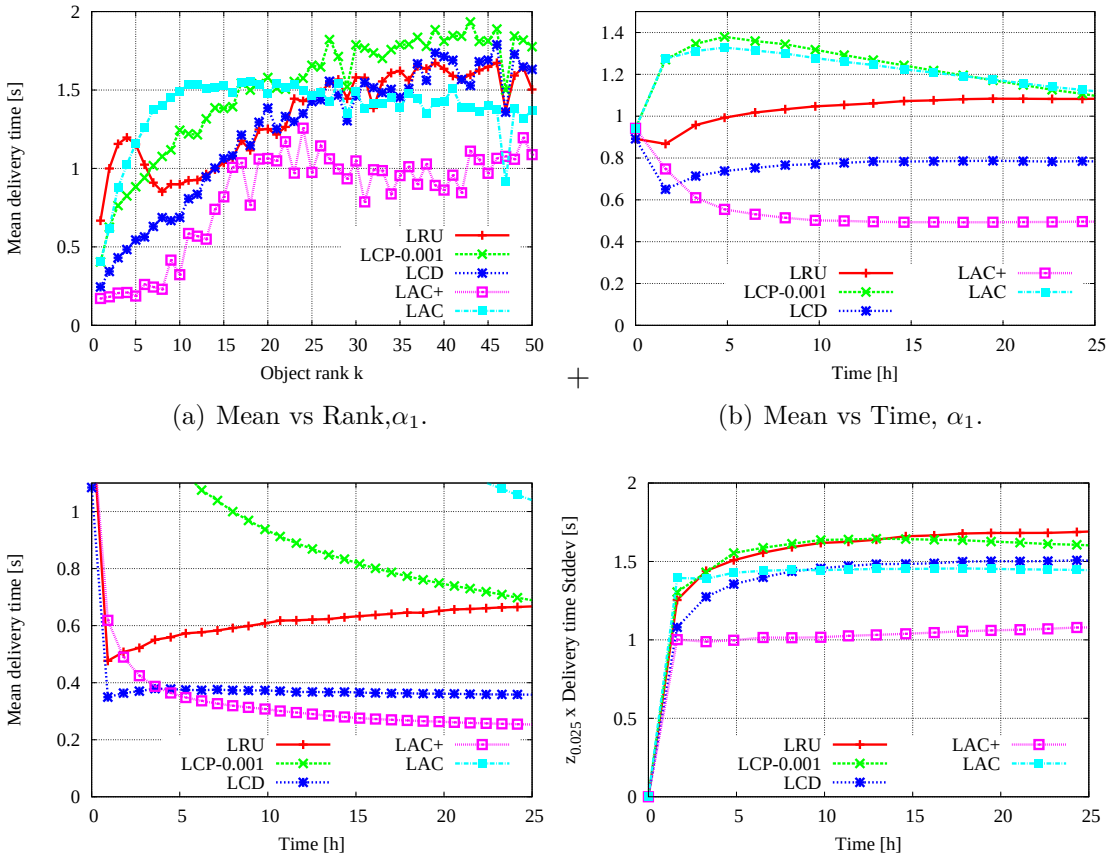
1% of the full catalog. Another interesting observation is that the delivery time standard deviation measured after hash table truncation, even though degraded at mean delivery time convergence, does not exceed 1.2 times the full catalog figure.

3.7.2 Fat tree with direct access to content repositories

We include this scenario to study FOCAL behavior in hierarchical networks with several paths with or without in-path caching opportunities. We consider the fat tree topology in Fig.3.6(b), with caches in every node storing 40 objects each. Content requests follow a Poisson process with intensity $\lambda = 1$ object/s. They uniformly address two repositories (Node 13 and 14 in Fig.3.6(b)), each one hosting a distinct catalog of $20,000$ objects, under the prefixes */Orange/* and */YouTube/*, ranked according to the same Zipf distribution. Two popularity profiles are considered in independent simulation runs: Zipf's skewness $\alpha_1 = 0.9$ and $\alpha_2 = 1.1$. LB-Perf is performed without popularity log-log linear fitting. To appreciate the impact on forwarding/caching, under Zipf's α_1 , the 400 most popular content items account for 50% of the traffic. This number drops to 30 with α_2. The presence of 10Mbps direct links to the repositories enriches the set of available paths, by making caching opportunistic: a node may choose not to forward Interests through the network of caches, rather to use the auxiliary direct link. Such configuration permits to understand whether and for which part of the catalog, in-network caching can be important to reduce end-user latency.

We report performance measures for this scenario in Fig.3.13. In this setting, FOCAL proves to outperform all other mechanisms under different metrics: it provides the best delivery time in average and standard deviation, attained within few hours. Such time scale is a typical busy period in access networks where caching performance would be mostly solicited in practice.

FOCAL is also robust to different workloads (here represented by two α factors) in contrast to other mechanisms like ϵ-LCP that fail to provide an acceptable performance bound. Indeed, when $\alpha = 1.1$ the average delivery time of ϵ-LCP converges two times slower to an almost two times higher value than that achieved by FOCAL. When $\alpha = 0.9$, ϵ-LCP provides more than two times higher latency than FOCAL, with very poor convergence time. On the other hand, looking at the way content items are managed by the different mechanisms, we observe a significant performance improvement for highly



(a) Mean vs Rank,α_1.

(b) Mean vs Time, α_1.

(c) Mean vs Time, α_2.

(d) Stddev vs Time, α_1.

Figure 3.13 – Fat tree topology, $\alpha_1 = 0.9, \alpha_2 = 1.1$

popular items, attaining gaims of a factor 5 when $\alpha = 0.9$ as reported in Fig.3.13(a).

3.7.3 US backbone-like scenario

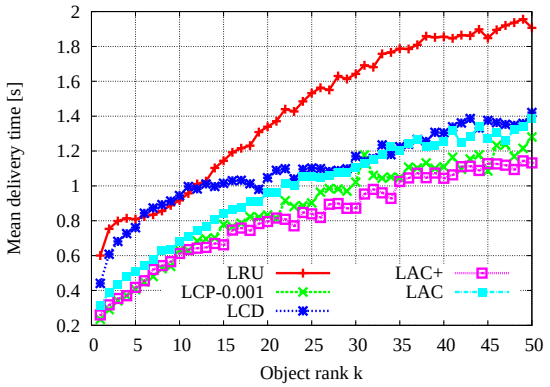
A backbone-like topology is made of core nodes which are access gateways to all clients attached to it (we build upon Abilene topology). Routing is much less hierarchical when compared to previous scenarios and Interest/Data traffic can flow in any direction. In such a setup, node cache size is equal to 40 content objects. Links in the access are set to 500Mbps, and from 15Mbps to 100Mbps in the core. In this scenario we use three content repositories, each containing $20,000$ objects. We give objects at Repository 4 the prefix */Netflix/*, Repository 8 the prefix */Orange/* and Repository 10 */YouTube/*. Clients are

equally interested in every catalog, *i.e.*, every client addresses every catalog with probability $1/3$. Client requests follow a Poisson process with intensity $\lambda = 2$ objects/s. Each repository (also referred to as producer) is queried by clients following a Zipf-distributed workload with skewness parameter equal to 0.9. Fig.3.6(c) summarizes the details of the network setup. While the content ranks have been so far fixed for the whole simulation, in this set of simulations, we confront the algorithms to a non-stationary content popularity distribution to introduce time locality. This is obtained by shuffling popularity rank every ten hours for every object in a given catalog. Every content's popularity rank changes over time while ranks remain Zipf-distributed. Note that this is far from being unrealistic. It widely pertains to real-world traffic where per-time-slot content popularity prevails [Imbrenda et al., 2014]. LB-Perf is performed without popularity log-log linear fitting.

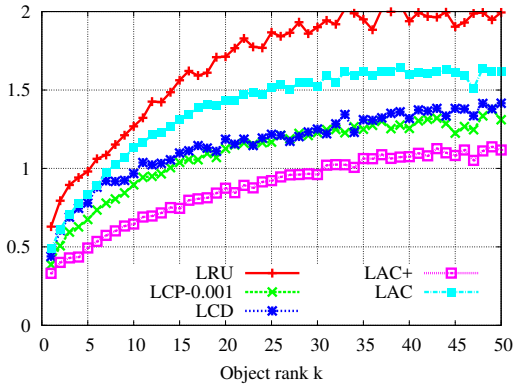
FOCAL clearly outperforms other algorithms with or without temporal locality in the request workload. From Fig.3.14(b) to 3.14(d), delivery time performance results demonstrate that it improves LCD and ϵ-LCP ($\epsilon = 10^{-3}$) almost as much as they improved the basic LRU policy. More striking, FOCAL reduces LB-Perf + LRU average delivery time by 50% and stabilizes the delivery time in reducing its standard deviation by 60%. By comparing Fig.3.14(a) to Fig.3.14(b) we observe that FOCAL catches temporal locality very well. This is due to the fact that the mechanism adapts very fast to new conditions both in terms of popularity and network congestion.

3.8 Conclusion

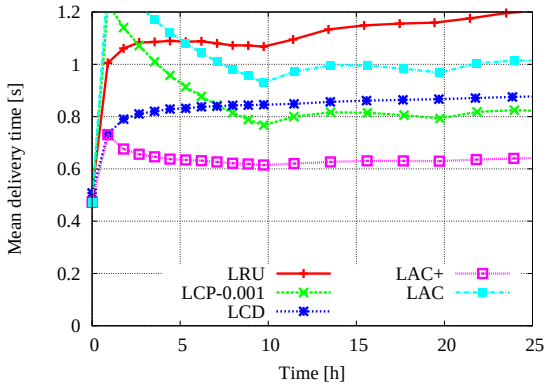
The chapter explores ICN techniques for end-user delay minimization via latency-aware forwarding and caching strategies. Based on the insights on latency-aware caching alone and on the interplay with load balancing forwarding, we introduce FOCAL, an approach combining novel caching and forwarding strategies to jointly reduce end-user experienced latency with no network signaling nor coordination between routers. FOCAL combines a latency-proportional probabilistic caching policy, with a load-aware dynamic forwarding strategy, that preferentially routes popular content requests through a single path (set of caches), while globally achieving minimum load, thus delay minimization. By means of NDN/CCN simulations, we show that our proposal may achieve significant latency reduction (e.g. up to 60% average/variance delay reduction over LRU with LB-Perf in Abilene-like scenario), coupled with faster convergence w.r.t. solutions based on probabilistic



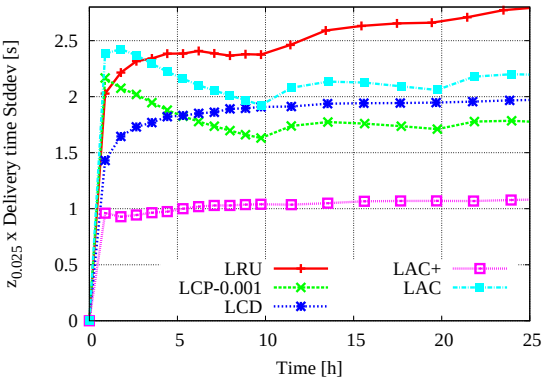
(a) Mean vs Rank, stationary.



(b) Mean vs Rank, non stationary.



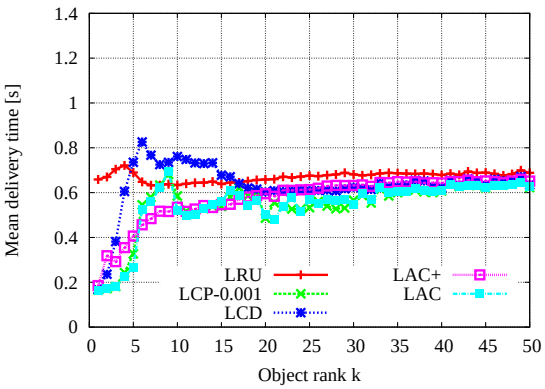
(c) Mean vs Time, non stationary.



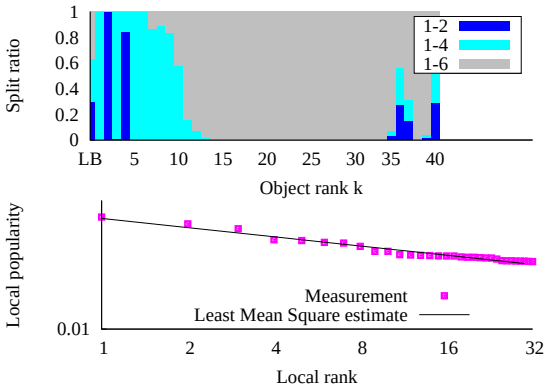
(d) Stddev vs Time, non stationary.

Figure 3.14 – Abilene-like topology with stationary (s) and non-stationary (ns) workload.

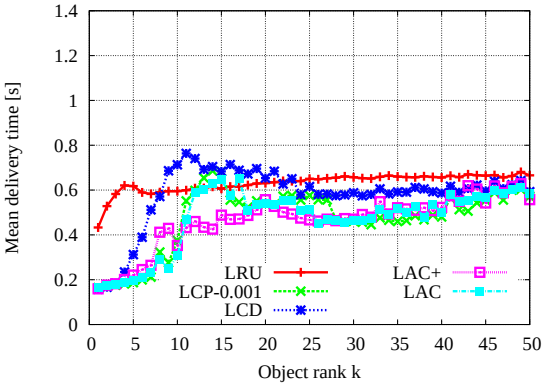
caching approaches. In presence of non-stationary phenomena, FOCAL outperforms all other approaches, demonstrating high self-adaptiveness to varying traffic/network conditions.



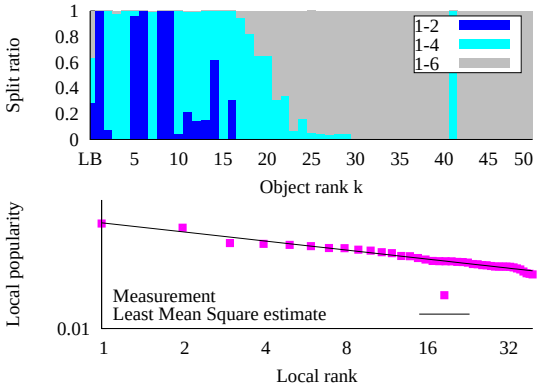
(a) LB-Perf: cache capacity = 5.



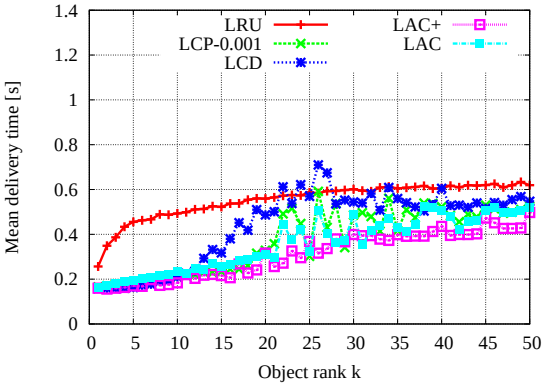
(b) FOCAL egress traffic: cache cap. = 5.



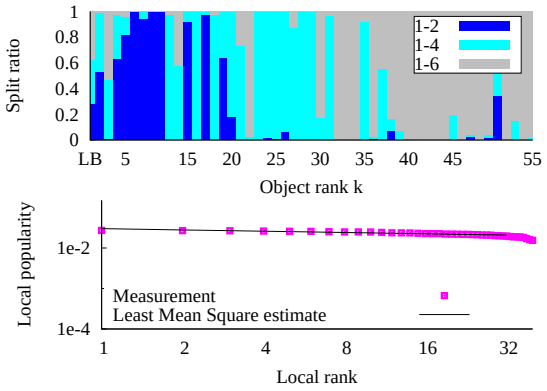
(c) LB-Perf: cache capacity = 10.



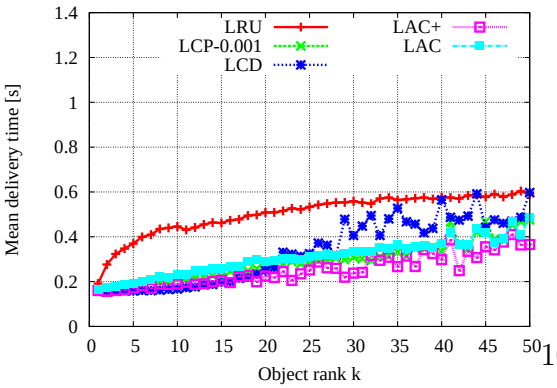
(d) FOCAL egress traffic: cache cap. = 10.



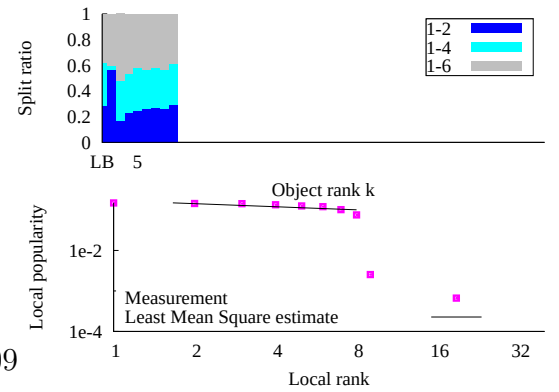
(e) LB-Perf: cache capacity = 25.



(f) FOCAL egress traffic: cache cap. = 25.



(g) LB-Perf: cache capacity = 40.



(h) FOCAL egress traffic: cache cap. = 40.

Figure 3.15 – Linear topology with forwarding branches: (a),(c),(e),(g) Mean delivery time w.r.t content rank. (b),(d),(f),(h) Node 1 egress traffic under FOCAL.

Chapter 4

On the Fairness of ICN

Can ICN be fair ?

Summary. *Cache networks are cornerstones to today's Internet, helping it to scale by an extensive resort to Content Delivery Networks (CDN). Inheriting from CDN's successful insights, ubiquitous caching through Information-Centric Networks (ICN) is increasingly regarded as a premier future Internet architecture contestant. However, the use of in-network caches seems to cause a distortion in the fairness of resource sharing among contents. Indeed, in legacy communication networks, link buffers were the principal resources to be shared. Under max-min flow-wise fair bandwidth sharing [Massoulié and Roberts, 1999], content throughput was not tied to content popularity. Including caches in this ecosystem raises new issues since common cache management policies such as (p-)LRU or even more, LFU, may look detrimental to low popularity objects, even though they significantly decrease the overall link load [Carofiglio et al., 2013a]. In this chapter, we demonstrate that, as surprising as it may seem, globally achieving* LFU is a first stage to content-wise fairness*. Indeed, any investigated content-wise α-fair allocation permanently stores the most popular contents in network caches by ensuring them a cache hit ratio of 1. As ICN caching traditionally pursues LFU objectives, content-wise fairness specifics remain only a matter of fair bandwidth sharing, keeping the cache management intact.*

Keywords: *Information-Centric Networks; Caching; Queuing; Fairness.*

Contents

4.1 Introduction

Today's Internet owes its scalability to caching. Indeed, most of Internet contents cross Content Delivery Networks and significant research is pushing for a better recourse, Information-Centric Networks. In ICN, and more specifically Named-Data Networking (NDN) or Content-Centric Networking (CCN), two leading ICN architectures, content objects are identified by their unique name. At every node/router, content Data packets are requested via matching Interest packets through egress faces. Interests and their satisfying Data counterparts follow rigorously the same path. This feature would be achievable without the Pending Interest Table (PIT) structure that keeps track of every requesting face and the requested content. Data packet names permits to store them, on every traversed node, in a finite memory referred to as Content Store (CS) or cache, managed by an eviction policy.

Caches and their eviction or management policies are the disruption that drives this chapter. Traditionally, networks are modeled as interconnected queues, fair schedulers operate on. The penetration of caching into the network layer clearly favors a few content objects, the most popular ones in case of the (evict the) Least Frequently Used management policy (LFU) and its approximations such as (p-)LRU or Leave-Copy-Down + LCD [Martina et al., 2013]. Filling caches steadily with the most popular items, meaning keeping their hit ratio to their maximum *i.e.*, one and letting others hit ratio at zero, entails the sacrifice of less popular objects [Carofiglio et al., 2013a]. This is at least a view comforted by state-of-art contributions on content-wise cache fairness [Tortelli et al., 2011] [Dehghan et al., 2016]. These works observe the hit ratio on a single cache or a network of caches and prescribe to adapt the cache management policy to fairness motives. For example, in [Dehghan et al., 2016], content-wise max-min fairness is only achievable if the hit ratios are forced to be equal for all content objects. In the same vein, proportional fairness impose to make content hit ratio proportional to popularity. A consequence of this is that ICN cannot be fair to contents without revising its caching algorithms. For them, LFU is definitely unfair. Remember flow-wise fairness as allocating resources such that every flow/route gets its fair share. By content-wise fairness, we denote allocating resources in such a way every content gets its fair share. This is the type of fairness this chapter analyses.

This chapter analyzes the fairness of content delivery throughput in accounting for

both cache hit ratio and link service rates, and comes up with different and optimistic conclusion. *ICN's traditional caching leads to fairness as-is*. The better the convergence to LFU, the better the feasible fairness. It remains that content-wise fairness just has to be implemented at the packet scheduling stage in ICN, similarly to flow-wise fairness in other networks [Kelly et al., 1998]. Our seemingly counter-intuitive results owe to the link service to others that balances the remnant cache presence of a few. Moreover, that persistence frees a maximal upstream link capacity to convey less popular objects. Another striking insight we get, is that a throughput-optimal content delivery network ends up being autonomous caches that never forward their miss traffic. Such a network would not be committed to locally satisfy requests.

The main contributions of this chapter are: (i) it unifies caches and network queues into a single content service rate model; (ii) it tackles for the first time content throughput fairness in ICN in formulating that as a tractable nonlinear optimization problem; (iii) it provides closed-form expressions of α-fair hit ratios and link service rates; (iv) it indicates that today's LFU-approximating caches policies do not need to be replaced for ICN to become fair. We articulate these contributions throughout the chapter as follows: Sec.4.2 recapitulates previous contributions on fairness in the context of cache networks. In Sec.4.3, we model the per-content throughput in unifying cache and network link contributions. Then we formalize α-fair allocations and key properties such as their Pareto-efficiency and LFU being the α-fair cache management policy, an important result. To ground the theory, a few trivial examples are analyzed in Sec.4.4. There are followed in Sec.4.5 with numerical evaluations that confirmed by means of a nonlinear problem solver our analytic insights.

4.2 Related work

Very few papers tackle the issue of fairness in networks of caches. In a paper [Tortelli et al., 2011] dedicated to the subject some time ago, authors analyze the fairness in Content-Centric Networks from the point of view of object dissemination across the network. Content-wise fairness was expressed as the total space content occupy with respect to their popularity. This study concluded that medium-popularity content were favored as they spread linearly with their popularity whereas the most popular item spread sublinearly. This approach is definitely useful to map the asymptotic replica spatial distri-

bution. However, it does not capture the fairness that really matters from a user point of view, the throughput fairness. [Shah and de Veciana, 2014] and [Shah and de Veciana, 2015] tackle the impact of fairness on delivery time in large scale CDN but ignores the cache specifics. This work models cache networks as classical networks of file-serving queues. Files are assumed pre-fetched and their long-term popularity is not taken into account.

Quite recently, [Dehghan et al., 2016] reverse-engineered popular LRU and LFU policy and found the utility function both policies optimize. These utility functions achieve various classes of single-cache storage α-fairness. They provided algorithms for adapting TTL-based caches to given α-fair objectives. Rapidly, [Neglia et al., 2016] applied this work's reverse engineering approach to a special case of a novel class of latency-aware (LAC) policies previously introduced by [Carofiglio et al., 2015c]. In [Neglia et al., 2016], authors show that LAC converges to the solution of a fractional knapsack problem (LFU) when their latency exponent tends to infinity.

The existing literature on the subject, because of its focus on hit ratio, concluded that caching policies had to adapt to the content-wise fair objective. Our contribution is disruptive because it joins cache and link queue occupation in order to analyze the throughput fairness. We show that cache networks, and ICN in particular, can be α-fair, for any $\alpha \geq 0$, as soon as they couple the classical highest popularity content persistence *i.e.*, the global LFU cache management policy, with a proper α-fair packet scheduler.

4.3 Cache Network Model

First we present a mathematical model that captures the dynamics of the entire network. The model views it as a network of queues where caches contribute to increase the service rate. We aim at maximizing a utility function of the admissible exogenous traffic rate. Refer to Table 4.1 for the notation and to 4.1 for the model used hereinafter.

4.3.1 Model assumptions

- The stochastic processes $\{\lambda_{k,n,b}(t)\}_{0\leq t\leq T}$ as exogenous rates, $\{\mu_{k,n,b}(t)\}_{0\leq t\leq T}$ as service rates and $\{h_{n,k}(t)\}_{0\leq t\leq T}$ as hit ratios are independent.

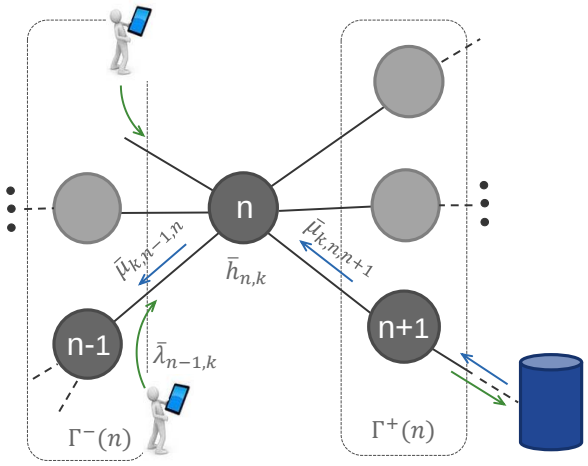


Figure 4.1 – Network conveying content k through cache n.

- The network routes a single prefix.
- Same object sizes.
- Cache size is never zero.
- Content servers are not clients.
- The exogenous traffic on a given node is the one that is the one generated by a local application that is not satisfied by the local cache.
- We assume hop-by-hop congestion control *i.e.*, interests are sent in average at a rate equivalent to the service rate of the link.

Let define a Pending Interest Queue (PIQ) size as the number of pending interests per content and per face. An interest queued in a PIQ is served when the matching data packet comes back.

Let $Q_{k,n,b}(t)$ be the size of the Pending Interest Queue for content k at time t for link (n, b). $h_{n,k}(t) \equiv \mathbb{1}_{\{k \text{ in cache } n \text{ at } t\}}$ indicates whether content k was found in cache n at time t. The time evolution upper bound of the PIQ size of content k for egress nodes $b \in \Gamma^{+}(n)$

$n \in \mathcal{N}$	ICN node identifier. $\mathcal{N} \subseteq \mathbb{N}$.
$t \in \mathbb{R}_+$	Instant a content retrieval occurs.
$k \in \mathcal{K}$	Content popularity rank. The one ranking first is the most popular, while rank $\lvert\mathcal{K}\rvert$ indicates the least popular object.
$\Gamma^-(n)$	Set of node n's ingress nodes.
$\Gamma^+(n)$	Set of node n's egress nodes.
$\bar{\mu}_{k,n,b}$	Long-term average of the service rate for content k on link (b,n)
$\bar{\lambda}_n$	Long-term average of exogenous interest rate at node n.
$\lambda_{n,k}$	Long-term interest rate for content k at node n.
q_k	Content k popularity. It is the probability that a requested content is content k. It is strictly ordered: $q_{k+1} < q_k$.
$\mathbb{1}_{\{.\}}$	Indicator function.
$\bar{\Lambda}_{n,k}$	Upper bound for the long-term average of exogenous interest rate for content k at node n .
$\varsigma(k)$	Set of content k servers.
$h_{n,k}$	hit ratio of content k at node n
$C_{b,n}$	Link (b,n) capacity in chunks/s.
x_n	Cache n capacity in objects.

Table 4.1 – Notation.

at node n follows:

$$\begin{aligned} Q_{k,n,b}(0) = 0, \forall b \in \Gamma^+(n) \text{ and} \\ \sum_{b\in\Gamma^+(n)} \frac{d}{dt} Q_{k,n,b}(t) \leq \lambda_{n,k}(t) \\ + (1 - h_{n,k}(t)) \sum_{a\in\Gamma^-(n)} \mathbb{1}_{\{Q_{k,a,n}(t)>0\}} \mu_{k,a,n}(t) \\ - \sum_{b\in\Gamma^+(n)} \mathbb{1}_{\{Q_{k,n,b}(t)>0\}} \mu_{k,n,b}(t). \end{aligned} \tag{4.1}$$

The service rate $\mu_{k,a,b}(t)$ of the PIQ is the data rate for content k on link (b,a) at time t. $\lambda_{n,k}(t) \equiv q_k \lambda_n(t)$ is the exogenous interest rate for content k at node n at time t. After some algebra (Proposition 3.3), the maximal admissible rate $\bar{\Lambda}_{n,k}$ is given by:

$$\bar{\Lambda}_{n,k} \equiv \sum_{b\in\Gamma^+(n)} \bar{\mu}_{k,n,b} - (1 - \bar{h}_{n,k}) \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n}, \forall n, k \tag{4.2}$$

constrained by the following bounds:

$$\sum_{k\in\mathcal{K}} \bar{\mu}_{k,a,n} \leq C_{n,a}, \quad \forall n, a \in \Gamma^-(n) \tag{4.3}$$

$$\sum_{k\in\mathcal{K}} \bar{h}_{n,k} = x_n, \quad \forall n \tag{4.4}$$

$$0 \leq \bar{h}_{n,k} \leq 1, \quad \forall n, k \tag{4.5}$$

$$\bar{\mu}_{k,a,n} \geq 0, \quad \forall n, k, a \in \Gamma^-(n) \tag{4.6}$$

$$\bar{\lambda}_{n,k} \leq \bar{\Lambda}_{n,k}, \quad \forall n, k. \tag{4.7}$$

We ignore throughout the chapter constraint (4.7) that imposes a lower bound to the content service rate. It means that the network will not guarantee that some content requested on a given node will be satisfied. This will entirely depend on the optimality of serving it.

4.3.2 Cache network capacity

The network provides a content delivery service through the coupling of disseminated caches and the links interconnecting them. The following expression unifies in a single expression the maximum service rate the network can deliver given cache hit ratios and link capacities.

It arises by first summing all maximum admissible rates at node n:

$$\begin{aligned}\sum_k \bar{\Lambda}_{n,k} &= \sum_k \Big[\sum_{b\in\Gamma^+(n)} \bar{\mu}_{k,n,b} - (1-\bar{h}_{n,k}) \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n} \Big] \\ &= \sum_{k,b\in\Gamma^+(n)} \bar{\mu}_{k,n,b} - \sum_{k,a\in\Gamma^-(n)} (1-\bar{h}_{n,k}) \bar{\mu}_{k,a,n}.\end{aligned}$$

Define $\bar{\boldsymbol{\mu}}_n^-$ as the ingress rate matrix $(\bar{\mu}_{k,a,n})_{k,a}$ and $\bar{\boldsymbol{h}}_n$ as the column vector $(\bar{h}_{n,k})_k$. It follows that

$$\begin{aligned}\sum_{k,a\in\Gamma^-(n)} \bar{\mu}_{k,a,n} \bar{h}_{n,k} = \|^t \bar{\boldsymbol{\mu}}_n^- \bar{\boldsymbol{h}}_n\|_1 &\leq \|^t \bar{\boldsymbol{\mu}}_n^-\|_1 \|\bar{\boldsymbol{h}}_n\|_1 \\ &\leq \sup_k \Big\{ \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n} \Big\} x_n,\end{aligned}$$

where ${}^t\bar{\boldsymbol{\mu}}_n^-$ is the transpose of the ingress rate matrix and $\|{}^t\bar{\boldsymbol{\mu}}_n^-\|_1$ is the operator norm [Boyd and Vandenberghe, 2004] associated to the Banach space ℓ^1 *i.e.*, $(\mathbb{R}^{|\mathcal{K}|}, \| \, \|_1)$, applied to the ingress rate matrix.

Then we sum all maximum admissible rates. Rates that are both egress to a node and ingress to another one cancel. We obtain:

$$\begin{aligned}\sum_{n,k} \bar{\Lambda}_{n,k} &\leq \sum_{\substack{k,b\in\varsigma(k)\\ n\in\Gamma^-(b)}} \bar{\mu}_{k,n,b} + \sum_n \sup_k \Big\{ \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n} \Big\} x_n \\ &\leq \sum_{\substack{b\in\cup_k\varsigma(k)\\ n\in\Gamma^-(b)}} C_{b,n} + \sum_n \sup_k \Big\{ \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n} \Big\} x_n. \end{aligned} \tag{4.8}$$

4.3.3 Problem formulation

We now plug the admissible rate into a fair utility function $U(\cdot)$. Define the network wide allocated rate for content k as

$$\phi_k \equiv \sum_{n\in\mathcal{N}} \bar{\Lambda}_{n,k} = \sum_{\substack{b\in\cup_k\varsigma(k)\\ n\in\Gamma^-(b)}} \bar{\mu}_{k,b,n} + \sum_{\substack{n\in\mathcal{N}\\ a\in\Gamma^-(n)}} \bar{h}_{n,k}\bar{\mu}_{k,a,n}. \tag{4.9}$$

The problem's objective is to find the optimal link service rates and hit ratios that

$$\underset{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}}{\text{maximize}} \sum_{k\in\mathcal{K}} q_k U(\phi_k/q_k), \tag{4.10}$$

given the α-fair utility function $U(\cdot)$. Weighted α-fairness was first introduced in [Mo and Walrand, 2000], and later adapted for cache allocation by [Dehghan et al., 2016]. However, as expressed in Eq.4.10, we advocate for a formulation of *weighted α-fairness* that operates on rates per weight unit. The reason for this is its convergence to *weighted max-min* fairness as $\alpha \to \infty$, whereas the original Mo and Walrand's weighted formulation decays into max-min fairness [Mo and Walrand, 2000]. Interestingly, as shown later in the chapter, our formulation gives solutions that are independent of α, shaping as such, just fair allocations.

$$\text{Since } q_k U(\phi_k/q_k) \equiv q_k \frac{(\phi_k/q_k)^{1-\alpha}}{1-\alpha} = q_k^\alpha \frac{\phi_k^{1-\alpha}}{1-\alpha}, \alpha \neq 1,$$

the objective simplifies to

$$\underset{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}}{\text{Maximize}} \quad \begin{cases} \sum_{k\in\mathcal{K}} q_k^{\alpha} U(\phi_k), & \text{if } \alpha \neq 1 \\ \sum_{k\in\mathcal{K}} q_k \log(\phi_k/q_k), & \text{otherwise.} \end{cases} \tag{4.11}$$

Special cases this weighted α-fairness framework encompasses are:

- for $\alpha = 1$, the objective is then said to be weighted-proportional fair [Kelly et al., 1998].
- for an infinite value of α, the objective is weighted max-min fair *i.e.*, $\max\min(\phi_k/q_k)$ [Radunović and Boudec, 2007].

In the rest of the document, the attribute *weighted* will be implied when ommitted. Define the vectors of decision variables $\bar{\boldsymbol{\mu}} \equiv (\bar{\mu}_{k,b,n})$ and $\bar{\boldsymbol{h}} \equiv (\bar{h}_{n,k})$. Also, define the vector of multipliers $\boldsymbol{\nu} \equiv \left(\nu_{\cdot}^{(i)} \geq 0\right)$ where (i) identifies the constraint. The Lagrangian of the problem is

$$\begin{aligned} \mathcal{L}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{h}}, \boldsymbol{\nu}) = & \sum_k q_k^{\alpha} U(\phi_k) \\ & - \sum_{n,a\in\Gamma^-(n)} \nu_{n,a}^{(1)} \Big[\sum_k \bar{\mu}_{k,a,n} - C_{n,a} \Big] \\ & - \sum_{n,k} \nu_n^{(2)} \Big[\sum_k \bar{h}_{n,k} - x_n \Big] \\ & - \sum_{n,k} \nu_{n,k}^{(3)} \bar{h}_{n,k} (\bar{h}_{n,k} - 1) \\ & + \sum_{n,a,k} \nu_{k,a,n}^{(4)} \bar{\mu}_{k,a,n}. \end{aligned} \tag{4.12}$$

Although $U(\cdot)$ is non-decreasing and concave, as ϕ_k is non-concave, the Karush-Kuhn-Tucker (KKT) conditions are simply necessary for optimality.

The first-order KKT conditions command that

$$\nabla_{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}} \mathcal{L}(\bar{\boldsymbol{\mu}}^*, \bar{\boldsymbol{h}}^*, \boldsymbol{\nu}^*) = \vec{0}, \tag{4.13}$$

where $\nabla_{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}}\mathcal{L}$ is the gradient of function $\mathcal{L}$ with respect to vectors $\bar{\boldsymbol{\mu}}$ and $\bar{\boldsymbol{h}}$. $\bar{\boldsymbol{\mu}}^* \equiv (\bar{\mu}^*_{k,b,n})$, $\bar{\boldsymbol{h}}^* \equiv (\bar{h}^*_{n,k})$, $\boldsymbol{\nu}^* \equiv (\nu^{(i)*}_{\cdot})$ are the optimal counterparts of the aforementioned vectors.

4.3.4 Solution

General α-fair allocation

For any $\alpha \geq 0$, the Lagrangian expands as follows:

$$\begin{aligned}
&\mathcal{L}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{h}}, \boldsymbol{\nu}) = \\
&\frac{1}{1-\alpha}\sum_{k} q_k^{\alpha}\Big[\sum_{\substack{b\in\cup_k\varsigma(k)\\ n\in\Gamma^-(b)}} \bar{\mu}_{k,b,n} + \sum_{\substack{n\in\mathcal{N}\\ a\in\Gamma^-(n)}} \bar{h}_{n,k}\bar{\mu}_{k,a,n}\Big]^{1-\alpha} \\
&\quad - \sum_{n,a\in\Gamma^-(n)} \nu^{(1)}_{n,a}\Big[\sum_{k}\bar{\mu}_{k,a,n} - C_{n,a}\Big] \\
&\quad - \sum_{n,k} \nu^{(2)}_{n}\Big[\sum_{k}\bar{h}_{n,k} - x_n\Big] \\
&\quad - \sum_{n,k} \nu^{(3)}_{n,k}\bar{h}_{n,k}(\bar{h}_{n,k} - 1) \\
&\quad + \sum_{n,a,k} \nu^{(4)}_{k,a,n}\bar{\mu}_{k,a,n}.
\end{aligned}$$

The first property of content-wise α-fair allocations in cache networks is their Pareto efficiency. An allocation is said to be Pareto efficient if any attempt to increase one content's share decreases the share of another content. In our optimization problem, it translates into link capacity being fully allocated.

Property 4.1 (Pareto efficiency for any $\alpha \geq 0$)**.** The α-fair bandwidth allocation is Pareto efficient as the optimal resource allocation uses the whole link capacities to serve content items i.e.,

$$\sum_{k\in\mathcal{K}} \bar{\mu}^*_{k,a,n} = C_{n,a}, \quad \forall n \in \mathcal{N}, \forall a \in \Gamma^-(n). \tag{4.14}$$

Proof. The partial derivatives with respect to the ingress capacities $\bar{\mu}_{k,a,n}$ give:

$$\bar{h}^*_{n,k} = \frac{\nu^{(1)*}_{n,a} - \nu^{(4)*}_{k,n,a}}{(q_k/\phi^*_k)^\alpha}, \forall k, n, a \in \Gamma^-(n).$$

As the sum of local hit ratio equals the size of the cache,

$$\Big[x_n + \sum_k \nu^{(4)*}_{k,a,n} \Big(\frac{\phi^*_k}{q_k}\Big)^\alpha\Big] \Big[\sum_k \Big(\frac{\phi^*_k}{q_k}\Big)^\alpha\Big]^{-1} = \nu^{(1)*}_{n,a} > 0,$$
$$\forall n, a \in \Gamma^-(n).$$

As $\nu^{(1)*}_{n,a}$ is strictly positive since the cache size is. By the complementary slackness conditions of the convex optimization framework, the related constraint must be saturated. That translates into Eq.4.14.

Moreover, first derivatives with respect to the server's ingress capacities $\mu_{k,n,b}$ give:

$$\Big(\frac{q_k}{\phi^*_k}\Big)^\alpha + \nu^{(4)*}_{k,n,b} = \nu^{(1)*}_{n,b} > 0, \forall k, b \in \varsigma(k), n \in \Gamma^-(b).$$

The multipliers being strictly positive, the corresponding constraints must be saturated. It makes Eq.4.14 hold and Pareto efficiency follow. □

Then comes our main result. It established that ICN, in adopting the Least Frequency Used as the caching policy maximizing content hit ratio, has de facto adopted the optimal caching for content throughput fairness.

Proposition 4.1 (LFU leads to α-fairness)**.** LFU is a cache management policy for a network seeking content-wise throughput α-fairness, for any $\alpha \geq 0$.

Proof. Let $f(\cdot)$ be the α-fair objective function. The increase of f with regards to an increase of content k's ingress rate is

$$df_{\mu_{k,a,n}} = \frac{\partial f(\bar{\mu}^*_{k,a,n})}{\partial \bar{\mu}^*_{k,a,n}} d\bar{\mu} = \bar{h}^*_{n,k} \Big(\frac{q_k}{\phi^*_k}\Big)^\alpha d\bar{\mu}.$$

Let $\epsilon(\alpha)$ be the increase of the α-fair objective function induced by an increase

of content 1's rate and the equivalent decrease of content k's rate, $k \uparrow \infty$.

$$\begin{aligned}\epsilon(\alpha) &\equiv df_{\mu_{1,a,n}} - \lim_{k\to\infty} df_{\mu_{k,a,n}} \\ &= \Big[\bar{h}^*_{n,1} - \lim_{k\to\infty} \bar{h}^*_{n,k}\Big(\frac{q_k/\phi^*_k}{q_1/\phi^*_1}\Big)^{\alpha}\Big] d\bar{\mu}. \end{aligned} \tag{4.15}$$

We aim at proving that $\nu^{(3)*}_{n,k} > 0, \forall n, k$. By KKT complementary slackness conditions, it would imply that $\bar{h}^*_{n,k} \in \{0, 1\}, \forall n, k$. The partial derivatives w.r.t. cache hit ratios give

$$\sum_{a\in\Gamma^-(n)} \bar{\mu}^*_{k,a,n} = \Big(\frac{\phi^*_k}{q_k}\Big)^{\alpha}\Big[\nu^{(2)*}_n + \nu^{(3)*}_{n,k}(2\bar{h}^*_{n,k} - 1)\Big], \forall n, k.$$

As stated in Eq.4.14, per-content services rates cumulatively equal the downlink capacity. This helps getting rid of the link service rate in the above expression and obtain that:

$$\begin{aligned}\sum_{k,a\in\Gamma^-(n)} \bar{\mu}^*_{k,a,n} &= \sum_k \Big(\frac{\phi^*_k}{q_k}\Big)^{\alpha}\Big[\nu^{(2)*}_n + \nu^{(3)*}_{n,k}(2\bar{h}^*_{n,k} - 1)\Big] \\ &= \sum_{a\in\Gamma^-(n)} C_{n,a}, \forall n.\end{aligned}$$

Then we substantiate the pivotal multiplier

$$\begin{aligned}&\nu^{(2)*}_n = \Big[\sum_k \Big(\frac{\phi^*_k}{q_k}\Big)^{\alpha}\Big]^{-1} \mathcal{C}_n, \forall n, \\ &\text{where } \mathcal{C}_n = \sum_{a\in\Gamma^-(n)} C_{n,a} - \sum_k \Big(\frac{\phi^*_k}{q_k}\Big)^{\alpha} \nu^{(3)*}_{n,k}(2\bar{h}^*_{n,k} - 1).\end{aligned}$$

By contradiction, suppose $\nu_{n,k}^{(3)*} = 0$. It entails

$$\sum_{a \in \Gamma^-(n)} \bar{\mu}_{k,a,n}^* = \frac{(\phi_k^*/q_k)^\alpha}{\sum_i (\phi_i^*/q_i)^\alpha} \mathcal{C}_n, \forall \alpha \geq 0$$
$$= \frac{\mathcal{C}_n}{|\mathcal{K}|}, \text{for } \alpha = 0 \text{ and } \alpha \to \infty.$$

However, by Eq.4.15, $\forall \alpha \geq 0$, $\bar{h}_{n,1}^* = 1$ and $\bar{h}_{n,k}^* = 0, k \uparrow \infty$ yield $\epsilon(\alpha) > 0$. Consequently, as $\nu_{n,k}^{(3)*} = 0$ does not lead to a maximum, $\nu_{n,k}^{(3)*} > 0$ and $\bar{h}_{n,k}^* \in \{0, 1\}$. □

This result is important as it shows that the LFU algorithm and its heuristics (LRU, p-LRU, LRU-k, LRU+LCD) can lead to α-fair networks, $\forall \alpha \geq 0$. Packet schedulers would be in charge of the other part of the optimal solution, a bandwidth sharing fair to contents. We refer to the latter as content-wise α-fair bandwidth sharing. It is mathematically tractable thanks to the concavity of the problem, given binary hit ratios, as concavity is a sufficient condition for the existence of a global optimum. Furthermore, content-wise α-fair bandwidth sharing is practically achievable within the ICN paradigm as packets are uniquely named after the content they carry.

We may also emphasize the novelty of this result, since previous works [Dehghan et al., 2016] reached very different conclusions. This is because they only looked at isolated caches, found that fairness required fractional hit ratios for each value of the fairness parameter α greater than zero, and suitably designed algorithms for TTL-based caches. Their caching algorithms consist in adjusting every content Time-To-Live (TTL) via gradient descent.

To wrap up, the following algorithm (Alg.3, Alg.4) is an example of distributed content-wise weighted max-min fairness implementation. It relies on a Deficit Round-Robin scheduler [Shreedhar and Varghese, 1995] to achieve content-wise max-min fair bandwidth allocation, given the LFU caching substrate.

Algorithm 3: Content-wise α-fair allocation in ICN

Input: Data packet, α
Cache.Insert(packet, Policy::LFU);
FairQueuing.Shape(packet, α);

Algorithm 4: Content-wise weighted max-min fair bandwidth sharing

```
function FairRate.Shape(Data packet, α)
    FairQueuing.Queue[packet.ContentName()].Push(packet);
    if α == ∞ then
        FairQueuing.SendData(Policy::DEFICIT_ROUND_ROBIN);
    end
end
```

4.4 Toy examples

We analyze two trivial cases to foster some intuition on the preceding results, and preclude limit case quandaries. We tackle the case of a connected client/server tandem then we illustrate the fairness problem on a client/cache/server bus.

4.4.1 Client/Server tandem

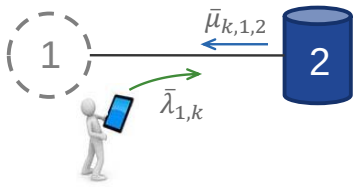


Figure 4.2 – Client/server topology.

A communication link conveys some exogenous traffic from a client node numbered 1 to a content server numbered 2. There is no cache in between. The α-fair objective writes:

$$\text{Maximize} \quad \sum_k q_k^\alpha \frac{(\bar{\mu}_{k,1,2})^{1-\alpha}}{1-\alpha}.$$

The optimal allocation for every content k, whatever $\alpha \geq 0$, is

$$\mu_{k,1,2} = q_k C_{2,1}.$$

4.4.2 Client/Cache/Server bus

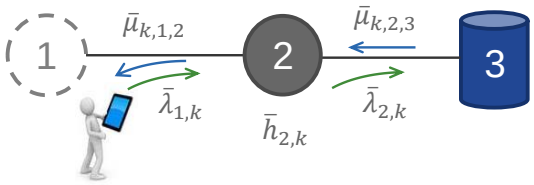


Figure 4.3 – Client/Cache/Server bus topology.

In this toy scenario, exogenous traffic at a client node 1 is conveyed towards a content server 3 through an intermediate cache 2.

Proportional fairness

The related objective writes

Maximize

$$\sum_k q_k \log \frac{\bar{h}_{2,k}\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3}}{q_k}.$$

Assume that $\bar{\mu}_{k,2,3} = 0$, *i.e.*, the server's egress link capacity is zero. Then, the following two concave terms have to be independently maximized:

$$\sum_k q_k \log \frac{\bar{h}_{2,k}}{q_k} + \sum_k q_k \log \frac{\bar{\mu}_{k,1,2}}{q_k}.$$

The optimal solutions are $\bar{h}_{2,k} = q_k x_2$ and $\bar{\mu}_{1,k} = q_k C_{2,1}$.

In case $\bar{\mu}_{k,2,3} > 0$ we are in a situation where the server can deliver data through its ingress link. The following demonstration shows LFU is the cache heuristics that always finds the unique optimum if any. By the way, this claim is later confirmed numerically in Sec.4.5.1.

• Assuming that the optimal $\bar{h}_{2,k} \in \{0, 1\}$, we can deduce the optimal link services rates.

To that aim, first define the two sets $\mathcal{K}_i \equiv \{k \in \mathcal{K} : \bar{h}_{2,k} = i\}, \forall i \in \{0,1\}$. $\mathcal{K}_0$ is the set of objects that are not stored in the cache, $\mathcal{K}_1$ is the set of object that are permanently cached. As such, $|\mathcal{K}_1| = x_2$. So, the concave objective yields

$$\text{Maximize} \quad \sum_{k\in\mathcal{K}_0} q_k \log \frac{\bar{\mu}_{k,2,3}}{q_k} + \sum_{k\in\mathcal{K}_1} q_k \log \frac{\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3}}{q_k}. \tag{4.16}$$

Define β as an optimal multiplier tied to the constraint upon the capacity of the link to the server. The KKT conditions holding,

$$\bar{\mu}_{k,2,3} = \frac{q_k}{\beta}, \forall k \in \mathcal{K}_0, \text{ and}$$
$$\bar{\mu}_{k,2,3} = \frac{q_k}{\beta} - \bar{\mu}_{k,1,2}, \forall k \in \mathcal{K}_1.$$

As

$$\sum_k \bar{\mu}_{k,2,3} = C_{3,2} = \frac{1}{\beta}\Big[\sum_{k\in\mathcal{K}_0} q_k + \sum_{k\in\mathcal{K}_1} q_k\Big] - \sum_{k\in\mathcal{K}_1} \bar{\mu}_{k,1,2},$$

we obtain

$$\frac{1}{\beta} = C_{3,2} + \sum_{k\in\mathcal{K}_1} \bar{\mu}_{k,1,2} \leq C_{3,2} + C_{2,1}.$$

Hence, the optimal rates satisfy

$$\bar{\mu}_{k,2,3} = q_k\Big[C_{3,2} + \sum_{k\in\mathcal{K}_1} \bar{\mu}_{k,1,2}\Big], \forall k \in \mathcal{K}_0, \tag{4.17}$$
$$\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3} = q_k\Big[C_{3,2} + \sum_{k\in\mathcal{K}_1} \bar{\mu}_{k,1,2}\Big], \forall k \in \mathcal{K}_1. \tag{4.18}$$

• We insert that solution into Eq.4.16 to explicit the sets $\mathcal{K}_i$. At the optimum, the

objective function reaches its supremum

$$S \equiv \sup_{\mathcal{K}_1} \Big\{ \log(C_{3,2} + \sum_{k \in \mathcal{K}_1} \bar{\mu}_{k,1,2}) : \\ |\mathcal{K}_1| = x_2 \text{ and } \sum_{k \in \mathcal{K}_1} \bar{\mu}_{k,1,2} \leq C_{2,1} \Big\}. \tag{4.19}$$

Define the network capacity $\kappa \equiv C_{2,1} + C_{3,2}$. The upper bound of S, denoted as $S^{\max}$ equals $\log \kappa$. As illustrated in Fig.4.4, a greedy algorithm finds the supremum $S \leq S_{\max}$ in piggybacking the x_2-most popular objects in $\mathcal{K}_1$. LFU is this greedy heuristics. As such, it greedily chooses the most popular contents as those worth being stored into the cache. Optimally, that also implies sharing the entire cache's ingress link capacity among these contents. The following observations can be made:

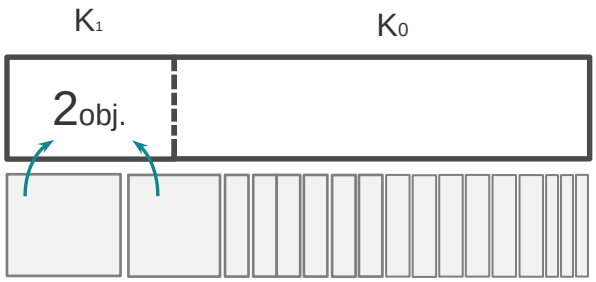


Figure 4.4 – Example of greedy resolution of Eq.4.19 with $x_2 = 2$. Piggybacking into $\mathcal{K}_1$ the two most significant contents, as LFU does, is optimal.

Observation 4.1 (Maximal solution existence)**.** The greedy algorithm finds a cache allocation and fair service rates such that the objective function equals its upper bound $S^{\max}$ *iif* $\sum_{k=1}^{x_2} q_k \geq C_{2,1}/\kappa$.

This happens when the cumulative fair service rate for the objects in cache exceeds the capacity of the link to that cache. Even if that solution might not be unique, (for example, in case of $C_{2,1}/\kappa$ being too small), whenever $S^{\max}$ is reachable, the greedy algorithm finds a solution achieving it.

Observation 4.2 (Maximal solution uniqueness)**.** There exists a unique combination of cache allocation and fair service rates such that the objective function equals its upper bound $S^{\max}$ *iif* $\sum_{k=1}^{x_2-1} q_k + q_{x_2+1} < C_{2,1}/\kappa$.

Indeed, if one can not replace the least popular object stored in the cache by some other and get a fair service rate exceeding the capacity of the cache's ingress link, then the LFU-provided cache configuration is the unique optimum.

Conversely, if the cumulative fair service rate of the x_2-most popular objects remains lower than the capacity of the cache's ingress link, $\sum_{k\in\mathcal{K}_1}\mu_{k,1,2} < C_{2,1}$. This happens when the cache's ingress link $(2,1)$ is over-provisioned. Consequently, the link service rates of never-cached objects $k \in \mathcal{K}_0$, $\mu_{k,1,2} > 0$, and the fair objective function can not reach its upper bound $S^{\max}$ as the network carries some miss traffic.

• To conclude, remember $\bar{h}_{2,k}$ was assumed to equal either 0 or 1. We show that $S^{\max}$ is also the upper bound of the general objective function we denote f, for any $\bar{h}_{2,k} \in [0,1]$. Indeed, as the objective function increases with any of the decision variables, any attempt to increase $S^{\max}$ to $S^{\max}+\epsilon$, $\forall\epsilon > 0$, necessarily increases some zero hit ratio by $\delta\bar{h} > 0$ and decreases a hit ratio of one by the same amount.

$$\epsilon = \Big[\frac{\partial f(\bar{h})}{\partial \bar{h}}\Big|_{\bar{h}_{2,k}=0} - \frac{\partial f(\bar{h})}{\partial \bar{h}}\Big|_{\bar{h}_{2,k}=1}\Big]\delta\bar{h} \qquad = -\frac{\bar{\mu}_{k,1,2}}{C_{2,1}+C_{3,2}}\delta\bar{h}.$$

As $\epsilon \leq 0$, the solution provided through LFU caching is the optimum for any $\bar{h}_{2,k} \in [0,1]$.

General α-fairness

Here the objective consists in the following:

Maximize

$$\sum_k \frac{q_k^\alpha}{1-\alpha}(\bar{h}_{2,k}\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3})^{1-\alpha}.$$

As demonstrated previously, the optimal $\bar{h}_{2,k}$ belong to $\{0,1\}$. It allows to reuse the aforementioned definition of the sets $\mathcal{K}_i$. We can calculate the optimal link services rates, owing to the concave objective function

$$\sum_{k\in\mathcal{K}_0} \frac{q_k^\alpha}{1-\alpha}(\bar{\mu}_{k,2,3})^{1-\alpha} + \sum_{k\in\mathcal{K}_1} \frac{q_k^\alpha}{1-\alpha}(\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3})^{1-\alpha}. \tag{4.20}$$

The KKT conditions yield

$$\bar{\mu}_{k,2,3} = \frac{q_k}{\beta^{1/\alpha}}, \forall k \in \mathcal{K}_0, \text{ and}$$
$$\bar{\mu}_{k,2,3} = \frac{q_k}{\beta^{1/\alpha}} - \bar{\mu}_{k,1,2}, \forall k \in \mathcal{K}_1.$$

It follows that

$$\sum_k \bar{\mu}_{k,2,3} = C_{3,2} = \frac{1}{\beta^{1/\alpha}} \Big[\sum_{k \in \mathcal{K}_0} q_k + \sum_{k \in \mathcal{K}_1} q_k \Big] - \sum_{k \in \mathcal{K}_1} \bar{\mu}_{k,1,2}$$

gives the following optimal rates:

$$\bar{\mu}_{k,2,3} = q_k \Big[C_{2,1} + \sum_{k \in \mathcal{K}_1} \bar{\mu}_{k,1,2} \Big], \forall k \in \mathcal{K}_0$$
$$\bar{\mu}_{k,1,2} + \bar{\mu}_{k,2,3} = q_k \Big[C_{2,1} + \sum_{k \in \mathcal{K}_1} \bar{\mu}_{k,1,2} \Big], \forall k \in \mathcal{K}_1,$$

making the optimal allocation identical to the proportional fairness case we presented earlier.

4.5 Evaluation

We numerically solved problem (4.10) using SCIP 3.2.1 [Achterberg, 2009] a Mixed Integer Non-Linear Program (MINLP) optimization suite. It actually performs *branch-cut-and-price* on mixed integer problems and invokes the Interior Point Optimizer IPOPT 3.12.5 [Wächter and Biegler, 2006] to solve relaxed nonlinear instances. IPOPT itself relies on PARDISO 5.0.0 [Kuzmin et al., 2013; Schenk et al., 2007, 2008] for tacking large-scale linear systems of equations when needed. We did not used SCIP's Mixed Integer Programming features since all our decision variables are real. It has essentially been used as an interpreter to the ZIMPL mathematical language [Koch, 2004] and a programming interface for IPOPT.

4.5.1 Client/Cache/Server bus

Consider the same bus topology as in Sec.4.4.2 above. Consider that exogenous requests address a catalog size of 80 objects. The content popularity follows a Zipf distribution of parameter 1. The cache budget is 10 objects. The link capacity from the cache to the client is 10 objects/s while the one from the server to the cache has a 20 objects/s capacity. Fig.4.5 depicts the results. LFU is clearly the proportionally fair caching policy. Also, observe that link capacities are shared proportionally to the content popularity. For instance, as anticipated by Eq.4.18, the sum of the ingress rates for content 1 is $2.9+3.1 = 6 = q_1 \times (C_{2,1} + C_{3,2}) = 0.2 \times (10 + 20)$.

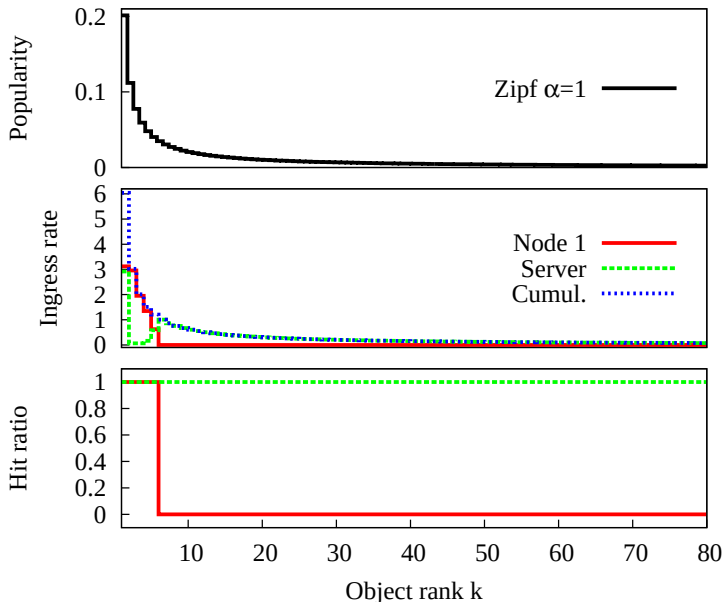


Figure 4.5 – Bus topology: proportionally fair allocation.

4.5.2 A simple network

Next, we evaluate a network of 3 cache-equipped routers/nodes and one server (Fig.4.6). The computational complexity of this nonconcave problem prevents the investigation of bigger instances. However, this setup suffices for characterizing the optima.

We set the total routers ingress link rates to respectively 10,15 and 20 objects/s. The capacity of the link to the content server is 30 objects/s. Cache capacities at every content

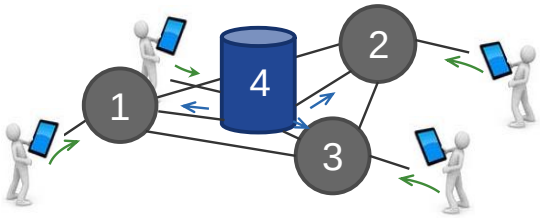


Figure 4.6 – Simple network topology.

router is 5, 6 and 7 objects. The content server is viewed as a node with a cache capacity that equals the catalog size.

The key insight is that every tractable instance entails an optimal caching consisting in the long-term storage of number of the most popular objects. This indicates that an LFU caching policy leads to content-wise α-fairness. Another implication is that at the optimum, the network does not convey any miss traffic. In other words, optimally, no interest crosses the nearest cache. Hence, the optimal network is a set of autonomous clusters centered on caches surrounded by their clients. Remember that such an optimum does not aim to guarantee interest satisfaction. We detail our observations below.

$(\alpha = 0)$-fairness

In the very particular case of zero-fairness, we consider a catalog of 2000 Zipf-ranked objects. The trivial optimum in Fig.4.7 depicts the whole network capacity being allocated to a single content, the most popular one. However, multiple optima exist, including a bandwidth allocation proportional to content popularity, as the problem turns out to be unweighted.

Proportional fairness

Proportional fairness first translates into ensuring a hit ratio of 1 to a majority of the most popular content. It is distinctive in Fig.4.8 where the Zipf skewness is 1 and Fig.4.9 featuring a Zipf parameter that equals 0.7. Given that persistent caching, fairness is actually enforced by an adequate link capacity sharing. The throughput fair share follows a curve matching content popularity, pointing out proportionality.

This is a strong result as it decouples caching and scheduling in the pursuit of content-wise fairness. Cache network fairness simplifies into legacy but content-wise queuing

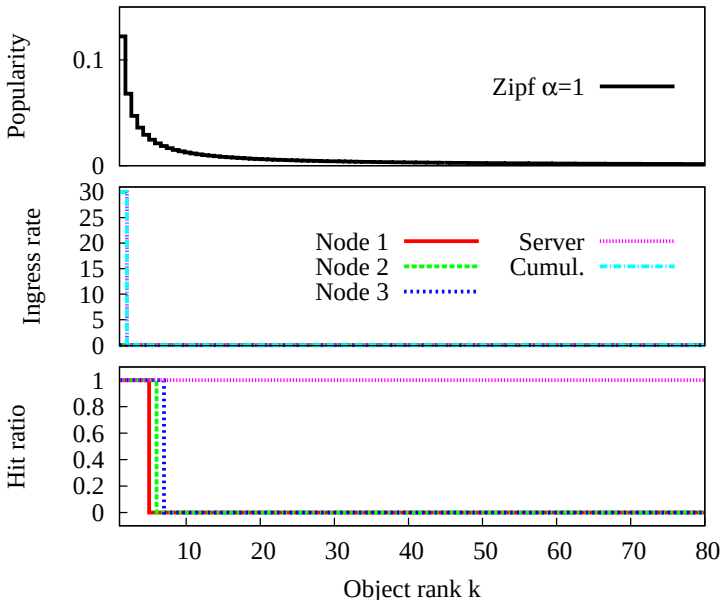


Figure 4.7 – Simple network: A zero-fair allocation for every content k at every cache n.

network fairness, given a few additional content servers formally known as caches.

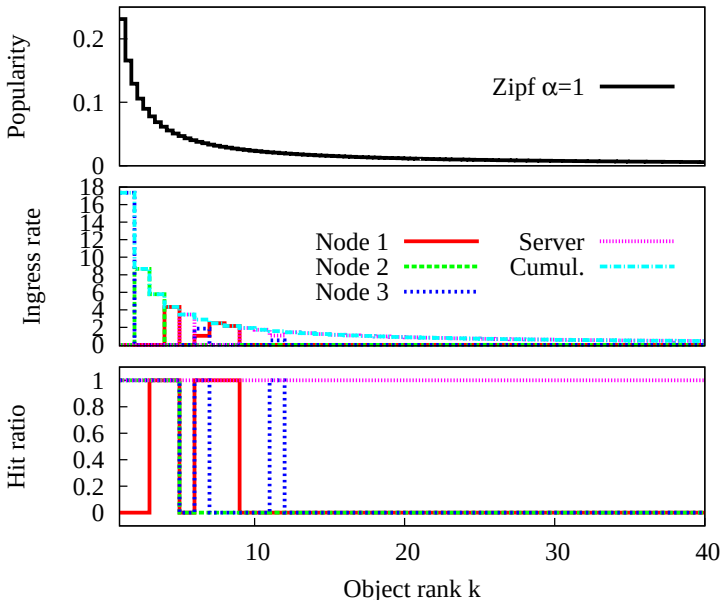


Figure 4.8 – Simple network: proportionally fair allocation for every content k at every cache n.

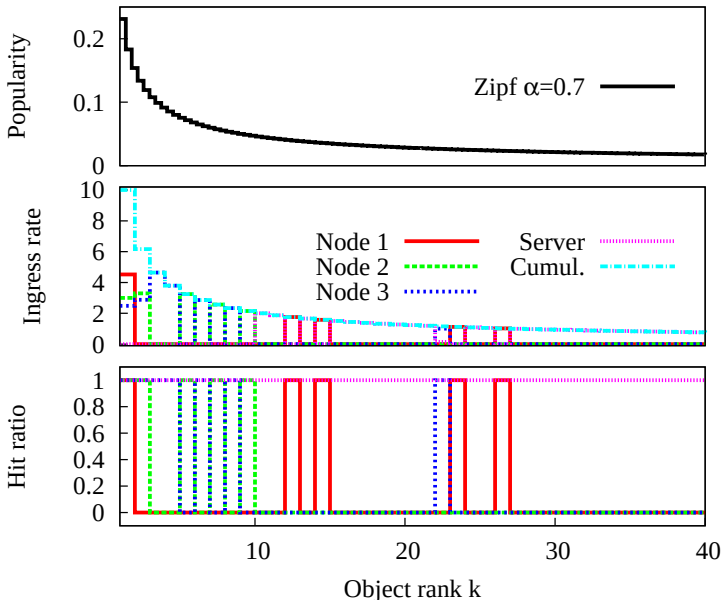


Figure 4.9 – Simple network: proportionally fair allocation with less skewed popularity distribution.

$(\alpha = 2)$-fairness

Even when $\alpha = 2$, fairness remains consistent with the previous observations. There is no fractional hit ratio. As a consequence, content-wise 2-fairness in a cache network does not require shared-time cache occupancy. The optimal link capacity share in this case is proportional to q_k where q_k is the probability that a requested object is of popularity rank k. It shows once more that the packet scheduler is entirely in charge of content-wise fairness.

Max-min fairness

We evaluate numerically max-min fairness as α-fairness with $\alpha = 9$. Computational limitations prevented us from exceeding this value. Although this is a quite loose approximation of an infinite α, the insight we get remains eloquent. As before, the recipe to fairness turns out to be persistent caching and a content-wise bandwidth fair sharing on top of the classical client-server network infrastructure. Observe that, as predicted analytically, fair resource sharing remains insensitive to α. The max-min fair share, just like in proportional fair sharing or any other case, are proportional to content popularity.

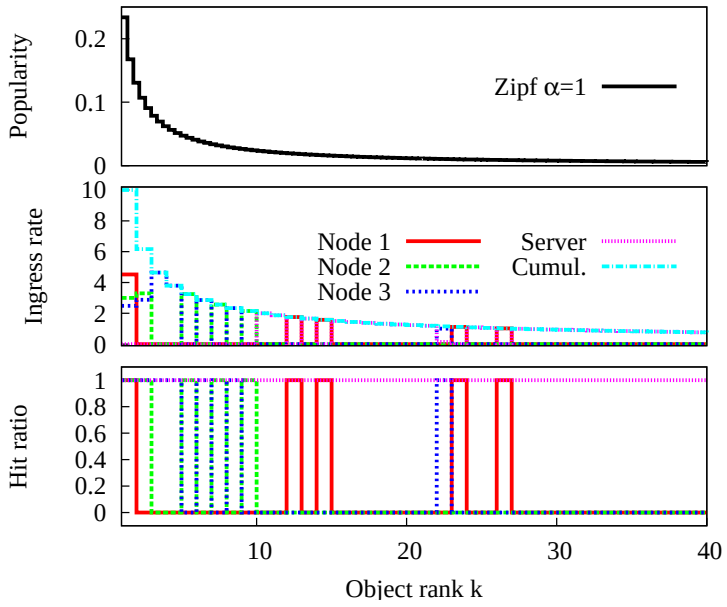


Figure 4.10 – Simple network: $(\alpha = 2)$-fair allocation for every content k at every cache n.

4.6 Conclusion

Cache networks, and more specifically Information-Centric Networks (ICN), are prominent solutions for communication infrastructure offloading. Early in the Internet history, cache engines have been inserted between the content consumers and the delivery servers to confine the recurrent traffic of very popular objects close the network edge. Nowadays, ICN, FemtoCaching [Golrezaei et al., 2012] and Fog-RAN [Sengupta et al., 2016; Shi et al., 2015] have furthered the ongoing caching penetration.

Throughout this chapter, we show that a resource allocation α-fair to content items, whatever $\alpha \geq 0$, can solely tackle the design of fair packet schedulers while ensuring that the most popular objects get permanently cached. In contrast to previous works, that focused on isolated caches, it appears that no fractional content hit ratios is necessary for the sake of fairness.

As a strong consequence, our analytic contribution suggests that content-wise fair allocation in cache network can be formulated within the existing frameworks pertaining to queuing networks [Massoulié and Roberts, 1999] in viewing as regular popular content servers caches equipped with LFU-approximating caching policies like p-LRU, LRU and

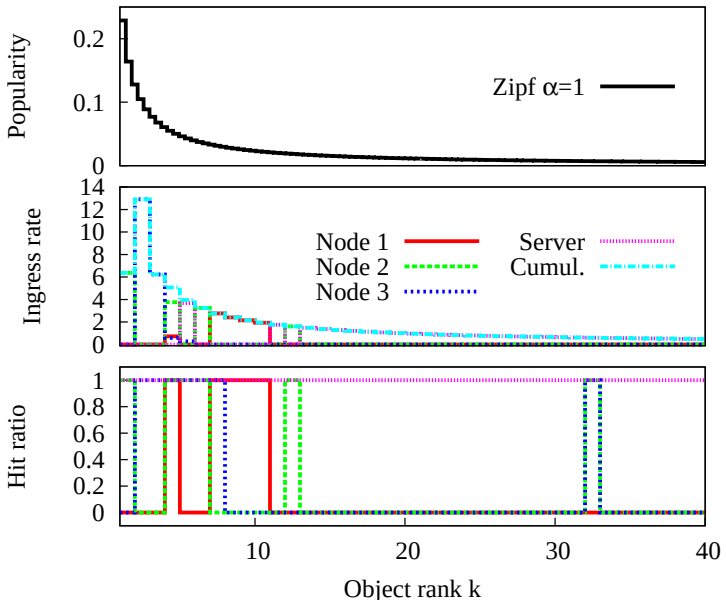


Figure 4.11 – Simple network: Approaching max-min fairness with ($\alpha = 9$)-fair allocation for every content k at every cache n.

Leave-Copy-Down heuristics or LAC+ [Carofiglio et al., 2015a]. To sum up, ICN can be α-fair, as long as the link service rate allocation is.

Chapter 5

Supervised Machine Learning-based Routing for NDN

Can ICN scale ?

Summary. *Named Data Networking (NDN) ambitions the rank of Future Internet Architecture in uniquely addressing content items by their name. In NDN, routers forward Interests for content after finding Longest-Prefix Matches (LPM) of content names in their Forwarding Information Base (FIB). However, the scalability of this structure is challenged by the huge global Internet namespace. In this chapter, we propose a novel approach to interest forwarding that compresses the FIB data structure into Artificial Neural Networks (ANNs). A bitwise trie splits the namespace and indexes ANNs. ANNs are offline trained by the control plane from the Routing Information Base and matching Interests. Then, they are made available to the data plane for interrogation. We demonstrate that this approach accelerates packet forwarding by several order of magnitude. Noteworthily, leveraging ANNs as memory and processor for directing packets towards next hops reminds of Asking For Directions to people in the street, incurring similar reliability regards.*

Keywords: *Information-Centric Networking; Scalable Forwarding; Machine Learning.*

Contents

5.1 Introduction

Information-Centric Networking (ICN) concentrates significant research effort to bring it to maturity. The elegance of the architecture and its adequateness to today's Internet usage makes it a prominent candidate to the next generation of network paradigms. This chapter is a contribution to its quest for scalability in presence of the global Internet namespace. As such, even if the enclosed proposal pertains to various implementations of ICN, we intend to emphasize the named content prerequisite in referring to Named Data Networking (NDN) [Zhang et al., 2014].

In NDN, users emit Interests for content items and retrieve data along the reverse path. Every traversed node maintains three major data structures: a Content Store (CS), which is a packet cache, a Pending Interest Table (PIT) making the NDN data plane stateful, and a Forwarding Information Base (FIB). More precisely, Interest packets that have been forwarded and waiting for matching Data packets to return have priorly been registered into the PIT, along with the requesting (inter)faces. The FIB contains prefixes and identified egress faces. It is kept up-to-date by a name-based routing plane. Its counterpart in the IP world might be the IP routing table. Upon an Interest arrival, a forwarding strategy identifies the corresponding egress faces from the content name by finding its longest matching prefix in the FIB. This operation is expensive and rather impossible to execute at wire speed without thorough optimization [Melazzi et al., 2013]. Several techniques have been investigated so far, most of them breaking names into logical components probed by means of hash tables [So et al., 2013; Fukushima et al., 2013], hierarchical hash tables [Yuan and Crowley, 2015] or distributed Bloom filters [Perino et al., 2014]. Such a hierarchical name assumption limits the scope of ICN to current DNS-like URLs and excludes the possibility of flat names. More recently, [Song et al., 2015] proposed to compress the FIB into a binary Patricia trie as the BSD kernel has done for IP addresses, and advocates for a speculative data plane.

Our work takes a radically new approach we call AFFORD (Ask For Directions). It considers every name $\vec{\mathbf{n}}$ as a whole and lets Artificial Neural Networks (ANNs) *learn*, *construct* a function f to quickly *compute*, for every egress face, a probability of relevance. ANNs, recently revived by Deep Learning [Deng and Yu, 2014], have regained some particular attention thanks to the tremendous computing power unleashed by modern CPUs and General-purpose Processing on Graphics Processing Units (GPGPU). The

broad range of machine learning applications might include packet forwarding as suggested throughout the chapter. An Artificial Neural Network mimics brains in consisting of interconnected nodes called neurons. A singular type of ANNs we hereafter manipulate is the *multilayer perceptron* (MLP), a feedforward ANN, which gathers artificial neurons into interconnected layers.

In a nutshell, we investigate the consequences of routers being trained to "guess" paths. We qualify the information retrieval via ANN-FIBs as guessing because of the unpredictable nature of the query outcome. Experiments show that ANN-FIBs, the FIB subsets implemented into ANNs are orders of magnitude smaller and faster, with accurate egress face guess. Conversely, it appears that supervised learning, which refers to the expensive process of presenting content names $\vec{\mathbf{n}}$ and matching bitmaps of egress faces $\vec{\mathbf{b}}$ to ANNs, has to be carried out offline, preferably inside the control plane. To this aim, the control plane would access the PIT for actual names, find their longest matching prefix in the Routing Information Base (RIB) populated by a named-based routing protocol and find, via a bitwise trie, the ANN-RIB in charge. Then it would train that ANN-RIB with pertaining $\vec{\mathbf{n}}$ and the target bitmap $\vec{\mathbf{b}}$ until a quadratic error $\|\vec{\mathbf{b}} - f(\vec{\mathbf{n}})\|_2$ gets minimal. This procedure involves a backpropagation algorithm [Rumelhart et al., 1985] that utilizes the well-known gradient descent technique. ANN-RIBs and their trie will be eventually copied as ANN-FIBs into the data plane for fast interrogation. As the namespace will grow, the accuracy of ANN-RIBs will decrease and the learning time will certainly increase. This is why instead of training always bigger ANN-RIBs, we propose a mechanism to dynamically split the namespace into tractable subsets by means of a bitwise trie.

The remainder of the chapter is organized as follows. Sec.5.2 describes related work. It sketches the recent community effort to mitigate the expensive FIB lookup. Then, Sec.5.3.1 and Sec.5.3.2 introduce our proposal, respectively AFFORD learning and forwarding algorithms. In Sec. 5.4 we analyze some intrinsic properties of our forwarding strategy and demonstrate it guarantees content delivery under no timeout assumption and mitigates flooding. AFFORD is evaluated in Sec.5.5. We investigate a diversity of scenarios to point out where AFFORD pertains. Then Sec.5.6 concludes the chapter and suggests perspectives.

5.2 Related work

Research on ICN has been prolific in pushing the limits of forwarding engines performance, aiming to help them sustain traffic at wire speed, regardless of the namespace size. Indeed in operation, it is likely that some ICN router will be required to fetch into 10^8-entries FIBs [So et al., 2013].

We can group contributions that mitigate huge FIB lookup cost into two categories. Those requiring hierarchical names and those capable of alleviating flat name forwarding as well. The majority belongs to the first category.

Hierarchical name forwarding

In this group, the iterative concatenation of an increasing number of name components (the maximum number denoted as d) allows to transform the Longest-Prefix Match (LPM) procedure into an $O(d)$ exact match at most. Among notable achievements is the implementation of a content router by [Perino et al., 2014], which queries a hash table-based FIB after probing contiguous Bloom filters (PBF) for prefix presence. Interestingly, their PBF (Prefix Bloom Filter) exploits the hierarchical name structure to the fullest. They compute a hash of a name component to identify the Bloom filter in charge. The component is either the first or subsequently another one to mitigate false positives in case of exceeded filter capacity.

[So et al., 2013] adopted a similar stance regarding hash table-based FIB while deeming Bloom filters less effective without hardware support. They offer to compensate the lack for hardware relief by innovating at the lookup phase. Thus, instead of looking for the longest-prefix first, which is vulnerable to DoS, the search starts from an intermediate component. This component's position M is a parameter that equals the component count of most FIB prefixes. Their solution expands the FIB with virtual prefixes whose purpose is to signal the existence of longer-than-M matches.

In a similar objective, [Fukushima et al., 2013] proposes a link layer protocol modification where routers convey within every Interest packet the component position they found a match at. Egress routers, likely to have similar prefixes in FIB, would greatly benefit from this information, number of them would realize $O(1)$ lookups. On the other hand, it means that every suffix in the FIB has to be expanded and labeled with its count of child suffixes (*i.e.*, the parent suffix and additional components). This is because a

protocol-advised FIB prefix is unambiguously the longest match only if that count is zero.

The method in [Yuan and Crowley, 2015] decreases the worst-case complexity of hash-based lookups to $O(log(d))$, in designing FIB as a binary tree of hash tables. d is the number of component in the name to process. Authors propose to store prefixes having the same number of components into the same hash table.

Whereas the above works did not use tries, [Wang et al., 2012] conducts longest-prefix match on a trie of compressed name components.

Name component-agnostic forwarding

Our proposal differs in that it does not require names to consist in sequences of components, even though some structure in the name, obvious or subtle, should improve the neural network performance.

Closer to our work, [Song et al., 2015] recently proposed to store the FIB into a compact binary *Patricia trie* [Morrison, 1968]. They do not assume any name structure. For sake of scalability, they also introduce a Speculative Binary Patricia (sBP) to conduct Longest-Prefix Classification (LPC) as a replacement of the conventional LPM. sBP stores in nodes the bit position to check for choosing the left or right child, in place of a token. Due to the resulting false positives increase, authors advocate for a speculative forwarding plane that could relieve FIB shortcomings.

5.3 AFFORD

Ask for Direction, abbreviated AFFORD, is a novel stochastic forwarding mechanism for named-object networking. In AFFORD, the probability to send an Interest to an egress face is calculated by ANN-FIBs, which are Artificial Neural Networks substituting regular FIBs. This contribution arises in times when content names are becoming network layer's first-class citizens, intended to make every single connected stakeholder reachable, from "things" to supercomputers. The number and length of these strings of characters referred to as names is deemed unlimited. Networked objects must be ready to cope with this unprecedented immensity. This perspective instills reservations about the perpetuation of current deterministic routing/forwarding models, exhaustively aware of the direction to every destination.

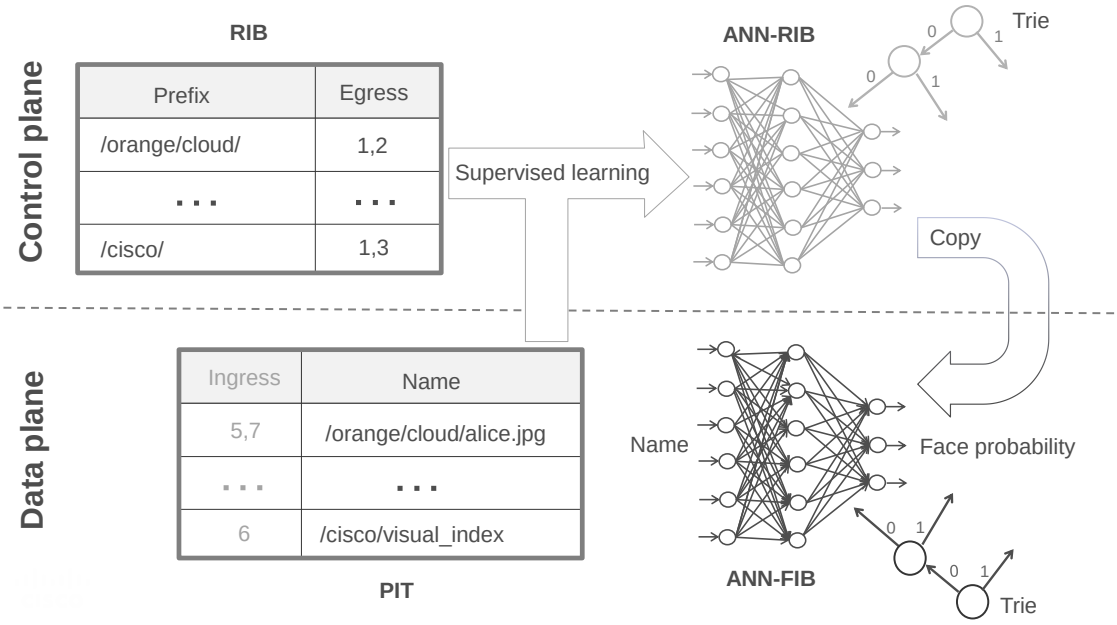


Figure 5.1 – The forwarding information management is depicted: the Artificial Neural Network is trained offline and made available to the data plane for online face selection.

Stochastic forwarding is an alternative we analyze below. For every object name, every egress face is affected a probability. This way, we relax the constraint of knowing the proper next hop with precision, hence the mandate to keep this information accurately. Stochastic forwarding is an unreliable forwarding but as demonstrated in Sec.5.4, unreliability and guaranteed delivery can coexist.

AFFORD operates in two parallel phases: the supervised learning phase that takes place inside the control plane, and the actual forwarding phase conducted inside the data plane.

5.3.1 AFFORD supervised learning

In machine learning, supervised learning is the training of a learning system with input signals and the target outputs. We choose in this chapter multilayer perceptrons (MLP) as learning systems.

Multilayer perceptrons

A MLP is an artificial neural network that organizes neurons into an input, one or several hidden layers and an output layer. Neurons from every layer feed those of the next one via synapses. A MLP works as follows: internally, it computes a value at every upper layer neuron that is the weighted sum of every input synapse, transformed by a nonlinear activation function. The logistic function is commonly adopted as activation function. That owes in part to its derivative, which only involves the function itself. This characteristic simplifies the implementation of the backward propagation of learning errors (backpropagation). Details about the backpropagation algorithm can be found in [Riedmiller, 1994; Rumelhart et al., 1985].

Layer setup

Define Γ^+ as the set of egress faces in the router. In AFFORD, a MLP reads on its input layer a signal consisting in an ASCII-encoded object name like $\vec{\mathbf{n}} \equiv (\text{ascii}('/'), \dots)$ and computes on its output layer a vector of probabilities $\vec{\mathbf{p}} \equiv (p_1, p_2, \cdots, p_i, \cdots)$. The probability position $i \in \Gamma^+$ in the vector is also the identifier of the corresponding face. The output vector components belong to $[0, 1]$, which is the codomain of the logistic activation function and, fortunately convenient for probabilities.

Note that only a single vector component can be read from or written in a neuron. Therefore, the number of input neurons is imposed by the maximal name length and the number of output neurons is $|\Gamma^+|$. The number of hidden layers a MLP should have is hard to determine, as well as the maximum number of training samples it must remember. In Sec.5.5, we successfully made a 3-layers and 35 neuron- MLP render the egress faces of a thousand names.

Learning purpose

Supervised learning aims at constructing a function $f : \mathbb{N}^m \to [0, 1]^{|\Gamma^+|}$ that assumes a maximal name length m. f must:

(i) render with the highest fidelity the training set in mapping accurately training sample names with the egress face bitmap;

(ii) estimate every face probability from longer names. We call this second role *Virtual Longest-Prefix Match* (VLPM). VLPM outputs a vector of probabilities that must

approximate to the face bitmap a LPM would have produced.

Learning procedure

The supervised learning of a MLP is an expensive process. It must be carried out periodically inside the control plane on the so-called ANN-RIBs, as soon as a routing protocol has updated the RIB. Each of the so-called ANN-RIBs runs in its own thread.

Since the learning time drastically increases with both MLP and dataset sizes, we propose to split the namespace with a bitwise trie. Every bottom level trie node references the ANN-RIB in charge of the subset of the namespace prefixed by its path in the trie.

Define $\Gamma^+_{\vec{\mathbf{n}}} \subseteq \Gamma^+$ as the subset of egress faces ensuring the delivery of named-content $\vec{\mathbf{n}}$. The *target* ANN-RIB output is a bitmap $\vec{\mathbf{b}} \equiv (\mathbb{1}_{\Gamma^+_{\vec{\mathbf{n}}}}(i))_{i \in \Gamma^+}$ indicating with certainty the egress faces for that name $\vec{\mathbf{n}}$. $\mathbb{1}_{\{\cdot\}}(\cdot)$ is the usual indicator function such that:

$$\mathbb{1}_{\Gamma^+_{\vec{\mathbf{n}}}}(i) = \begin{cases} 1 & \text{if } i \in \Gamma^+_{\vec{\mathbf{n}}}, \\ 0 & \text{otherwise.} \end{cases}$$

An ANN-RIB behaves as a function f_t at training time t. The sequence f_t is expected to converge pointwise to f. During the training phase, RIB prefix vectors are iteratively injected into the ANN-RIB in charge according to the bitwise trie. Then the ANN-RIB computes the error $\vec{\mathbf{b}} - f_t(\vec{\mathbf{n}})$ with respect to the target bitmap and backpropagates it to every lower layer neuron. Right after that, we do the same with PIT entries read from a PIT log and LPMatched with the RIB entries to obtain the target egress faces. RIB and PIT injection into ANN-RIBs are conducted iteratively until the following end conditions get fulfilled. Algorithm 5 summarizes all this.

Learning end

Let $V \equiv \left\{(\vec{\mathbf{n}}, \vec{\mathbf{b}}) \in \mathbb{N}^m \times \left\{0,1\right\}^{|\Gamma^+|}\right\}$ be a training set of *(name, interface bitmap)* pairs.

Definition 5.1 (Mean rendering error)**.** We define the mean rendering error $\mathrm{e}(t)$ at training time t as the mean absolute error between every target probability (in $\{0,1\}$) and the rendered probability (in $[0,1]$), averaged over the whole training set. It measures the accuracy of a ANN-RIB output with regard to the target bitmaps,

Algorithm 5: Perform supervised learning inside the control plane using RIB and PIT extracts. ANN-RIBs are trained to match names with egress faces and periodically copied into the data plane as ANN-FIBs.

```
At update time (every ΔT);
RIB.Update();
ANN-RIBs.Trie.Update();
repeat
    foreach RIBEntry in RIB do
        egressFaces ← RIBEntry.EgressFaces();
        ANN-RIB ← ANN-RIBs.Trie.Find(RIBEntry.Prefix());
        ANN-RIB.Learn(RIBEntry.Prefix(), egressFaces);
    end
    foreach PITEntry in PIT do
        RIBEntry ← RIB.LPMatchEntry(PITEntry.Name());
        egressFaces ← RIBEntry.EgressFaces();
        ANN-RIB ← ANN-RIBs.Trie.Find(RIBEntry.Prefix());
        ANN-RIB.Learn(PITEntry.Prefix(), egressFaces);
    end
until ANN-RIBs.Trained();
ANN-FIBs ← ANN-RIBs;
```

when given **exact** names from the training set. Formally,

$$e(t) = (|V||\Gamma^+|)^{-1} \sum_{(\vec{\mathbf{n}},\vec{\mathbf{b}})\in V} \|\vec{\mathbf{b}} - f_t(\vec{\mathbf{n}})\|_1. \tag{5.1}$$

Synapse weights, updated by backpropagation, are only saved when $e(t)$ decreases. Learning stops when a maximum number of iterations is reached or when, given a parameter κ,

$$e(t) > \kappa \times \inf\{e(l), l < t\}. \tag{5.2}$$

We set in this work κ to 1.5. If the end condition (5.2) is fulfilled, $e(t)$ likely reached its global minimum. Therefore, the ANN-RIB must restore the weights it saved at $e(t)$ minimum.

As soon as an ANN-RIB's configured capacity exceeds (for instance, a maximum of 1000 entries to render), add a new level of nodes to the bitwise trie. Then, move and duplicate the ANN-RIBs attached the parent nodes into the child nodes. Finally, re-train the ANN-RIBs for them to focus on the smaller namespace prefixed by their path into the

trie.

In parallel, inside the data plane, ANN-RIBs lighter copies, the ANN-FIBs will be interrogated for fast forwarding decisions. By lighter copies, we mean that a few internal MLP fields like the propagated errors or the synapse weight differentials used by the backpropagation algorithm during the learning phase are irrelevant in the data plane and should not be copied.

5.3.2 AFFORD forwarding

Algorithm 6: Use the Artificial Neural Networks ANN-FIBs jointly as forwarding information memories and longest-prefix match processors.

```
Input: Interest packet, s
ANN-FIB ← ANN-FIBs.Trie.Find(Interest.Name());
egressFaces ← ANN-FIB.Query(Interest.Name());
egressFaces.SortByProbability();
rank ← 1;
foreach face in egressFaces do
    if rank / egressFaces.Size() ≤ s then
        face.SetProbability( ceil( face.GetProbability() ) );
        rank++;
    end
    if rand() / RAND_MAX < face.GetProbability() then
        Forwarder.Forward(face.ID(), Interest);
    end
end
```

The actual forwarding in AFFORD is straightforward. As summarized in Algorithm 6, it obeys to the following few steps:

- Use the name carried in the Interest packet to locate, by means of the bitwise trie, the ANN-FIB in charge.
- Interrogate the ANN-FIB in charge of that namespace segment and retrieve a vector of face probabilities.
- Set to 1 a fraction s of the highest probabilities. This is to ensure content delivery in a single attempt through the corresponding faces.
- Send the Interest towards interface i with probability p_i.

5.4 Analysis

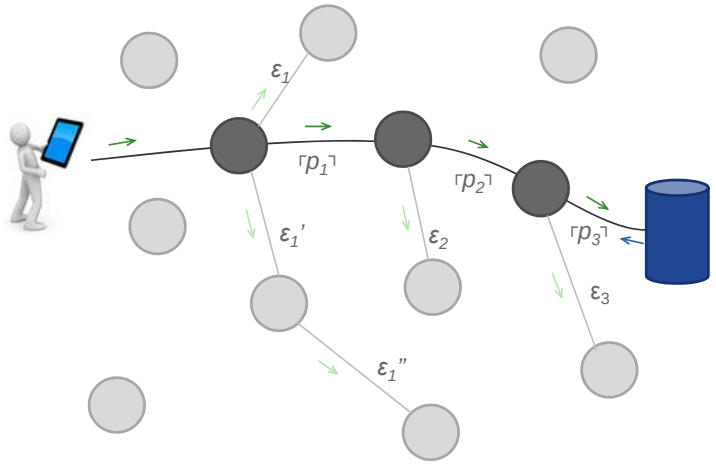


Figure 5.2 – Forwarding under unreliable information: diffuse with probability ϵ interests along low probability paths and deterministically convey interests along high probability paths. Diffusion along low probability paths eventually fades, bounding flooding.

Asking for directions supposes some level of unreliability in the response. This is also true with Artificial Neural Networks, which by design render learned samples with an imperfect accuracy. We show in this section that forwarding under unreliable information still guarantees delivery in a finite mean number of attempts. This holds under the assumption that there is no PIT timeout.

Let $\mathcal{R}_k$ be the set of routes from the user to content k provider. A route $r \in \mathcal{R}_k$ is a sequence of nodes n. Define $p_{k,r,n}$ as the ANN-calculated probability that content k is delivered through node n's successor along route r. At each node, the top fraction s of egress routes see their $p_{k,r,n}$ rounded to 1.

Proposition 5.1 (Guaranteed delivery)**.** AFFORD guarantees content k delivery in a single attempt along at least $(s^{\sup|r|} \times 100)$-percent of the routes and in average in less than $(\prod_{n \in r} p_{k,r,n})^{-1}$ attempts along every route r.

Proof. The maximum route length from the user to the content k provider is $L \equiv \sup\{|r|, r \in \mathcal{R}_k\}$. Define route R such that $|R| = L$. Therefore, the fraction of the crossed routes $r \ni n, n \in R$ having their probability rounded to $1 = \lceil p_{k,r,n} \rceil$ is at least $\prod_{n \in R} s = s^L$.

In a worst case scenario where the ANN locally assigns some route a weak probability, the lowest probability that an interest gets eventually satisfied through such a route r is $\prod_{n \in r} p_{k,r,n}$. Inverting the latter gives a lower bound to the mean number of necessary attempts. □

AFFORD may assign non-zero probability to unfeasible routes as a computational artifact. The following corollary ensures that delivery attempts through such routes should ultimately fade, limiting network flooding.

Corollary 5.1.1 (Unfeasible route fading)**.** Stochastic forwarding through routes comprising low probabilities $\epsilon_{k,r,n} < 1$ fades as

$$\lim_{|r| \to \infty} \prod_{n \in r} \epsilon_{k,r,n} = 0. \tag{5.3}$$

5.5 Evaluation

In this section we experimentally verify the feasibility of AFFORD. We aim at checking for potential gains in replacing of the regular FIB structure with an Artificial Neural Network (ANN-FIB). For this purpose, we based our implementation on a free Multilayer Perceptron written in C++ and available under MIT license at `https://github.com/sylbarth/mlp`. The enclosed activation function is the logistic function $A(x) = (1+e^{-\gamma x})^{-1}$ where its steepness γ is set to 0.2. The backpropagation algorithm relies on a gradient descent configured with learning rate η set to 0.25. We modified its backpropagation algorithm to alleviate an encountered inertia. That inertia was due to a non-standard dependency of new weight updates on the old ones. ANN's input is every name character and its output is a vector of egress face probabilities. For example, the ASCII code of each character in $(/, O, r, a, n, g, e, /, c, l, o, u, d, /)$ padded with zeros is given to a neuron in the input layer and $(0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$ is the target output vector meaning "forward to face 2 with certainty".

Content popularity follows a Zipf distribution with skewness parameter $\alpha = 1$. Names are randomly generated and consist of alphanumeric characters, "_" and "-" . Still, some structure is enforced. One third of the name characterizes its producer identifier, one third is the actual content name and the last part is a chunk identifier. All names are deemed inserted into the router's FIB. We compare ANN-FIB interrogations to regular FIB's character-by-character longest-prefix match, for the lookup to be name component agnostic. For all indicated figures to be easily reproducible, we chose to run these tests on off-the-shelf hardware equipped with an Intel Core i5 2.5GHz quad-core CPU with 3MB L3 cache and 4GB of DDR3 SDRAM. Fig.5.3, Fig.5.4 and Fig.5.5 depict the results.

5.5.1 Tiny-size FIB

We first evaluate ANN-FIB performance under a tiny catalog of 100 objects (or chunks). ANN-FIB has to be interrogated for forwarding each Interest through one among 10 egress faces. Name are 20 characters length maximum. We first configure a 3-layer perceptron. The first layer, the input one, contains 20 neurons. The third layer, the output one, consists in 10 neurons. The middle layer, which is the hidden one, consists in $(20+10)/2 = 15$ neurons. After training ANN-RIB for $22s$, it provides a rendering accuracy of about 10^{-3}. Fig.5.4 reports that the mean absolute error of the rendering is rapidly

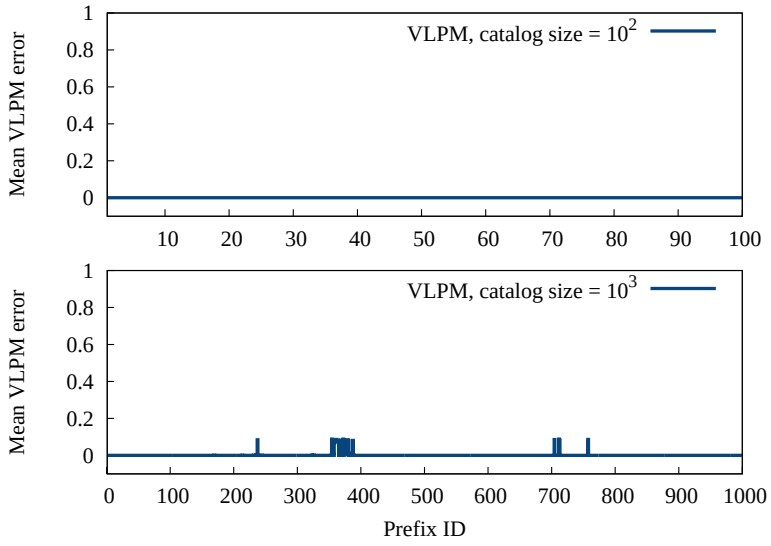
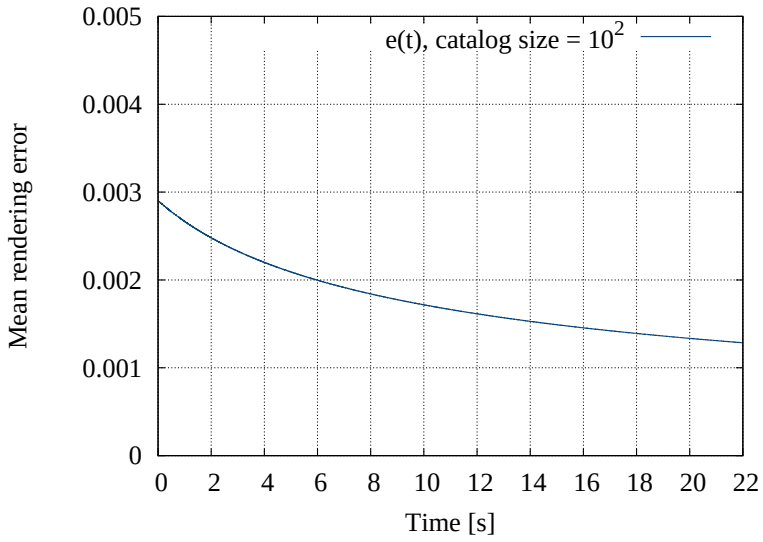
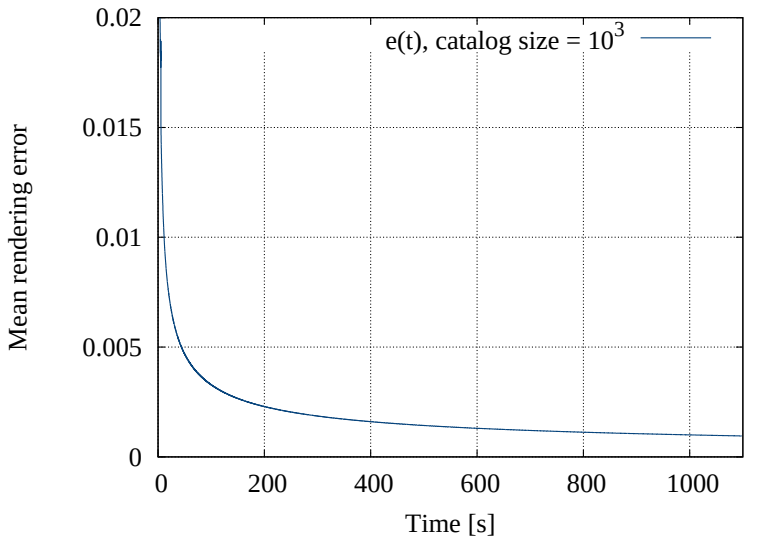


Figure 5.3 – Evaluation results: accurate Virtual Longest-Prefix Match.



(a) Error vs Learning Time, catalog = 10^2 objects



(b) Error vs Learning Time, catalog = 10^3 objects

Figure 5.4 – Evaluation results: Low mean rendering error $e(t)$.

satisfactory. Define the Virtual Longest-Prefix Match (VLPM) error as the mean absolute error between every training sample's output probability vector and 10^4 output probability vectors of the same training sample padded with random characters. Fig.5.3 reports a mean Virtual Longest-Prefix Match error of almost zero. It shows that the ANN-FIB correctly maps prefixes and complementing names towards the same egress faces. 10^7 in-

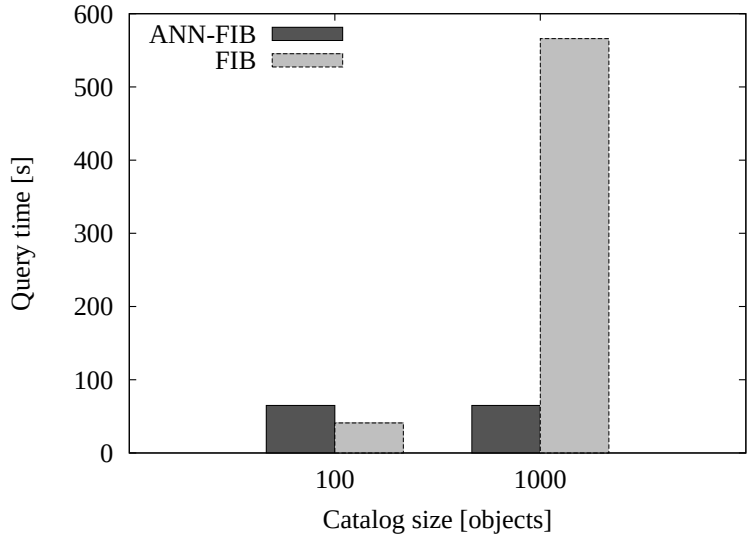
terrogations using ANN-FIB take $65s$ whereas the longest-prefix match on a regular FIB only takes $41s$. Both FIB and ANN-FIB sizes are approximately 2KB. We observe here that in spite of its remarkable accuracy, ANN-FIB does not demonstrate any clear benefit.

5.5.2 Medium-size FIB

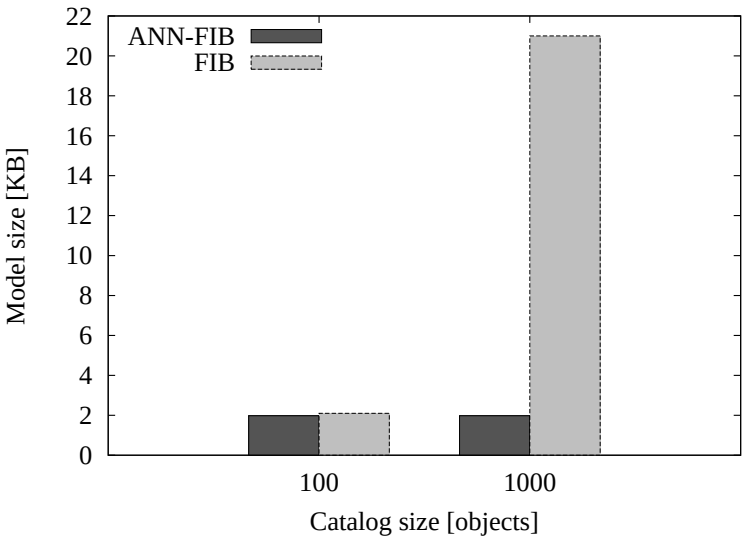
Second, we evaluate ANN-FIB performance under a medium-size catalog of a thousand objects (or chunks). Like previously, ANN-FIB is a 3-layer perceptron. Layers consisting respectively of 20, 15 and 10 neurons. After training ANN-RIB for $1098s$, it renders egress faces with an average accuracy lower than 10^{-3}. As showed in Fig.5.3 by the very low mean Virtual Longest-Prefix Match error, an ANN-FIB is generally capable of performing accurate LPM without being taught the actual LPM algorithm. 10^7 interrogations using ANN-FIB still take $65s$ whereas the longest-prefix match on a regular FIB takes $566s$. ANN-FIB sizes is still 2KB whereas FIB size is now 10 times bigger. ANN-FIB exhibits striking scalability in being approximately one order of magnitude smaller and faster than a regular FIB structure.

5.5.3 Big-size FIB

Finally, we consider a catalog of 10^5 objects. We enter a domain where the namespace has to be split for efficiency. The bitwise trie indexes 96 ANN-FIBs. ANN-FIBs remain 3-layer perceptrons. Layers gather respectively of 20, 15 and 10 neurons. 10^7 interrogations using ANN-FIBs take pretty much the same time as in the medium-size evaluation whereas the longest-prefix match on a classical FIB takes 1.4 day. ANN-FIB total size approximates to 200KB whereas FIB size is 10 times bigger. ANN-FIBs exhibits here also remarkable scalability in being more than 3 orders of magnitude (*i.e.*, a thousand times) faster than a regular FIB structure. To be fair, this comparison should have considered an improved, trie-indexed distributed FIB instead of a monolithic FIB. It would have significantly reduced the number of entries to perform longest-prefix match on. In doing so, comparative results would have fallen back to those of the medium-size scenario.



(a) Query time vs catalog size



(b) Model size vs catalog size

Figure 5.5 – Evaluation results: ANN-FIB size and speed outperform regular FIB.

5.6 Conclusion and future work

FIB scalability is a major impediment to future NDN adoption. This chapter proposes a novel research direction: replacing the FIB table with Artificial Neural Networks, ANN-FIBs, offline trained inside the control plane. We call this new forwarding scheme AFFORD after similar procedure in streets where human brains store, process and forward

inquiries. The accuracy, speed and size of the ANN-FIB outperforming conventional FIB by orders of magnitude, are particularly promising. Possible applications might dwell in core network ICN routers. The control plane supervising ANN-RIB learning might be local or centralized within an SDN controller. Future work includes speeding up the learning process by GPGPU and improving the estimation accuracy by investigating various types of neural networks and activation functions. Ultimately, we target the challenge of replacing billion-entries FIB with this solution.

Chapter 6

Conclusion and future work

In this thesis we have demonstrated the benefits of equipping LRU caches with a latency-aware add-on. Indeed, making an early caching decision using the latency of retrieved objects, on the purpose to minimize content delivery time, is as effective as intuitive. The LAC+ mechanism is, to the best of our knowledge, the first recipe for fast p-LRU convergence towards LFU. ICN being increasingly regarded as a 5G forwarding plane candidate, [Kutscher, 2016] sees in our fully-distributed latency-aware caching algorithms advances in the quest for 5G ultra-low latency objective.

Multipath forwarding commands a joint caching and forwarding optimization because of their interdependence. This is why we elaborate FOCAL, a latency-aware joint caching and forwarding scheme that combines LAC+, single-path forwarding for the most popular objects (Perf) and load balancing (LB) for the others. We analyzed it theoretically and through extensive simulations. FOCAL is able to quickly adapt to network conditions and traffic characteristics and cuts by more than half the mean delivery time achieved by LRU, typically on Abilene-like scenarios (Sec.3.7.3).

We have analyzed ICN fairness from the viewpoint of the throughput allocation to contents. We have demonstrated that ICN is ready for α-fair content delivery inspite of caching algorithms that utterly favor the most popular objects. We identified the sub-system where α-fairness should be enforced to remain, as in traditional networks, the network schedulers.

These works simplify the modeling of ICN by capturing in the same equation cache and communication links contributions to the dynamics of local Interest packet backlogs,

being aligned with [Yeh et al., 2014]. To the best of our knowledge, no other work has evaluated both theoretically and empirically networks of caches under bandwidth constraints.

We have contributed to the research effort towards FIB scalability in proposing AFFORD, a supervised machine learning-based routing scheme. Supervised machine learning algorithm and parameter selection being test-driven [Kotsiantis et al., 2007; Amancio et al., 2014], more artificial neural networks, hardware infrastructures, learning algorithms remain to be investigated. We believe such an example of dealing with imprecise routing information, to be one of the few sustainable paths towards the forthcoming named Internet-of-Everything era, due to its unlimited namespace.

Yes, we believe in-network caching to be here to stay. Number of next-generation networking use cases, including pervasive video, tactile Internet or high mobility over heterogeneous networks, will require it. Cached content will need to be fetched and trusted. So far, we can not figure out any way to proceed other than naming data and cryptographically binding name, data and producer through signature. Reliable multipath and multicast native support must definitely become a basic network service. Eventually, that network may not be called ICN, but the lessons learned from the current research effervescence about ICN shall end up in an heir technology.

Significant efforts remain to be put into off-path caching investigation. Intuitively, a wealth of inter-domain traffic would be saved if a node could find the nearest replica to satisfy an Interest instead of being limited to on-path caches. Whereas [Fayazbakhsh et al., 2013] doubts it is worth the routing overhead (probably due to the lack of coordination among caches in their experiments), [Saino et al., 2013] reported about 30% inter-domain traffic reduction by spreading content objects across intra-domain caches and routing to them, both using hash values. By measuring data retrieval latency, LAC+ is able to rule out those coming from intra-domain caches and to perform implicit coordination. The challenge is to keep nodes updated about the nearest replica. Can we rely on the control plane for this, considering the transient nature of cached objects? An idea would be to only advertise those having the highest hit ratio. Updates might be less frequent assuming that, at some timescale, cache dynamics reach a steady-state. Optionally, AFFORD would learn egress faces towards content replicas and the corresponding hit ratios. Stochastic forwarding would diffuse interests according to either advertised hit ratios or Artificial Neural Network-rendered probabilities.

Another concern is whether popularity can be learned from deeper in the access network, considering content churn. [Elayoubi and Roberts, 2015] proposed to prefetch objects instead of caching them reactively, exploiting for this the knowledge gathered closer to the network core where traffic aggregation provides the most relevant popularity figures. More recently, [Leconte et al., 2016] took a similar stance, advocating learning about popularity in clusters of correlated caches and pre-fetching local LRU caches by faking Interest for the globally-identified most popular objects. Interestingly, they pointed out that caching content's first chunks deeper into the access network suffices to decrease latency, hence to improve user QoE. The relevance of such global popularities for very local interests must be questioned. While this problem was demonstrated in [Leconte et al., 2016], authors do not give any practical way to identify the local caches to cluster on how good they correlate under Shot Noise traffic. Moreover, both works command centralized approaches requiring controllers, management and explicit signaling. We observed that, in a signaling-free way, LAC+ quickly picks and firmly holds globally popular objects that are locally meaningful, based on how delivery time drops when they are stored in cache.

These are, in our opinion, research directions it would be interesting to explore experimentally by means of real traffic over real NDN testbeds. The source of latency, real traffic characteristics, the complexity of the interaction between flows and the role of congestion control, might command adjustments or re-design. NDN will certainly mature being used and confronted to reality, as it has been for the TCP/IP stack (*i.e.*, the Internet protocol suite) for more than three decades.

Bibliography

Achterberg, T. (2009). SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41. http://mpc.zib.de/index.php/MPC/article/view/4.

Ahlgren, B. and al. (2008). Design considerations for a network of information. In *Proc. of ACM CoNEXT*.

Amancio, D. R., Comin, C. H., Casanova, D., Travieso, G., Bruno, O. M., Rodrigues, F. A., and da Fontoura Costa, L. (2014). A systematic comparison of supervised classifiers. *PloS one*, 9(4):e94137.

Augé, J., Carofiglio, G., Grassi, G., Muscariello, L., Pau, G., and Zeng, X. (2015). Anchor-less Producer Mobility in ICN. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 189–190. ACM.

Badov, M., Seetharam, A., Kurose, J., Firoiu, V., and Nanda, S. (2014). Congestion-aware caching and search in information-centric networks. In *Proceedings of the 1st International Conference on Information-centric Networking*, ICN '14, pages 37–46, New York, NY, USA. ACM.

Bansal, S. and Modha, D. S. (2004). CAR: Clock with adaptive replacement. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, FAST '04, pages 187–200, Berkeley, CA, USA. USENIX Association.

Ben-Porat, U., Bremler-Barr, A., and Levy, H. (2013). Vulnerability of network mechanisms to sophisticated ddos attacks. *IEEE Transactions on Computers*, 62(5):1031–1043.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Bertsekas, D. P., Gallager, R. G., and Humblet, P. (1992). *Data networks*, volume 2. Prentice-Hall International New Jersey.

Bianchi, G., Detti, A., Caponi, A., and Blefari Melazzi, N. (2013). Check before storing: What is the performance price of content integrity verification in LRU caching? *SIGCOMM Comput. Commun. Rev.*, 43(3):59–67.

Blaze, M., Feigenbaum, J., and Lacy, J. (1996). Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173. IEEE.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE.

Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, J., Gjessing, S., Fairhurst, G., Griwodz, C., and Welzl, M. (2014). Reducing internet latency: A survey of techniques and their merits.

Carlucci, G., De Cicco, L., and Mascolo, S. (2015). HTTP over UDP: an experimental investigation of QUIC. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 609–614. ACM.

Carofiglio, G., Gallo, M., and Muscariello, L. (2012). Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *proc. of ACM Sigcomm ICN workshop*.

Carofiglio, G., Gallo, M., and Muscariello, L. (2013a). Bandwidth and Storage Sharing Performance in Information Centric Networking. *Elsevier Science, Computer Networks Journal, Vol.57, Issue 17*.

Carofiglio, G., Gallo, M., and Muscariello, L. (2013b). On the performance of bandwidth and storage sharing in information-centric networks. *Comput. Netw.*, 57(17):3743–3758.

Carofiglio, G., Gallo, M., Muscariello, L., Papalini, M., and Wang, S. (2013c). Optimal Multipath Congestion Control and Request Forwarding in Information-Centric Networks. In *Proc. of IEEE ICNP*.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015a). Analysis of Latency-Aware Caching Strategies in Information-Centric Networking. In *Proc. of ACM CoNEXT, CCDWN Workshop*.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015b). FOCAL: Forwarding and Caching with Latency awareness in Information-Centric Networking. In *Proc. of IEEE GLOBECOM (WKSHPS), ICNS*.

Carofiglio, G., Mekinda, L., and Muscariello, L. (2015c). LAC: Introducing latency-aware caching in information-centric networks. In *Proc. of IEEE LCN*.

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. (2007). Delay-tolerant networking architecture. Technical report, IETF.

Che, H., Tung, Y., and Wang, Z. (2006). Hierarchical web caching systems: Modeling, design and experimental results. *IEEE J.Sel. A. Commun.*, 20(7):1305–1314.

Choungmo Fofack, N. E., Nain, P., Neglia, G., and Towsley, D. (2012). Analysis of TTL-based Cache Networks. In *ValueTools - 6th International Conference on Performance Evaluation Methodologies and Tools - 2012*, Cargèse, France. RR-7883 : http://hal.inria.fr/hal-00676735/.

Cinlar, E. (2013). *Introduction to stochastic processes*. Courier Corporation.

Cisco, V. N. I. (2016). The zettabyte era–trends and analysis. *Cisco white paper*.

Dan, A. and Towsley, D. (1990). An approximate analysis of the lru and fifo buffer replacement schemes. *Proc of. SIGMETRICS*.

Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., and Tay, Y. (2016). A utility optimization approach to network cache design. *arXiv preprint arXiv:1601.06838*.

Dehghan, M., Seetharam, A., Jiang, B., He, T., Salonidis, T., Kurose, J., Towsley, D., and Sitaraman, R. (2015). On the complexity of optimal routing and content caching in heterogeneous networks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 936–944. IEEE.

Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3–4):197–387.

Elayoubi, S.-E. and Roberts, J. (2015). Performance and cost effectiveness of caching in mobile access networks. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 79–88. ACM.

Ellison, C. et al. (1996). Establishing identity without certification authorities. In *USENIX Security Symposium*, pages 67–76.

Eum, S., Nakauchi, K., Murata, M., Shoji, Y., and Nishinaga, N. (2012). CATT: Potential Based Routing with Content Caching for ICN. In *Proc. of ACM SIGCOM ICN Workshop*.

Fayazbakhsh, S. K., Lin, Y., Tootoonchian, A., Ghodsi, A., Koponen, T., Maggs, B., Ng, K., Sekar, V., and Shenker, S. (2013). Less pain, most of the gain: Incrementally deployable icn. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 147–158, New York, NY, USA. ACM.

Floyd, S. and Fall, K. (1999). Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking (ToN)*, 7(4):458–472.

Fredj, S. B., Bonald, T., Proutiere, A., Régnié, G., and Roberts, J. W. (2001). Statistical bandwidth sharing: a study of congestion at flow level. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 111–122. ACM.

Fricker, C., Robert, P., and Roberts, J. (2012). A versatile and accurate approximation for LRU cache performance. In *Proceedings of the 24th International Teletraffic Congress*, ITC '12, pages 8:1–8:8. International Teletraffic Congress.

Fukushima, M., Tagami, A., and Hasegawa, T. (2013). Efficiently looking up non-aggregatable name prefixes by reducing prefix seeking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 340–344. IEEE.

Gallo, M., Kauffmann, B., Muscariello, L., Simonian, A., and Tanguy, C. (2012). Performance evaluation of the random replacement policy for networks of caches. *CoRR*, abs/1202.4880.

Garcia-Luna-Aceves, J., Dabirmoghaddam, A., and Mirzazad-Barijoug, M. (2014). Understanding optimal caching and opportunistic caching at" the edge" of information-centric networks. In *Proceedings of the 1st international conference on Information-centric networking*.

Garetto, M., Leonardi, E., and Martina, V. (2016). A unified approach to the performance analysis of caching systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 1(3):12.

Garetto, M., Leonardi, E., and Traverso, S. (2015). Efficient analysis of caching strategies under dynamic content popularity. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2263–2271. IEEE.

Gasti, P., Tsudik, G., Uzun, E., and Zhang, L. (2013). Dos and ddos in named data networking. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7. IEEE.

Gelenbe, E. (1973). A unified approach to the evaluation of a class of replacement algorithms. *IEEE Transactions on Computer*, 22(6):611–618.

Georgiadis, L., Neely, M. J., and Tassiulas, L. (2006). *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc.

Ghali, C., Tsudik, G., and Uzun, E. (2014). Network-layer trust in named-data networking. *ACM SIGCOMM Computer Communication Review*, 44(5):12–19.

Golrezaei, N., Shanmugam, K., Dimakis, A. G., Molisch, A. F., and Caire, G. (2012). Femtocaching: Wireless video content delivery through distributed caching helpers. In *INFOCOM, 2012 Proceedings IEEE*, pages 1107–1115. IEEE.

Habib, S., Qadir, J., Ali, A., Habib, D., Li, M., and Sathiaseelan, A. (2016). The past, present, and future of transport-layer multipath. *arXiv preprint arXiv:1601.06043*.

Hamilton, R., Iyengar, J., Swett, I., and Wilk, A. (2016). QUIC: A UDP-based secure and reliable transport for HTTP/2. *IETF, draft-tsvwg-quic-protocol-02*.

He, J. and Rexford, J. (2008). Toward internet-wide multipath routing. *IEEE network*, 22(2):16–21.

Hemmati, E. and Garcia-Luna-Aceves, J. (2015). A new approach to name-based link-state routing for information-centric networks. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 29–38. ACM.

Hochbaum, D. S. (2007). Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, 153(1):257–296.

Hoque, A., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., and Wang, L. (2013). Nlsr: named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 15–20. ACM.

Housley, R., Polk, W., Ford, W., and Solo, D. (2002). Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. Technical report.

Imbrenda, C., Muscariello, L., and Rossi, D. (2014). Analyzing Cacheable Traffic in ISP Access Networks for Micro CDN Applications via Content-centric Networking. In *Proc. of ACM ICN*.

Ioannou, A. and Weber, S. (2014). Towards on-path caching alternatives in information-centric networks. In *Proc. of IEEE LCN (Poster)*.

Ion, M., Zhang, J., and Schooler, E. M. (2013). Toward content-centric privacy in icn: Attribute-based encryption and routing. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 39–40. ACM.

Jacobson, V., Smetters, D., Thornton, J., and al. (2009a). Networking named content. In *Proc. of ACM CoNEXT*.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009b). Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12, New York, NY, USA. ACM.

Jang, K., Han, S., Han, S., Moon, S. B., and Park, K. (2011). Sslshader: Cheap ssl acceleration with commodity processors. In *NSDI*.

Jelenković, P. R. (1999). Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *Annals of Applied Probability*, pages 430–464.

Jelenković, P. R. and Kang, X. (2008). Characterizing the miss sequence of the lru cache. In *in Proc. of ACM SIGMETRICS, MAMA Workshop*.

Jelenković, P. R. and Radovanović, A. (2004). Optimizing LRU caching for variable document sizes. *Comb. Probab. Comput.*, 13(4-5):627–643.

Katsaros, K. V., Chai, W. K., Wang, N., Pavlou, G., Bontius, H., and Paolone, M. (2014). Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications. *IEEE Network*, 28(3):58–64.

Kelly, F. P., Maulloo, A. K., and Tan, D. K. (1998). Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252.

Kleinberg, R. (2007). Geographic routing using hyperbolic space. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 1902–1909. IEEE.

Koch, T. (2004). *Rapid Mathematical Prototyping*. PhD thesis, Technische Universität Berlin.

Koponen, T., Chawla, M., Chun, B., Ermolinskiy, A., Kim, K., Shenker, S., and Stoica, I. (2007). A data-oriented (and beyond) network architecture. In *Proc. of ACM SIGCOMM*.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. (2010). Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106.

Kurose, J. (2014). Information-centric networking: The evolution from circuits to packets to content. *Computer Networks*, 66:112–120.

Kutscher, D. (2016). It's the network: Towards better security and transport performance in 5g. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 656–661.

Kuzmin, A., Luisier, M., and Schenk, O. (2013). Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations. In Wolf, F., Mohr, B., and Mey, D., editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 533–544. Springer Berlin Heidelberg.

Laoutaris, N., Che, H., and Stavrakakis, I. (2006). The LCD interconnection of LRU caches and its analysis. *Elsevier Science, Performance Evaluation*.

Laoutaris, N., Syntila, S., and Stavrakakis, I. (2004). Meta algorithms for hierarchical web caches. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 445–452.

Leconte, M., Paschos, G., Gkatzikis, L., Draief, M., Vassilaras, S., and Chouvardas, S. (2016). Placing dynamic content in caches with small population. In *Computer Communications (INFOCOM), 2016 IEEE Conference on*.

Leonardi, E. and Torrisi, G. L. (2015). Least recently used caches under the Shot Noise Model. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 2281–2289. IEEE.

Li, Z. and Simon, G. (2011). Time-shifted tv in content centric networks: The case for cooperative in-network caching. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Llorca, J., Tulino, A. M., Guan, K., Esteban, J., Varvello, M., Choi, N., and Kilper, D. C. (2013). Dynamic in-network caching for energy efficient content delivery. In *INFOCOM, 2013 Proceedings IEEE*, pages 245–249. IEEE.

Lychev, R., Jero, S., Boldyreva, A., and Nita-Rotaru, C. (2015). How secure and quick is QUIC? provable security and performance analyses. In *2015 IEEE Symposium on Security and Privacy*, pages 214–231. IEEE.

Martina, V., Garetto, M., and Leonardi, E. (2013). A unified approach to the performance analysis of caching systems. *CoRR*, abs/1307.6702.

Massoulié, L. and Roberts, J. (1999). Bandwidth sharing: objectives and algorithms. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1395–1403. IEEE.

Maternia, M. and El Ayoubi, S. E. (2016). *5G-PPP use cases and performance evaluation models*. 5G-PPP.

Médard, M. (2008). *Delay Models and Queueing*. EECS, MIT.

Megiddo, N. and Modha, D. S. (2003). ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, FAST '03, pages 115–130, Berkeley, CA, USA. USENIX Association.

Megyesi, P., Krämer, Z., and Molnár, S. (2016). How quick is QUIC?

Melazzi, N. B., Detti, A., and Pomposini, M. (2013). Scalability measurements in an information-centric network. In *Measurement Methodology and Tools*, pages 81–106. Springer.

Merkle, R. C. (1987). A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer.

Ming, Z., Xu, M., and Wang, D. (2012). Age-based cooperative caching in information-centric networks. In *Proc. of IEEE INFOCOM NOMEN Workshop*.

Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., and Mahanti, A. (2011). Characterizing web-based video sharing workloads. *ACM Trans. Web*, 5(2):8:1–8:27.

Mo, J. and Walrand, J. (2000). Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking (ToN)*, 8(5):556–567.

Morrison, D. R. (1968). PATRICIA- practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM (JACM)*, 15(4):514–534.

Naor, M. and Pinkas, B. (1999). Oblivious transfer with adaptive queries. In *Annual International Cryptology Conference*, pages 573–590. Springer.

Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., and Steenkiste, P. (2014). The cost of the s in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140. ACM.

Naylor, D., Schomp, K., Varvello, M., Leontiadis, I., Blackburn, J., López, D. R., Papagiannaki, K., Rodriguez Rodriguez, P., and Steenkiste, P. (2015). Multi-context tls (mctls): Enabling secure in-network functionality in tls. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 199–212. ACM.

Neglia, G., Carra, D., Feng, M., Janardhan, V., Michiardi, P., and Tsigkari, D. (2016). *Access-time aware cache algorithms*. PhD thesis, Inria Sophia Antipolis.

Nguyen, D., Sugiyama, K., and Tagami, A. (2015). Congestion price for cache management in information-centric networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 287–292. IEEE.

Olmos, F., Graham, C., and Simonian, A. (2015). Cache miss estimation for non-stationary request processes. *arXiv preprint arXiv:1511.07392*.

Olmos, F., Kauffmann, B., Simonian, A., and Carlinet, Y. (2014). Catalog dynamics: Impact of content publishing and perishing on the performance of a lru cache. In *Teletraffic Congress (ITC), 2014 26th International*, pages 1–9. IEEE.

Papadimitriou, D., Welzl, M., Scharf, M., and Briscoe, B. (2011). Open research issues in internet congestion control. Technical report.

Papadopoulos, F., Krioukov, D., Boguna, M., and Vahdat, A. (2010). Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Paschos, G., Bastug, E., Land, I., Caire, G., and Debbah, M. (2016). Wireless caching: technical misconceptions and business barriers. *IEEE Communications Magazine*, 54(8):16–22.

Perino, D., Varvello, M., Linguaglossa, L., Laufer, R., and Boislaigue, R. (2014). Caesar: a content router for high-speed forwarding on content names. In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*, pages 137–148. ACM.

Psaras, I., Chai, W. K., and Pavlou, G. (2012). Probabilistic in-network caching for information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 55–60, New York, NY, USA. ACM.

Qadir, J., Ali, A., Yau, K.-L. A., Sathiaseelan, A., and Crowcroft, J. (2015). Exploiting the power of multiplicity: a holistic survey of network-layer multipath. *IEEE Communications Surveys & Tutorials*, 17(4):2176–2213.

Radhakrishnan, S., Cheng, Y., Chu, J., Jain, A., and Raghavan, B. (2011). Tcp fast open. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 21. ACM.

Radunović, B. and Boudec, J.-Y. L. (2007). A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking (TON)*, 15(5):1073–1083.

Raiciu, C., Niculescu, D., Bagnulo, M., and Handley, M. J. (2011). Opportunistic mobility with multipath tcp. In *Proceedings of the sixth international workshop on MobiArch*, pages 7–12. ACM.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278.

Roberts, J. and Sbihi, N. (2013). Exploring the memory-bandwidth tradeoff in an information-centric network. *CoRR*, abs/1309.5220.

Rossini, G. and Rossi, D. (2014). Coupling caching and forwarding: Benefits, analysis, and implementation. In *Proceedings of the 1st international conference on Information-centric networking*, pages 127–136. ACM.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

Saino, L., Psaras, I., and Pavlou, G. (2013). Hash-routing schemes for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 27–32. ACM.

Schenk, O., Bollhöfer, M., and Römer, R. A. (2008). On large-scale diagonalization techniques for the anderson model of localization. *SIAM Rev.*, 50(1):91–112.

Schenk, O., Wächter, A., and Hagemann, M. (2007). Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications*, 36(2-3):321–341.

Sengupta, A., Tandon, R., and Simeone, O. (2016). Cloud RAN and edge caching: Fundamental performance trade-offs,. In *Proc. IEEE International workshop on Signal Processing advances in Wireless Communications (SPAWC)*.

Shah, V. and de Veciana, G. (2014). Performance evaluation and asymptotics for content delivery networks. In *INFOCOM, 2014 Proceedings IEEE*, pages 2607–2615. IEEE.

Shah, V. and de Veciana, G. (2015). Impact of fairness and heterogeneity on delays in large-scale content delivery networks. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 375–387. ACM.

Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. (2015). Blindbox: Deep packet inspection over encrypted traffic. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 213–226. ACM.

Shi, Y., Zhang, J., Letaief, K. B., Bai, B., and Chen, W. (2015). Large-scale convex optimization for ultra-dense cloud-ran. *IEEE Wireless Communications*, 22(3):84–91.

Shreedhar, M. and Varghese, G. (1995). Efficient fair queueing using deficit round robin. *ACM SIGCOMM Computer Communication Review*, 25(4):231–242.

Singla, A., Chandrasekaran, B., Godfrey, P., and Maggs, B. (2014). The internet at the speed of light. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, page 1. ACM.

Singla, A., Chandrasekaran, B., Godfrey, P., and Maggs, B. (2015). Towards a speed of light internet. *arXiv preprint arXiv:1505.03449*.

So, W., Narayanan, A., and Oran, D. (2013). Named data networking on a router: fast and DoS-resistant forwarding with hash tables. In *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*, pages 215–226. IEEE Press.

Song, T., Yuan, H., Crowley, P., and Zhang, B. (2015). Scalable name-based packet forwarding: From millions to billions. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 19–28. ACM.

Sourlas, V., Flegkas, P., and Tassiulas, L. (2014). A novel cache aware routing scheme for information-centric networks. *Elsevier Science, Computer Networks*, 59:44–61.

Starobinski, D. and Tse, D. (2001). Probabilistic methods for web caching. *Perform. Eval.*, 46(2-3):125–137.

Tassiulas, L. and Ephremides, A. (1992). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *Automatic Control, IEEE Transactions on*, 37(12):1936–1948.

Tortelli, M., Cianci, I., Grieco, L., Boggia, G., and Camarda, P. (2011). A fairness analysis of content centric networks. In *Network of the Future (NOF), 2011 International Conference on the*, pages 117–121. IEEE.

Traverso, S., Ahmed, M., Garetto, M., Giaccone, P., Leonardi, E., and Niccolini, S. (2013). Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12.

Tucker, H. G. (2013). *A graduate course in probability*. Courier Corporation.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57.

Wang, Y., He, K., Dai, H., Meng, W., Jiang, J., Liu, B., and Chen, Y. (2012). Scalable name lookup in ndn using effective name component encoding. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 688–697. IEEE.

Wang, Y., Li, Z., Tyson, G., Uhlig, S., and Xie, G. (2013a). Optimal cache allocation for content-centric networking. In *Proc. of IEEE ICNP*.

Wang, Y., Rozhnova, N., Narayanan, A., Oran, D., and Rhee, I. (2013b). An improved hop-by-hop interest shaper for congestion control in named data networking. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 55–60. ACM.

Yao, A. C.-C. (1986). How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE.

Yeh, E., Ho, T., Cui, Y., Burd, M., Liu, R., and Leong, D. (2014). VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks. In *Proc. of ACM ICN*, pages 117–126.

Yu, Y.-T., Bronzino, F., Fan, R., Westphal, C., and Gerla, M. (2015). Congestion-aware edge caching for adaptive video streaming in information-centric networks. In *Proc. of IEEE CCNC Conference*.

Yuan, H. and Crowley, P. (2015). Reliably scalable name prefix lookup. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems*, pages 111–121. IEEE Computer Society.

Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. (2014). Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73.

Zhang, L. and al. (2010.). Named Data Networking (NDN) Project. http://named-data.net/ndn-proj.pdf.

Appendices

Appendix A

ZIMPL Mathematical Programs

A.1 ZIMPL code for section 3.4.2

```
#########################  MAIN PROBLEM  ############################

set K              := {1 .. 1000};           # Content ranks
set R              := {1 .. 3};              # Routes
set A              := {<k,r> in K * R};      # Ranks x routes
param alpha        := 1;                     # Zipf skewness.
param lambda := 10;# Demand rate. Test values: 0.5,1,2,5,7,10,20
param c            := sum <k> in K: exp(-ln(k) * alpha);
param q[<k> in K]  := exp(-ln(k) * alpha)/c; # Zipf popularity;
param mu[R]        := <1> 5, <2> 3, <3> 2;   # Bottleneck link rate
param x[R]         := <1> 10, <2> 10, <3> 10;# Cache budget
param epsilon      := 0.001;                 # Lower bound
defset contents(r) := {<k,r> in A};          # Route r's contents
defset routes(k)   := {<k,r> in A};          # Content k's routes
                                             # To be optimized:
var w[A] real >= 0 <= 1;                     # Route weights
var h[A] real >= 0 <= 1;                     # Content hit ratios.
   Test
                                             # max values: 0.8, 0.5
var rho[R] real >= epsilon <= 1;             # Bottleneck link loads
var lnN[R] real >= ln(epsilon);              # Log of mean number
                                             # of active transfers
minimize latency: sum<r> in R: exp(lnN[r]);  # Objective
                                             # Constraints:
subto c1:                                    # Cache budget
forall <r> in R: sum<k,r> in contents(r): h[k,r] == x[r];
subto c2:                                    # Mandatory delivery
forall <k> in K: sum<k,r> in routes(k): w[k,r] == 1;
subto c3:                                    # Link load formula
forall <r> in R do
rho[r] == sum<k,r> in contents(r): q[k] * w[k,r] * (1 - h[k,r]) *
   lambda / mu[r];
subto c4:                                    # Log of M/G/1-PS E[N]
forall <r> in R: lnN[r] == ln(rho[r]) - ln(1 - rho[r]);
```

A.2 ZIMPL code for section 3.4.3.0

```
#########################  LRU SUB-PROBLEM  ##########################

set K               := {1 .. 14};               # Content ranks
set R               := {1 .. 2};                # Routes
set A               := {<k,r> in K * R};        # Ranks x routes
param alpha         := 1;                       # Zipf skewness.
param c             := sum <k> in K: exp(-ln(k) * alpha);
param q[<k> in K]   := exp(-ln(k) * alpha)/c;   # Zipf popularity;
param x[R]          := <1> 5, <2> 6;  # Cache size
#param n[R]          := <1> 1, <2> 1; # Flow bundle's factor
defset contents(r) := {<k,r> in A};  # Route r's contents
defset routes(k)   := {<k,r> in A};  # Content k's routes
                                     # To be optimized:
var i[A] binary;                     # Indicator of content
   presence
var h[A] real;                       # Content hit ratio
var H[A] real;                       # Content hit ratio x
   popularity
var T[R] real;                       # Cache Characteristic Time
maximize hit: sum<k,r> in A: H[k,r];  # Objective
                                     # Constraints:
subto c1:                            # Unicity
forall <k> in K: sum<k,r> in routes(k): i[k,r] <= 1;

subto c2:                            # Flow bundle size is limited
forall <r> in R: sum<k,r> in contents(r): i[k,r] == x[r] + 1;
#forall <r> in R: sum<k,r> in contents(r): i[k,r] == n[r] * x[r];
subto c3:                            # Weighting hit ratio
forall <k,r> in A: H[k,r] == q[k] * h[k,r];

subto c4:                            # Compute hit ratio
forall <k,r> in A: h[k,r] == i[k,r] * (1 - exp(-q[k] * T[r]));

subto c5:                            # Compute Characteristic Time
forall <r> in R: sum<k,r> in contents(r): h[k,r] == x[r];
```

Appendix B

Résumé étendu

Les Réseaux Centrés sur l'Information (ICN) [Jacobson et al., 2009a] constituent un paradigme de communication émergent visant à pallier la croissance effrénée du traffic à travers l'Internet d'aujourd'hui. Comme caractéristique principale, *(i)* ICN remplace au sein de la couche réseau le "où" par le "quoi", en acheminant les paquets sur la base de noms de contenus. *(ii)* ICN utilise extensivement la fonction de mise en cache au niveau de la couche réseau. Tandis que de moins en moins de trafic IP demeure cachable du fait du chiffrement de bout-en-bout des canaux de communication, [Naylor et al., 2014], ICN, à travers, par example, son implémentation NDN [Zhang and al., 2010] offre une alternative prometteuse en ignorant l'identité des utilisateurs (pas d'adresse de la machine hôte) et en authentifiant les fournisseurs de contenus à partir de lq signature numérique contenue dans chaque paquet de données. De ce fait, chaque contenu peut être conservé en cache sous pour cela sacrifier toute confidentialité ou toute capacité à vérifier la provenance desdits contenus. Ceci trouve son importance dans le fait que la mise en place de mémoires-caches à travers les liens acheminant des données entre noeuds de communication s'est avérée être un moyen reconnu d'amélioration des performances du réseau en retenant à la périphérie de celui-ci un certain nombre de contenus, principalement les plus populaires. Ceci contribue à réduire la charge du réseau et la latence des téléchargements.

Une telle minimisation de la latence est un pilier de l'architecture des réseaux 5G, qui par ailleurs draine une quantité considérable d'efforts de recherche.

Cette thèse est une contribution à cet effort. Nous y abordons divers aspects de NDN: *(i)* Nous proposons, analysons et implémentons deux mécanismes de mise en cache sensibles à la latence LAC et LAC+ qui réduisent de manière significative le temps de téléchargement des contenus;

(ii) Nous proposons, analysons et implémentons un algorithme conjoint de mise en cache et de diffusion sensible à la latence nommé FOCAL, visant à l'amélioration des performances des algorithmes de gestion de caches;

(iii) Nous analysons l'équité dans les réseaux ICN, étant donné l'ubiquité des caches dans de tels réseaux;

(iv) Finalement, nous proposons une approche de diffusion de paquets basée sur des réseaux de neurones dans l'optique d'aider au passage à l'échelle du FIB (Forward Information Base) NDN au sein d'un espace de nommage Internet illimité.

Dans ce résumé, nous procédons de la manière suivante. Nous y présentons ICN du point de vue de la gestion de caches et de la diffusion de paquets. Ensuite nous énonçons

les problèmes que cette thèse traite conjointement à un condensé des contributions scientifiques qui y répondent.

B.1 Information-Centric Networking

Les réseaux-centrés sur l'Information visent à repenser l'Internet afin de le mettre en adéquation avec ses défis actuels et futurs. Introduit par [Jacobson et al., 2009b], ce nouveau paradigme propose le passage au niveau de la couche réseau, d'un modèle de communication orienté machines à une schéma centrés sur des contenus nommés.

Ce travail séminal reconnait qu'il y a plusieurs décennies, les réseaux visaient à partager des ressources aussi rares que chères telles que des lecteurs rapides de bandes ou des supercalculateurs. A cette époque, insérer des adresses-machine source et destination dans les paquets afin d'indiquer au réseau vers *où* les transférer faisait sens. Le but des réseaux a significativement changé depuis, ceci dû à la démocratisation de la puissance de calcul et des larges capacités de stockage. De nos jours l'Internet sert majoritairement à transporter des contenus/informations et pour ce faire, nécessite qu'on lui indique *quoi* fournir.

Alors que le World Wide Web, les réseaux pair- à-pair (P2P) et les réseaux de diffusion de contenus (CDNs) ont introduit la diffusion de données nommées à travers l'Internet, l'originalité d'ICN réside dans son ubiquité. ICN peut à la fois etre exécuté comme un réseau overlay comme ses prédécesseurs ou comme un protocole de couche 3 capable de remplacer IP partout où IP règne aujourd'hui.

La communauté scientifique tablant sur ICN est persuadé que de réinventer l'Internet de la sorte devrait intrinsèquement en améliorer la sécurité , le passage à l'échelle et la robustesse.

Parmi les architectures ayant adopté cette approche, on compte Data-Oriented Network Architecture (DONA) [Koponen et al., 2007] et Network of Information (NetInf) [Ahlgren and al., 2008]. Elles diffèrent des deux architectures les plus adoptées, CCN and NDN, par le fait qu'elles reposent sur un modèle de type publier-souscrire dans un arbre de Resolution Handlers de confiance (DONA) ou dans une table de hachage distribuée (NetInf).

B.1.1 Content-Centric / Named-Data Networking

Content-Centric Networking (CCN) [Jacobson et al., 2009b] et Named-Data Networking (NDN) [Zhang and al., 2010; Zhang et al., 2014] sont deux architectures majeures et très similaires de réseaux centrés sur l'information. NDN commença en tant que branche du code de CCN, CCNx, avant d'être entièrement redéveloppé. En revanche, d'un point de vue architectural, CCN et NDN n'ont pas divergé. Dans les deux architectures, chaque morceau de contenus est identifié par un nom unique, requis via des paquets dits *Intérêts* et rapportés encapsulés dans des paquets de *Données*, à travers des (*inter*)faces. Parce que chaque paquet encapsule aussi son nom, il peut être persisté à chaque saut dans un *Content Store*, un cache, et livré du noeud local à tous ceux qui l'on requis.

B.1.2 Fonctionnement de NDN

Comme dépeint par Fig.1.1, à chaque noeud traversé, NDN/CCN conserve dans une structure de données nommée *Pending Interest Table* (PIT), une trace des faces d'où provient l'intérêt, l'instant de son passage et le nom du contenu souhaité.

Si un morceau de contenu est trouvé dans le Content Store, il s'agit d'un évènement nommé *hit*. La donnée est alors acheminée en utilisant les informations d'états contenus dans l'entrée du PIT correspondante.

Par contre, si le morceau de contenu n'est pas trouvé dans le Content Store, il s'agit d'un évènement *miss*. Dès lors, le plus long préfixe incluant le nom du contenu est recherché dans la *Forwarding Information Base* (FIB) du noeud. La FIB est remplie par un protocole de routage orienté nom et renvoie, pour chaque préfixe routable, les faces de sortie pour les morceaux de contenus manquants.

Ensuite, l'intérêt est transmis selon la stratégie de diffusion configurée qui peut-être par exemple, Meilleure route, diffusion large, équilibrage de charge (LB), équilibrage de charge (LB) avec transmission persistante, que nous analysons dans cette thèse.

Lorsque le paquet de données correspondant revient, il est pris en charge pas un algorithme de gestion de cache tel que LAC+ que nous proposons dans ce document. Finalement, ce paquet est renvoyé aux clients et l'entrée correspondante est supprimée de la PIT.

Du fait de leur importance, de leurs fondamentaux communs et opérations globalement identiques, *NDN et CCN sont désignés par NDN* dans ce document afin d'être bref.

B.1.3 Transmission dans NDN

Un routeur IP n'a de visibilité ni sur la fenêtre de contrôle de flux du receveur, ni sur la fenêtre de contrôle de congestion de l'émetteur. Il ne peut donc influer sur les sessions TCP en cours, qui sont conçues pour être de bout-en-bout.

La spécificité du plan de données NDN est qu'il n'est pas sans état. Les PIT enregistrent, saut après saut les intérêts non satisfaits et les faces d'où ils proviennent.

Chaque paquet de données suit le chemin inverse de celui de l'intérêt correspondant. Puisque un nonce identifie de manière unique les intérêts portant sur les mêmes morceaux de contenus, la PIT peut détecter et éliminer ceux d'entre eux qui bouclent.

Etant donné ces qualités architecturales, chaque noeud NDN met à disposition un ou plusieurs algorithmes de transmission au sein de ce qu'il convient de nommer sa couche stratégie.

Essentiellement deux types d'algorithmes opèrent au niveau de cette couche: les contrôleurs côté *récepteur*, souvent de type Incrément augmentation-retrait multiplicatif (AIMD); et les régulateurs d'intérêts *saut-par-saut* ou, lorsque de multiples interfaces de sortie existent pour un préfixe donné: équilibreurs de charge, diffuseurs ou sélecteurs du meilleur chemin.

Le but d'un contrôleur côté récepteur est d'utiliser toute la capacité disponible grâce à une fenêtre de congestion globale; Il peut être complété par de la gestion active de file afin d'anticiper toute pénurie de capacité et décrémenter la fenêtre de congestion en conséquence. Cette fenêtre de congestion sera décrémentée avec une probabilité proportionnelle au temps d'aller-retour lissé [Carofiglio et al., 2013c].

Grâce à une symétrie parfaite entre paquet Intérêt et paquet de Données, le contrôle de flux et de congestion sont réalisables à chaque saut par le biais de la régulation d'intérêts. La régulation d'intérêts saut-par-saut présente l'avantage d'une prise de décision rapide et répartie en présence de trafic irrégulier [Carofiglio et al., 2012; Wang et al., 2013b]. Il est par ailleurs intéressant de noter que la mise en tampon des intérêts prend bien moins de place que celle des paquets de données. En outre, le support natif du multi-chemin dans NDN autorise l'équilibrage de charge à travers plusieurs interfaces de sorties. De plus, il permet de choisir entre différents chemins lorsque surviennent congestions ou coupures de lien [Carofiglio et al., 2013c].

B.1.4 Caching in NDN

[Imbrenda et al., 2014] a démontré grâce à des captures de trafic réel que des caches de taille négligeable (100Mo), placés chez l'usager pouvait réduire la charge de 25% dans les réseaux d'accès fibre. Il a par ailleurs montré que placer des caches de 100Go en aval d'un lien de backhaul pouvait décharger de 35% ledit lien. Parfois remis en cause, l'usage de caches dans le réseaux, en travers des chemins, se voit ainsi conforté dans sa légitimité. Ce travail établit qu'il n'est aucunement nécessaire de déployer toute la mémoire cache en bordure du réseau, comme le suggérait pourtant [Fayazbakhsh et al., 2013; Garcia-Luna-Aceves et al., 2014] qui arguait l'insuffisante redondance dans le trafic émanant de chaque usager.

L'approche multi-niveau que constitue la mise en cache en travers des chemins combine l'exploitation de la redondance au sein du trafic généré par chaque usager, avec celle de plusieurs usagers et groupe d'usagers.

Du point de vue architectural, NDN correspond à ce que [Paschos et al., 2016] prévoit pour les réseaux sans fil de prochaine génération, préconisant la pénétration de CDN dans le réseau RAN (Radio Access Network), couvrant les périphériques mobiles et les stations de base, pour exploiter toute forme de redondance du trafic.

Par conséquent, alors que les routeurs IP sont incapables de servir des paquets en tampon à toute autre session, les paquets mis en cache dans NDN sont entièrement réutilisables car ils sont nommés de manière unique. Il est prévu que chaque n?ud NDN contiennent un cache. En cela, les caches font partie intégrante de l'architecture. Même si les caches sont censés être beaucoup plus grands que les tampons de paquets des routeurs d'aujourd'hui, ils restent néanmoins finis. Un algorithme de gestion de cache doit décider des objets à conserver et de ceux à expulser. Sont souvent évoqués : Premier arrivé- Premier servi (FIFO), expulse le moins utilisé (LFU) ou diverses améliorations à la politique LRU visant à expulser le moins récemment utilisé (p -LRU, LRU + LCD) [Laoutaris et al., 2004].

Les performances d'un réseau de données nommées (NDN) dépendent clairement de la politique de gestion de cache choisie. Nous décrivons ci-après quelques-unes.

L'algorithme Least Frequently Used

LFU consiste à expulser l'objet qui a enregistré le moins de téléchargements lors de la dernière fenêtre de temps. De cette façon, seuls les objets les plus populaires de cette période demeurent en cache. En supposant la distribution de popularité stationnaire, c'est-à-dire que la popularité du contenu ne change pas avec le temps, LFU est parmi les politiques les plus efficaces pour ce qui est de la réduction de la charge. Cependant, il souffre de deux faiblesses:

(i) L'insensibilité à l'état du réseau. LFU gère tous les contenus uniquement en fonction de leur popularité locale. Il ignore simplement le véritable but de la mise en cache, qui est l'amélioration de la qualité d'expérience de l'utilisateur (QoE).

(ii) Réaction tardive aux variations de popularité. LFU doit recueillir des statistiques pendant un délai relativement long s'il entend prendre en compte tous les contenus de popularité significative. Pour cette raison, il fonctionne parfaitement lorsqu'il existe une distribution de popularité de contenu stationnaire. Dans ce cas, la moyenne à long terme du ratio de téléchargement d'un contenu donné est la probabilité qu'un téléchargement quelconque le transmette. Cette dernière condition caractérise le modèle de référence indépendant (IRM) pour l'analyse de la performance du cache, nous utilisons dans nos travaux. Dans le cadre strict de l'hypothèse IRM, LFU est l'algorithme optimal [Martina et al., 2013]. Cependant, les modèles à bruit de tir (SNM) [Leonardi and Torrisi, 2015; Olmos et al., 2015] dans lesquels les objets sont publiés puis périssent offrent des alternatives plus réalistes mais moins calculables qui elles, ne favorisent pas LFU.

Nous proposons de remédier à ces deux faiblesses grâce à la politique Latence-Aware Caching+ (LAC+) que nous présentons dans la section suivante.

L'algorithme First-In-First-Out

Selon FIFO, le contenu le plus ancien dans le cache est le premier évacué. Requérir un objet et le trouver dans le cache (qualifié d'événement succès ou hit) ne change rien à son sort car il laisse le cache non modifié. Un tel fonctionnement n'exploite pas pleinement la popularité des contenus, car le cache n'a aucun moyen de marquer des objets sollicités en modifiant leur état en conséquence. Cependant, il convient de noter que le taux de

hits d'un contenu dans un cache FIFO reflète toujours la popularité de ce contenu. Les objets populaires ont un taux d'accès bien plus élevé en raison du ralentissement de la dynamique d'éviction due à des hits plus fréquents et à une insertion plus fréquente après les événements de type miss (qui sont le contraire des hits).

Un cache FIFO est souvent implémenté par une file d'attente: chaque nouvelle insertion décale les objets déjà présents d'une place en direction de la sortie du cache. Sa simplicité en fait la politique de gestion de cache par défaut dans NDN. De plus, du point de vue ingénierie, la recherche de morceaux de contenus possède une complexité moindre dans les files d'attente orientées en morceaux; ceci grâce à une moindre fragmentation du cache. En effet, par delà les retransmissions de paquets, les morceaux d'un même contenu restent contiguë dans le cache.

L'algorithme Least Recently Used

La politique LRU expulse l'objet dont le dernier accès est le plus ancien. Une implémentation simple de cette politique consiste à déplacer un objet trouvé dans un cache FIFO, à l'avant de ce dernier. Cette implémentation est nommée algorithme *Move-To-Front* (MTF) [Starobinski and Tse, 2001]. Le mécanisme MTF possède un meilleur taux de hits que FIFO grâce à un échantillonnage de popularité plus efficace. Les objets les plus populaires ont tendance à rester plus longtemps dans les caches LRU en raison d'une sorte de ré-initialisation de leur temps-de-vie (TTL), propriété identifiée par [Choungmo Fofack et al., 2012]. L'adaptation sans heurts de LRU aux changements de popularité et sa simplicité en font une alternative beaucoup plus réaliste à LFU, expliquant en quelque sorte pourquoi il a été fréquemment étudié [Jelenković and Kang, 2008; Fricker et al., 2012; Leonardi and Torrisi, 2015]. Le principal inconvénient de LRU reste son plus bas taux de hits, asymptotiquement $e^{\gamma} \approx 1,78$ fois loin de LFU, où γ est la constante d'Euler. [Jelenković, 1999].

L'algorithme LRU avec insertion aléatoirement d'une copie (p-LRU)

Considérons un cache LRU et choisissons, au lieu de toujours insérer chaque objet nouvellement téléchargé après un événement Miss, de le faire avec une probabilité fixe p. Cette description est celle de la politique p-LRU. Aussi surprenant que cela puisse paraître, ce faisant, volontairement en appliquant cette insertion stochastique à l'aide d'un généra-

teur de nombres pseudo-aléatoires (PRNG), ou en raison d'une corruption involontaire de données [Bianchi et al., 2013], augmente le taux de hits du cache. En fait, plus p se rapproche de zéro, plus proche p-LRU est d'un système LFU [Martina et al., 2013]. C'est un résultat important qui a présidé à la l'élaboration de nos algorithmes LAC et LAC+.

Toutefois, lorsque p tend vers zéro, de moins en moins d'objets sont stockés. Raison pour laquelle, p-LRU, $p \to 0$ ne peut fonctionner que sous l'hypothèse d'une distribution de popularité stationnaire.

B.2 Contributions

Ce travail de thèse comporte 4 volets que nous introduisons ci-après:

1. LAC/LAC+: Algorithmes de gestion de cache sensibles à la latence
2. FOCAL: Transmission et gestion de cache conjointes et sensibles à la latence
3. Equité dans les réseaux centrés sur l'information
4. AFFORD: Ask For Directions, routage basé sur l'apprentissage automatique

B.3 LAC/LAC+: Algorithmes de gestion de cache sensibles à la latence

Nous avons conçu une famille de politiques aléatoires de gestion de cache exploitant la latence des téléchargements lors de la mise en cache. Cette latence est le temps écoulé entre l'envoi d'une requête et la réception de sa réponse.

Les mécanismes de gestion de cache LAC et LAC+ [Carofiglio et al., 2015a,c] fonctionnent comme il suit:

Chaque fois qu'un objet arrive du réseau, il est stocké dans le cache avec une faible probabilité globale. Cette probabilité augmente considérablement lorsque le contenu présente un temps de téléchargement (ou latence) exceptionnellement long. En tant que tel, il s'agit de greffons au-dessus d'un cache LRU qui l'alimentent à une cadence régulée. De cette façon, ces algorithmes de mise en cache *favorise implicitement les objets populaires et laborieux à télécharger*.

Le compromis sous-jacent à un tel mécanisme de mise en cache est de conjuguer une taille de cache nécessairement limitée et la minimisation du temps de téléchargement.

Puisque les caches visent intrinsèquement à réduire la distance dans le réseau et combattre la congestion, ils doivent prendre en compte la popularité des contenus et autres facteurs de délais.

Une probabilité d'insertion globalement faible échantillonne des objets populaires tandis que la latence de récupération des données est une métrique simple, localement mesurable et cohérente révélant distance parcourue par lesdites données ou congestion.

Le champs d'application de nos politiques de gestion de cache sensibles à la latence est large: Réseaux centrés sur l'information, centres de données, réseaux de distribution de contenus (CDN), l'optimisation de serveurs multiprocesseur.

Pour finir, nous avons écrit deux implémentations C++ de LAC et LAC+, une pour le simulateur CCN niveau paquets (CCNPL-Sim [3]) et une autre pour le simulateur NDN basé sur NS-3 (NDNSim) , qui par ailleurs partage du code avec le daemon de transmission NDN (NFD)[4].

Le chapitre 2 porte sur l'analyse et l'évaluation des algorithmes LAC et LAC+.

B.3.1 Description

Dans ce travail, nous analysons deux algorithmes distribués, LAC [Carofiglio et al., 2015c] et LAC+ [Carofiglio et al., 2015b], qui visent à minimiser le délai de téléchargement moyen dans les réseaux centrés sur l'information, et ce sans coordination entre les caches et sans de coûts de signalisation. Notez qu'une mesure fine de la latence est disponible en ICN car les requêtes transmises à travers une interface trouvent réponse à travers cette même interface. A l'échelle du réseau, ceci permet un routage symétrique et la mesure de la latence induite par le réseau en amont.

Les deux algorithmes fonctionnent de la manière suivante: *Lorsqu'un client sollicite à l'instant t un objet de rang k, $k \in \mathcal{K}$, si cet objet se trouve dans un cache le long du chemin et il est retourné au demandeur, ou ce cache le téléchargera, le conservera avec une probabilité $p_K(t)$ ou pas, avec une probabilité de $1 - p_K(t)$ et finalement enverra cet objet au demandeur.* Nous appelons probabilité de décision p_K.

[3] http://systemx.enst.fr/ccnpl-sim.html
[4] http://named-data.net/doc/NFD/current/

Pour LAC, la probabilité d'insertion d'un objet de rang k dans le cache à l'instant t est:

$$p_k(t) \equiv \min\left(\epsilon\frac{T_k(t)^\beta}{\bar{T}(t)^\gamma}, 1\right). \tag{B.1}$$

Pour LAC+, la probabilité de décision $p_k^+(t)$ combine deux termes:

$$p_k^+(t) \equiv p_k(t) + (1 - p_k(t))\Theta_k(t) \tag{B.2}$$

où $T_k(t)$ est la latence du contenu de rang k monitorée jusqu'à l'instant t et $\bar{T}_k(t)$, $\bar{T}(t)$ sont respectivement la moyenne pour le contenu de rang k au cours du temps et pour tous les contenus mis en cache, calculée jusqu'à l'instant t. Les moyennes sont glissantes (filtre EMA). Nous avons obtenu des résultats satisfaisants en configurant dans le filtre un poids pour les valeurs passées de 0.9

ϵ est un très petit réel. β et γ sont des paramètres d'intensité utilisés par LAC afin de creuser l'écart en probabilité entre téléchargements de grande et de petite latence. Les objets présentant une petite grande latence sont téléchargés plus tôt. Les objets à latence négligeable obtiennent une très petite probabilité de décision mais seront éventuellement conservés s'ils sont populaires.

Pour LAC+, puisqu'il possède une fonction de capture des aberrations (outliers) $\Theta_k(\cdot)$, β and γ sont habituellement fixés à 1.

Soient μ_t et σ_t la moyenne et l'écart-type de tous les $\bar{T}_i(t), \forall i \in \mathcal{K}$, à un noeud donné. Nous définissons le z^{th} quantile comme il suit:

$$Q_z(t) = \mu_t + z\sigma_t. \tag{B.3}$$

Ceci permet d'expliciter le second terme de p_k^+. $\Theta_k(t)$ est un indicateur quantitatif à l'instant t du fait qu'un objet de rang k object est une aberration du point de vue des latences moyennes:

$$\Theta_k(t) \equiv \max\left(\frac{\bar{T}_k(t) - Q_z(t)}{\sup_{i\in\mathcal{K}}\{\bar{T}_i(t)\} - Q_z(t)}, 0\right). \tag{B.4}$$

$(z = 1)$ semble être une valeur raisonnable nous ayant donné satisfaction.

Pour résumer, LAC+ capture les objets les plus populaires échantillonnés par le premier terme de $p_k^+(t)$ ou les aberrations grâce au second terme de $p_k^+(t)$.

B.3.2 Analyse

La dynamique du système réseau est complexe à capturer en un modèle simple en raison du couplage étroit entre la performance de téléchargement et fonction de mise en cache: la première est clairement affectée par les conditions du réseau, tandis que la charge réseau découle des performances des caches et vice versa. C'est pourquoi nous nous sommes concentrés sur le cas d'un cache unique en développant analytiquement certaines limites de performance exprimées en termes de taux de miss. Brièvement nous démontrons que les approches asymétrique comme LAC et LAC+ surpassent les mécanismes précédemment connus, typiquement Starobinski-Tse-Jelenković-Radovanović's (STJR). LAC et LAC+ sont du genre $asym$ -LRU par opposition aux systèmes où les opérations d'insertion / remplacement sont soumises symétriquement à la même probabilité (sym -LRU). LCP est un cas particulier de mécanisme asymétrique où l'insertion dans le cache est déterminée par une probabilité constante p. Se reporter au tableau B.1 pour la notation utilisée tout au long de cette section. Les variables pourraient être plus tard libellée avec l'algorithme courant en indice.

t	Instant de réception d'un objet.
x	Capacité du cache local en nombre d'objets
τ_x	Seuil temporel caractéristique pour remplir un cache de capacité x.
λ	Taux de requête total.
λ_k	Taux de requête pour l'objet de rang k, $k \in \mathcal{K}$.
q_k	Popularité de l'objet de rang k. $q_k = \lambda_k/\lambda$.
$\varphi_{k,\tau}$	Probabilité de réception d'au moins une requête pour l'objet de rang k pendant τ secondes.
M_k	Taux de miss asymptotique de l'objet de rang k.
$\{\mathrm{VRTT}_{k,t}\}_{t\geq 0}$	Processus stochastique modélisant la latence de téléchargement de l'objet de rang k.
$\{p_{k,t}\}_{t\geq 0}$	Processus de la probabilité de mise en cache de l'objet de rang k.
p	Probabilité de mise en cache aléatoire telle que $p \overset{d}{=} p_{k,t}, \forall k, t$ sous hypothèse i.i.d.
$\{\pi_{k,t}\}_{t\geq 0}$	Processus de la probabilité de miss de l'objet de rang k.
$\{\mathcal{M}_{k,t}\}_{t\geq 0}$	Processus de comptage des miss de l'objet de rang k. Il est censé augmenter tous les $1/m_k$ cycles avec $m_k = \mathbb{E}[\pi_{k,t}]$.

Table B.1 – Notation.

Hypothèses

Considérons un petit nombre d'hypothèses permettant, au demeurant, d'obtenir un modèle simple et résoluble.

- *Distribution de popularité Zipf:* Nous supposons que la popularité d'un objet suit une loi de Zipf généralisée. Ainsi $\forall k \in \mathcal{K}, \quad q_k = ck^{-\alpha}$ avec $1/c = \sum_{i \in \mathcal{K}} i^{-\alpha}$ and le paramètre d'asymétrie $\alpha > 0$. Cette hypothèse est largement répandue dans la littérature [Breslau et al., 1999; Mitra et al., 2011].

- *Requêtes poissoniennes :* Nous supposons que les clients demandent des objets selon un processus de Poisson d'intensité $\lambda > 0$, de manière similaire à [Carofiglio et al., 2013b; Badov et al., 2014].

- *Modèle de Référence Indépendant* La corrélation temporelle entre les demandes d'objet est négligé ici comme dans [Starobinski and Tse, 2001] et [Fricker et al., 2012]. Il est néanmoins prévu pour les extensions futures de ce travail.

- *LRU:* Nous nous concentrons sur la politique de remplacement LRU car largement adopté et dont la mise en ?uvre la plus courante consiste à déplacer l'objet le plus récemment servi à l'avant d'une liste. Cela permet d'étudier l'algorithme Move-To-Front en tant que LRU. [Jelenković and Radovanović, 2004].

- *Same object size:* Par souci de simplicité, nous supposons que tous les objets récupérés ont la même taille comme dans [Che et al., 2006]. Le modèle pourra être amélioré plus tard et incorporer des objets de taille variable.

- VRTT$_{k,t}, \forall t \geq 0$ sont strictement positifs, indépendants et identiquement distribués (i.i.d.).

- $p_{k,t} \in]0, 1], \forall k, t \geq 0$ sont aussi i.i.d. Par conséquent $\exists p \overset{d}{=} p_{k,t}$ et $\pi_{k,t}, \forall t$ sont aussi i.i.d..

- L'approximation par temps caractéristique (dite de "Che") [Che et al., 2006] comme étendue par [Fricker et al., 2012] est un outil essentiel de ce travail. lI stipule que dans les caches LRU, le temps avant expulsion de tout objet est correctement approximé par une constante unique τ_x.

Taux de miss

Soit $\pi_{k,t}$ la probabilité de miss de l'objet de rang k à l'instant t et $\varphi_{k,\tau}$ la probabilité de recevoir au moins une requête concernant un objet de rang k pendant τ secondes.

Proposition B.1. *Si nous nous restreignons à un ensemble dénombrable de probabilités de mise en cache, et en supposant que l'approximation de Che prévaut, le ratio de miss M_k^{asym} d'algorithmes asymétriques tels que LAC et LAC+, pour l'objet de rang k est:*

$$M_k^{asym} = \sum_u \mathbb{P}[p = u] \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1-u)}, \tag{B.5}$$

$$\tau_x \text{ being the root of } \sum_{k \in \mathcal{K}} (1 - M_k^{asym}) = x. \tag{B.6}$$

Proof. Voir 2.4.2 □

Notons que $\varphi_{k,\tau_x} \equiv 1 - e^{-\lambda_k \tau_x}$ sous arrivée poissonienne d'objets.

Prendre en compte toutes les valeurs de p dans Eq.(B.6) n'est pas aisément résoluble numériquement. La proposition suivante montre que toutes les valeurs de p peuvent être efficacement remplacées par leur espérance.

Proposition B.2. *En supposant des arrivées de requêtes poissoniennes, le ratio de miss d'un cache M_k^{asym} est correctement approximé en utilisant l'espérance $\mathbb{E}[p]$ de la probabilité de mise en cache p quand $\mathbb{E}[p]$ tend vers zéro ou lorsque la popularité de l'objet tend vers zéro ou encore si le cache est très large:*

$$M_k^{asym} \approx \frac{1 - \varphi_{k,\tau_x}}{1 - \varphi_{k,\tau_x}(1 - \mathbb{E}[p])}. \tag{B.7}$$

Ce résultat est important car il établit des conditions réalisables pour l'équivalence asymptotique entre l'utilisation d'une probabilité de mise en cache variable p et l'usage de son espérance $\overline{p} = \mathbb{E}[p]$.

Cependant, l'inconvénient opérationnel d'un $\overline{p}$ constant et petit est qu'il remplace considérablement le moment où les objets populaires sont stockés dans le cache pour la première fois. LCP, par exemple, en souffre parce que le moment d'insertion dans le cache est prévu à $\frac{1}{\lambda_k \overline{p}}$. Par conséquent, Le temps moyen de téléchargement d'un objet converge lentement. LAC+ apporte une solution pour varier de manière adéquate p afin de mettre en cache les objets les plus coûteux au plus tôt.

Proof. Voir 2.2 □

Borne inférieure

Fournir une approximation sous forme close du taux de miss de $asym$-LRU et son temps caractéristique τ_x^{asym} est difficile. Au lieu de cela, nous démontrons sa supériorité par rapport au mécanisme sym-LRU analytiquement tractable. Moyennant une certaine perte de généralité, α, l'exposant de la loi Zipf, également appelé son paramètre d'asymétrie, est supposé strictement supérieur à 1.

Dans le mécanisme symétrique sym-LRU, les règles MTF sont conditionnées par la même probabilité dans les deux cas hit et miss. En revanche, dans $asym$-LRU la décision MTF est conditionnée par une probabilité uniquement en cas de miss.

Proposition B.3. *Supposons que les $VRTT_{k,t}, \forall k, t$ sont i.i.d., un large catalogue et un large cache, la probabilité de miss stationnaire des algorithmes LRU symétriques, pour tout objet de rang k, est approximée par:*

$$M_k^{sym} = \exp\left\{-\frac{x^\alpha}{k^\alpha \Gamma(1-\frac{1}{\alpha})^\alpha}\right\}, \tag{B.8}$$

où $\Gamma(\cdot)$ est la fonction Gamma.

Proof. Voir 2.4 □

l'expression en forme close de la Proposition 2.3 est intrinsèquement la même que celle de LRU dans [Carofiglio et al., 2013b]. Cette observation induit le corolaire suivant.

Corollary B.3.1. *En supposant que les $VRTT_{k,t}, \forall k, t$ sont i.i.d.,*

$$M_k^{sym} = M_k^{LRU}$$

i.e., *sym-LRU se comporte en régime stationnaire comme LRU.*

Il en découle que $asym$-LRU surpasse sym-LRU grâce à sa convergence vers la politique de gestion de cache Least Frequently Used [Martina et al., 2013]. On en déduit la Proposition 2.4, laquelle procède de la notion de conservation ϵ-permanent , a notion que nous introduisons ci-dessous.

Definition B.1. *Un objet est dit ϵ-permanemment conservé si son taux de miss est inférieur à un petit réel noté ϵ.*

Dans ce contexte, notons $\eta_{mechanism}$ le nombre d'objets populaires ϵ- permanemment conservés grâce à un mécanisme de mise en cache,

Proposition B.4. *Lorsque l'espérance de la probabilité de mise en cache tend vers zéro, asym-LRU héberge ϵ- permanemment un nombre plus important d'objets populaires que sym-LRU* i.e.,

$$\eta_{asym} \geq \eta_{sym}.$$

Proof. Voir 2.4. □

Notons $LAasym$ l'algorithme LRU équipé d'un mécanisme stochastique et asymétrique de mise en cache sensible à la latence (LAC et LAC+) et notons $LAsym$ l'algorithme LRU modifié pour le MTF stochastique et symétrique sensible à la latence (STJR).

Corollary B.4.1. *Lorsque l'espérance de la probabilité de mise en cache tend vers zéro,*

$$\exists \kappa \geq 1 : \eta_{LAasym} \geq \kappa \eta_{LAsym}. \tag{B.9}$$

Cela signifie généralement que la performance des caches LRU équipés de LAC ou de LAC+ peut dépasser d'un facteur κ donné celle de sym-LRU, donc de LRU étudiée analytiquement et largement dans les travaux précédents [Carofiglio et al., 2013b]. De nombreuses simulations soutiennent ces résultats mathématiques, où souvent $\kappa > 2$ et d'où découle une réduction drastique de la durée de téléchargement de contenus.

B.4 FOCAL: Transmission and gestion de cache conjointes et sensibles à la latence

La gestion de cache doit être soutenue par une stratégie de transfert de paquets appropriée [Rossini and Rossi, 2014; Dehghan et al., 2015] car la performance du premier est pilotée par le processus d'arrivée de la demande, duquel le second est responsable. En même temps, le trafic à transmettre est le trafic miss du cache local. De cette observation, il apparaît clairement que *la gestion de cache et le transfert doivent être optimisés conjointement*.

C'est le but de FOCAL [Carofiglio et al., 2015b, 2016]. Il combine l'algorithme de gestion de caches *LAC+*, alimentées par une nouvelle stratégie de transfert *LB-Perf* qui dirige constamment les objets les plus populaires à travers les mêmes interfaces, en s'assurant régulièrement ces dernières peuvent se le permettre, tout en équilibrant la charge du reste du trafic .

Le chapitre 3 présente FOCAL. Nous y montrons comment FOCAL est déduit de la solution optimale d'un problème de minimisation de la latence. De plus, nous analysons la stabilité de FOCAL, évaluons en profondeur son implémentation au sein de CCNPL-Sim en C++ sur diverses topologies de réseau et analysons sa sensibilité à différents paramètres.

B.4.1 Algorithmes de gestion de caches sensibles à la latence

L'effet conjoint de la mise en cache et de le transfert de paquets sensibles à la latence, repose sur un nouveau mécanisme de mise en cache stochastique, exploitant des informations de latence. Ce mécanisme ne requiert aucune coordination explicites entre les caches. Cette nouvelle politique de mise en cache sensible à la latence est appelée *LAC+* et s'appuie sur la proposition de LAC dans [Carofiglio et al., 2015c] que nous avons résumé précédemment. L'innovation qu'apporte LAC+ par rapport à LAC consiste à renforcer la dépendance à la latence dans les décisions probabilistes de mise en cache, sur la base d'une surveillance et d'une estimation en ligne du moment de second ordre de la distribution de la latence.

B.4.2 Stratégies de transmission sensibles à la latence

La réduction de la latence peut également être obtenue grâce à des stratégies de transfert de requêtes de type saut-par-saut visant à minimiser *i)* distance au premier cache où le l'évènement hit se produit, *ii)* la congestion dans le réseau. Dans cette optique, la présence de multiples chemins est clairement essentielle. En utilisant comme base de comparaison l'approche *transfert aléatoire uniforme* qui sélectionne à l'aveugle avec des interfaces de sortie de probabilité égales dans le FIB, nous ciblons la famille d'approches d'équilibrage de charge dynamiques et distribuées dont l'objectif est d'orienter les demandes de contenus dans le temps et à travers les interfaces de sortie disponibles de manière

à minimiser *i) -ii)* en moyenne. Dans [Carofiglio et al., 2013c], un schéma d'équilibrage de charge est dérivé d'une optimisation conjointe du taux de transfert/contrôle de congestion, et du transfert multi-chemin sous l'objectif de minimiser la charge du lien le plus chargé du réseau. La minimisation de la charge du lien le plus chargé implique une réduction significative de la latence moyenne globale telle qu'elle peut être appréciée dans les scenarii simulés. Ci-après, nous nous référons à une telle approche simplement comme *Load Balancing (LB)*. LB sélectionne au hasard les interfaces de sortie disponibles par entrée FIB selon les poids calculés. Au début, chaque interface a le même poids égal à un et le processus de transfert aléatoire est uniforme par rapport aux interfaces de sortie disponibles. Cela permet de sonder toutes les interfaces disponibles et de surveiller le nombre moyen d'intérêts en attente (PI) par entrée FIB et par interface. Une telle métrique reflète la latence résiduelle due à la distance au premier cache, ainsi que le niveau de congestion. Après cette phase initiale, le calcul des poids pilotant la sélection de l'interface de sortie consiste simplement à prendre le nombre moyen de PI par entrée FIB et par interface de sortie normalisée au nombre moyen total de PI par entrée FIB (de sorte que les poids soient compris entre 0 et 1). Idéalement, LB fonctionne sur les entrées FIB par contenu permettant un assez fin équilibrage de charge à l'échelle des flux. Cependant, une approximation faisable qui restreint le nombre d'états à maintenir dans le FIB remplace les entrées par contenu par des entrées par préfixe regroupant ainsi tous les noms de contenu derrière le même préfixe (la recherche FIB est supposée être celle du préfixe le plus long abrégé LPM). Notez que dans nos simulations, nous adoptons une approche LB par contenu avec l'objectif de quantifier ses meilleures performances atteignables.

LB peut améliorer significativement le débit / la latence globale perçue par l'utilisateur final, en comparaison avec au transfert aléatoire uniforme grâce à l'utilisation informée par la charge, de plusieurs routes par le téléchargement. Cependant, il n'est pas capable de réaliser une coordination implicite des caches le long de différents routes, en raison de sa répartition aléatoire pondérée, qui répartit les Intérêts pour le même contenu sur toutes les interfaces de sortie en fonction des poids. Pour comprendre ce problème, considérons le cas de trois interfaces de sortie disponibles pour un contenu donné k avec les poids associés, w_1, w_2, w_3. À chaque demande de contenu k, LB divise le processus d'arrivée d'intérêt au fil du temps en trois processus de sortie, avec un taux de w_1, w_2, w_3 par rapport au total, sans sélectionner la même interface de sortie pour une demande de morceaux de contenu donnée. En conséquence, le processus d'arrivée dans les caches le long des trois

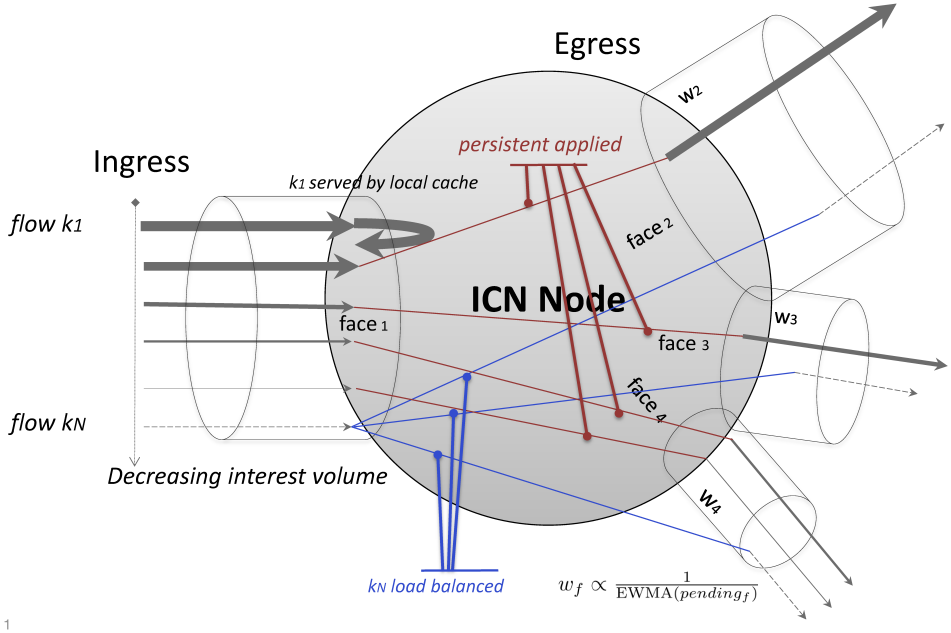


Figure B.1 – L'algorithme de sélection d'interfaces: équilibrage de charge avec sélection persistante d'interface pour les contenus les plus populaires.

chemins a les mêmes caractéristiques que l'original (à l'exception du taux), ce qui conduit les caches à fonctionner indépendamment et à stocker les mêmes objets.

Intuitivement, un tel comportement avantage les objets les plus populaires avec un taux de hits élevé sur tous les chemins disponibles, mais réduit les performances globales des caches en raison du manque de coordination entre eux. En revanche, la division des Intérêts de manière à persister la sélection d'une ou de quelques interfaces de sortie unique au cours du temps pour chaque contenu (tout en conservant l'équilibrage de charge par préfixe en fonction des poids LB) différencierait le processus d'arrivée dans les caches le long des trois chemins de l'exemple précédent, donc réaliserait une coordination implicite des caches, conduisant à de meilleures performances globales. Cette idée, basée sur la Proposition 3.1, la Proposition 3.2 et le Corollaire 3.2.1, inspire notre proposition d'un schéma d'équilibrage de charge amélioré, que nous nommons *LB- Perf* (équilibrage de charge avec un transfert permanent).

Algorithm 7: Les contenus les plus populaires sont échantillonnés dans PopularFiles. Puis des faisceaux de flux sont créés, un par entrée FIB, en fonction du volume d'intérêt observé. Ensuite, on leur associe les interfaces qui les transféreront en mode persistant

```
At update time (every ΔT);
Faces are ranked every ΔT_f ≫ ΔT;
T ← T + ΔT ;
IsPersistentDisabled = FALSE;
foreach FileName in PopularFiles do
    prefix ← GetFIBPrefix(FileName) ;
    OuputFaces ← GetOutputFaces(prefix) ;
    FlowBundle ← GetFlowBundle(prefix) ;
    Face ← OuputFaces.Begin();
    Sort(FlowBundle by InterestCounter) ;
    if ( T ≥ ΔT_f ) then
        Sort(FaceRecord by weight) ;
        T ← 0 ;
    end
    CumSum ← 0 ;
    foreach ( FlowRecord in FlowBundle ) do
        CumSum ← CumSum + FlowRecord.Popularity() ;
        while ( Face ≠ OuputFaces.End() ) do
            if ( CumSum < Face.weight ) then
                FlowRecord.SetFace(Face);
                break ;
            else
                CumSum ← CumSum + FlowRecord.Popularity() ;
                Face ← OuputFaces.Next();
            end
        end
    end
end
```

Algorithm 8: Sélection persistante d'interface de sortie basée sur la popularité des contenus.

```
At Interest I arrival with name /p/file_name/chunk_name ;
I matches name prefix /p in the FIB ;
OuputFaces ← GetOutputFaces(prefix = /p) ;
if ( IsPersistentDisabled ) then
    LoadBalancing.Update(OuputFaces.weights) ;
    FaceID ← LoadBalancing.SelectFrom(OuputFaces) ;
else
    PopularitySampler.Insert(I) ;
    if (PopularitySampler.Find(FileName(I))) then
        PopularFiles.ManageHit(I) ;
    end
    if (PopularFiles.IsPopular(FileName(I))) then
        FlowRecord ← FlowBundle.Find(FileName(I)) ;
        FlowRecord.InterestCounter++ ;
        FlowBundle.Norm++ ;
        FaceID ← FlowRecord.GetFace() ;
        if ( FaceID isEmpty ) then
            FaceID ← LoadBalancing.GetFace(OuputFaces) ;
        end
    else
        FaceID ← LoadBalancing.GetFace(OuputFaces) ;
    end
end
DoSendInterest(I, FaceID) ;
function ManageHit(Interest = I)
    prefix ← GetFIBPrefix(FileName(I)) ;
    FlowBundle ← GetFlowBundle(prefix) ;
    if PopularitySampler.IsPopular(Filename(I)) then
        FlowRecord ← FlowBundle.Find(FileName(I)) ;
        if FlowRecord isEmpty then
            FlowBundle.Insert(FileName(I)) ;
        end
    end
end
```

Dans une première phase, LB-Perf calcule les poids par préfixe à associer aux interfaces de sortie disponibles comme dans le cas de LB. FOCAL est également équipé d'un échantillonneur de popularité qui identifie en régulièrement les objets les plus populaires et stocke leur nom localement. La méthode d'estimation de popularité à la volée est hors-de-portée de cette thèse, bien que dans nos simulations, nous ayons utilisé un filtre k-LRU [Martina et al., 2013]. Dans nos simulations, nous définissons pour chaque sous-cache du k-LRU une capacité de 50 objets (dans le premier scénario simple) ou à 160 (dans les autres scénarii). Les objets trouvés dans le dernier sous-cache sont considérés d'une popularité élevée et tous leurs morceaux sont alors comptabilisés afin d'estimer la taille du flux correspondant.

Par essence, l'estimation de la taille des flux est volatil. Pour le stabiliser, nous supposons que les tailles de flux normalisées suivent une loi de puissance inconnue. Pour inférer les paramètres de cette loi, nous calculons le logarithme des tailles de flux normalisées et ajustons le modèle linéaire obtenu en utilisant les moindres carrés ordinaires.

Pour une mise en oeuvre plus générale de notre mécanisme, nous ne suggérons pas d'utiliser k-LRU qui, tout en étant implémenté, exige que k soit très grand lorsque la popularité est mesurée en termes de trafic observé et non en termes de nombre de demandes de contenu. Cependant, une demande de contenu (objet ou fichier) est difficile à identifier dans la pratique, car différents clients peuvent demander le même objet en utilisant des permutations distinctes des numéros de séquence des morceaux de contenus.

Pour un préfixe donné, les contenus les plus populaires sont regroupés en faisceaux de flux comme indiqué dans Algorithme 7. Chaque faisceau contient des éléments avec une popularité consécutive jusqu'à la taille relative de l'interface qui les acheminera, d'où le fait que la taille de chaque faisceau dépende du poids de l'interface, comme indiqué dans Algorithme 7. Ainsi, pour des éléments plus populaires, une seule interface de sortie est toujours sélectionnée en commençant par les interfaces les moins congestionnées (avec des poids plus élevés). Pour tous les autres éléments, la sélection des interfaces obéit à la règle LB standard, voir Algorithme 8. Chaque ΔT secondes, les contenus les plus populaires sont réassignés à des faisceaux de flux en fonction des poids des interfaces qui, d'autre part, sont eux aussi mis à jour indépendamment.

B.5 Équité dans les réseaux centrés sur l'information

Nous avons étudié l'équité de l'allocation du débit aux contenus lorsque les caches deviennent omniprésents dans un réseau [Bonald et al., 2017]. Etant donné que la mise en cache des objets les plus populaires est l'approche suivie par nos propres algorithmes de gestion de cache, une distorsion sévère de l'équité en serait-elle une conséquence que des algorithmes de gestion de cache sont supposés résorber?

Il s'avère que s'assurer que les objets les plus populaires occupent les caches, comme LFU l'organise, est le pendant gestion en cache d'une solution aux problèmes de distribution de contenus α -équitable - l'ordonnancement équitable niveau contenus des paquets en est le complément.

En d'autres termes, les politiques de gestion de cache n'ont pas besoin d'être conçues pour l'équité, comme l'ont suggéré les travaux précédents. Ne se concentrant pas sur le taux de hits du cache, mais sur l'équité du débit, nous montrons que l'α *-équité en ICN peut être traitée de la même manière que dans les réseaux traditionnels, par l'intermédiaire de l'ordonnancement des paquets*.

B.5.1 Cache Network Model

D'abord, nous présentons un modèle mathématique qui capture la dynamique de l'ensemble du réseau. Le modèle le considère comme un réseau de files d'attente où les caches contribuent à augmenter le taux de service. Nous visons à maximiser une fonction d'utilité du trafic exogène admissible. Se référer au tableau B.2 pour la notation et à B.2 pour un synoptique du modèle utilisé ci-après.

Hypothèses

- Les processus stochastiques $\{\lambda_{k,n,b}(t)\}_{0\leq t\leq T}$ en tant que taux exogène, $\{\mu_{k,n,b}(t)\}_{0\leq t\leq T}$ en tant que taux de service et $\{h_{n,k}(t)\}_{0\leq t\leq T}$ en tant que taux de hits sont indépendant.
- Le réseaux route un seul préfixe.
- Tous les objets ont la même taille.

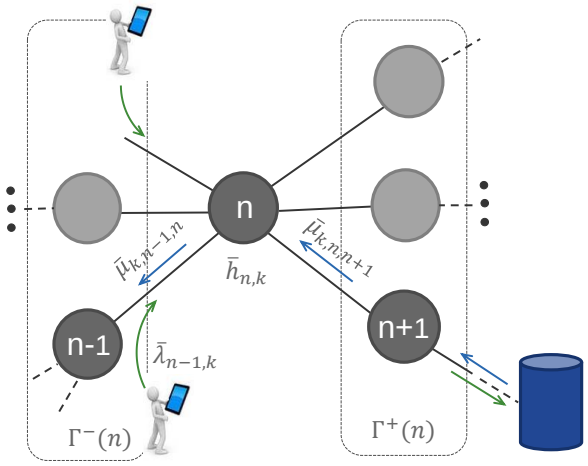


Figure B.2 – Réseaux transportant le contenu k à travers le cache n.

- La taille des caches n'est jamais nulle.

- Les serveurs de contenus ne sont pas eux-mêmes des clients.

- Le trafic exogène à un noeud donné est celui généré par une application locale qui n'est pas satisfait par le cache local.

- Nous supposons un contrôle de la congestion saut-par-saut c'est-à-dire que les intérêts sont envoyés en moyenne à un taux équivalent au taux de service du lien.

Nous définissons une file d'attente d'intérêts (PIQ) comme le nombre d'intérêts en attente par contenu et par face. Un intérêt mis en file d'attente dans une PIQ est servi lorsque le paquet de données correspondantes revient.

Soit $Q_{k,n,b}(t)$ la taille de la file d'attente d'intérêts pour le contenu k à l'instant t pour le lien (n, b). $h_{n,k}(t) \equiv \mathbb{1}_{\{k \text{ in cache } n \text{ at } t\}}$ indique si le contenu k a été trouvé dans le cache n à l'instant t. L'évolution au cours du temps de la taille de la PIQ du contenu k, pour les

$n \in \mathcal{N}$	Identifiant de noeud ICN. $\mathcal{N} \subseteq \mathbb{N}$.
$t \in \mathbb{R}_+$	Instant auquel un contenu est reçu.
$k \in \mathcal{K}$	Rang du contenu en terme de popularité. $k = 1$ est celui du contenu le plus populaire, tandis que $\|\mathcal{K}\|$ correspond à l'objet le moins populaire.
$\Gamma^-(n)$	Ensemble des noeuds d'entrée du noeud n.
$\Gamma^+(n)$	Ensemble des noeuds de sortie du noeud n.
$\bar{\mu}_{k,n,b}$	Moyenne à long terme du taux de service pour le contenu k sur le lien (b, n)
$\bar{\lambda}_n$	Moyenne à long terme du taux d'arrivée exogène des intérêts au noeud n.
$\lambda_{n,k}$	Moyenne à long terme du taux d'arrivée exogène des intérêts pour le contenu k au noeud n.
q_k	Popularité du contenu k. Il s'agit de la probabilité qu'un contenu lorsque requis soit le contenu k. Elle est strictement ordonnée: $q_{k+1} < q_k$.
$\mathbb{1}_{\{.\}}$	Fonction indicatrice.
$\bar{\Lambda}_{n,k}$	Borne supérieure de la moyenne à long terme du taux d'arrivée exogène des intérêts du contenu k au noeud n .
$\varsigma(k)$	Ensemble des serveurs des contenus k.
$h_{n,k}$	Taux de hits du contenu k au noeud n
$C_{b,n}$	Capacité du lien (b, n) en morceaux de contenu/s.
x_n	Capacité du cache n en nombre d'objets.

Table B.2 – Notation.

noeuds de sortie $b \in \Gamma^+(n)$, au noeud n est bornée supérieurement comme suit:

$$\begin{aligned} Q_{k,n,b}(0) = 0, \forall b \in \Gamma^+(n) \text{ and} \\ \sum_{b\in\Gamma^+(n)} \frac{d}{dt} Q_{k,n,b}(t) \leq \lambda_{n,k}(t) \\ + (1 - h_{n,k}(t)) \sum_{a\in\Gamma^-(n)} \mathbb{1}_{\{Q_{k,a,n}(t)>0\}} \mu_{k,a,n}(t) \\ - \sum_{b\in\Gamma^+(n)} \mathbb{1}_{\{Q_{k,n,b}(t)>0\}} \mu_{k,n,b}(t). \end{aligned} \tag{B.10}$$

Le taux de service $\mu_{k,a,b}(t)$ de la PIQ est le débit du contenu k sur le lien (b, a) à l'instant t. $\lambda_{n,k}(t) \equiv q_k \lambda_n(t)$ est le taux d'arrivée exogène des intérêts pour le contenu k au noeud n et à l'instant t. De la Proposition 3.3, on déduit le débit admissible maximal $\bar{\Lambda}_{n,k}$ donné par:

$$\bar{\Lambda}_{n,k} \equiv \sum_{b\in\Gamma^+(n)} \bar{\mu}_{k,n,b} - (1 - \bar{h}_{n,k}) \sum_{a\in\Gamma^-(n)} \bar{\mu}_{k,a,n}, \forall n, k \tag{B.11}$$

sous les contraintes suivantes:

$$\sum_{k\in\mathcal{K}} \bar{\mu}_{k,a,n} \leq C_{n,a}, \quad \forall n, a \in \Gamma^{-}(n) \tag{B.12}$$

$$\sum_{k\in\mathcal{K}} \bar{h}_{n,k} = x_n, \quad \forall n \tag{B.13}$$

$$0 \leq \bar{h}_{n,k} \leq 1, \quad \forall n, k \tag{B.14}$$

$$\bar{\mu}_{k,a,n} \geq 0, \quad \forall n, k, a \in \Gamma^{-}(n) \tag{B.15}$$

$$\bar{\lambda}_{n,k} \leq \bar{\Lambda}_{n,k}, \quad \forall n, k. \tag{B.16}$$

Nous ignorons dans ce travail la contrainte (B.16) qui impose une borne inférieure au taux du service des contenus Cela signifie que le réseau ne garantit pas que certains contenus demandés sur un noeud donné seront satisfaits. Cela dépendra entièrement de l'optimalité de le servir.

Formulation du problème Nous allons maintenant insérer le taux d'arrivée admissible dans une fonction d'utilité équitable $U(\cdot)$. Le débit alloué au contenu k sur tout le réseaux est définit par

$$\phi_k \equiv \sum_{n\in\mathcal{N}} \bar{\Lambda}_{n,k} = \sum_{\substack{b\in\cup_k \varsigma(k) \\ n\in\Gamma^{-}(b)}} \bar{\mu}_{k,b,n} + \sum_{\substack{n\in\mathcal{N} \\ a\in\Gamma^{-}(n)}} \bar{h}_{n,k}\bar{\mu}_{k,a,n}. \tag{B.17}$$

La fonction objectif consiste à trouver le taux de service optimal pour chaque contenu et chaque lien, ainsi que le taux de hits tels que

$$\underset{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}}{\text{maximize}} \sum_{k\in\mathcal{K}} q_k U(\phi_k/q_k), \tag{B.18}$$

avec $U(\cdot)$ une fonction d'utilité α-équitable.

L' α-équité pondérée a été introduite dans [Mo and Walrand, 2000], puis adaptée pour l'allocation de caches par [Dehghan et al., 2016]. Cependant, comme exprimé dans l'équation B.18, nous préconisons une formulation de l'α-*équité pondérée* qui opère sur des taux par unité de poids. La raison de ce choix est sa convergence vers l'équité de *max-min pondérée* lorsque $\alpha \to \infty$, tandis que la formulation pondérée originale de Mo

et Walrand tend vers l'équité max-min [Mo and Walrand, 2000]. Fait intéressant, notre formulation donne des solutions indépendantes de α, produisant donc des allocations simplement équitables.

$$\text{Since } q_k U(\phi_k/q_k) \equiv q_k \frac{(\phi_k/q_k)^{1-\alpha}}{1-\alpha} = q_k^{\alpha} \frac{\phi_k^{1-\alpha}}{1-\alpha}, \alpha \neq 1,$$

L'objectif se simplifie en

$$\underset{\bar{\boldsymbol{\mu}},\bar{\boldsymbol{h}}}{\text{Maximizer}} \quad \begin{cases} \sum_{k\in\mathcal{K}} q_k^{\alpha} U(\phi_k), & \text{if } \alpha \neq 1 \\ \sum_{k\in\mathcal{K}} q_k \log(\phi_k/q_k), & \text{otherwise.} \end{cases} \tag{B.19}$$

Les cas particuliers que cette α-équité pondérée inclut sont:

- pour $\alpha = 1$, l'objectif est dit celui de l'équité proportionnelle pondérée [Kelly et al., 1998].
- lorsque α est infini, l'objectif correspond à l'équité max-min pondérée c'est-à-dire $\max\min(\phi_k/q_k)$ [Radunović and Boudec, 2007].

Par la suite, l'attribut *pondéré* sera sous-entendu lorsque omis.

Solution

Allocation générale α-équitable La première propriété des allocations de α-équitables des contenus dans les réseaux de cache est leur efficacité de Pareto. On dit que l'attribution est Pareto-efficace si toute tentative d'augmenter la part d'un contenu diminue la part d'un autre contenu. Dans notre problème d'optimisation, cela se traduit par le fait que la capacité des liens ait été entièrement attribuée.

Property B.1 (Efficacité au sens de Pareto pour tout $\alpha \geq 0$)**.** L'allocation de bande passante α-équitable est Pareto-efficace car l'allocation de ressource optimale utilise

toute la capacité des liens pour servir les contenus *c'est-à-dire que*

$$\sum_{k\in\mathcal{K}} \bar{\mu}^*_{k,a,n} = C_{n,a}, \quad \forall n \in \mathcal{N}, \forall a \in \Gamma^-(n). \tag{B.20}$$

Proof. Voir Propriété 4.1 □

A présent, notre résultat principal. Il a établi que ICN, en adoptant LFU comme politique de gestion de cache maximisant le taux de hits , a de facto adopté la politique de gestion de cache optimale quant à l'équité du débit des contenus.

Proposition B.5 (LFU satisfait l'α-équité)**.** LFU est une politique de gestion de cache pour réseaux visant à α-équité des débits alloués aux contenus, pour tout $\alpha \geq 0$.

Proof. Voir Proposition 4.1. □

Ce résultat est important car il montre que l'algorithme LFU et ses heuristiques (LRU, p-LRU, LRU-k, LRU + LCD) peuvent conduire à des réseaux α-équitables, $\forall\alpha \geq 0$. Les ordonnanceurs de paquets seraient chargés de l'autre partie de la solution optimale, un partage de bande passante équitable entre contenus. Nous nous désignons ce dernier en tant que partage de bande passante aux contenus α-équitable. Il est mathématiquement tractable grâce à la concavité du problème, compte tenu des taux de hits binaires, car la concavité est une condition suffisante pour l'existence d'un optimum global. En outre, un partage de bande passante aux contenus qui soit α-équitable est réalisable en pratique dans le paradigme ICN car les paquets sont nommés de manière unique d'après les contenus qu'ils transportent.

Il convient également souligner la nouveauté de ce résultat, car les travaux antérieurs [Dehghan et al., 2016] ont abouti à des conclusions très différentes. C'est parce qu'ils ont seulement examiné des caches isolés, ont constaté que l'équité nécessitait des taux de hits fractionnés pour chaque valeur du paramètre d'équité α supérieur à zéro et des algorithmes pour les caches TTL conçus de manière idoine. Leurs algorithmes de mise en cache consistent à ajuster les Temps-à-vivre (TTL) de chaque contenu via une descente en gradient.

Pour finir, l'algorithme suivant (Alg. 9, Alg. 10) est un exemple d'implémentation de l'équité max-min pondérée par contenu distribué. Il s'appuie sur un ordonnanceur de

type Deficit Round-Robin [Shreedhar and Varghese, 1995] pour réaliser une allocation de bande passante aux contenus qui soit max-min équitable, compte tenu d'un substrat de gestion de cache LFU.

Algorithm 9: Allocation α-équitable des contenus dans ICN

```
Input: Data packet, α
Cache.Insert(packet, Policy::LFU);
FairQueuing.Shape(packet, α);
```

Algorithm 10: Allocation max-min équitable des bandes-passantes de contenus

```
function FairRate.Shape(Data packet, α)
    FairQueuing.Queue[packet.ContentName()].Push(packet);
    if α == ∞ then
        FairQueuing.SendData(Policy::DEFICIT_ROUND_ROBIN);
    end
end
```

B.6 AFFORD: Ask For Directions, machine learning-based routing

La dernière contribution de cette thèse traite du coût prohibitif de l'exécution de l'algorithme de recherche du plus long préfixe dans les FIB de taille considérable. En effet, compte tenu de l'espace de noms illimité qui caractérisera l'Internet des objets à venir, il sera difficile de transférer des paquets à chaque noeud en escomptant une connaissance exacte des chemins à prendre. Avec Ask For Directions (AFFORD) [Mekinda and Muscariello, 2016], nous proposons *d'entrainer un ensemble de petits réseaux de neurones artificiels dans le plan de contrôle et les interroger rapidement, dans le plan de données, à propos des sauts suivants les plus probables*.

Le chapitre 5 explique et analyse AFFORD, cette nouvelle approche de routage. Nous en faisons en outre l'évaluation sur des jeux de données de tailles diverses.

Cache, Process and Forward in Information-Centric Networking

Léonce MEKINDA

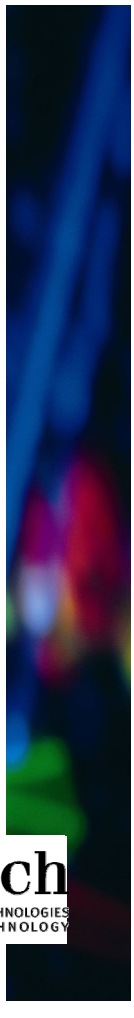
ABSTRACT: This thesis investigates how making content caching and forwarding latency-aware can improve data delivery performance in Information-Centric Networks (ICN). We introduce a new very effective contribution to the existing content caching toolset. The designed mechanism leverages retrieval time observations to decide whether to store an object in a network cache, based on the expected delivery time improvement. We demonstrate that our distributed latency-aware caching mechanism, LAC+, outperforms state of the art proposals and results in a reduction of the content mean delivery time and standard deviation of LRU caches by up to 60%, along with a fast convergence to these figures.

In a second phase, we conjointly optimize the caching function and the multipath request forwarding strategies as both coexist in ICN at network level and should reinforce each other. To this purpose, we introduce the mixed forwarding strategy LB-Perf, directing the most popular content towards the same next hops to foster egress caches convergence, while load-balancing the others.

Third, we address ICN fairness to contents. We show that traditional ICN caching, which favors the most popular objects, does not prevent the network from being globally fair, content-wise. The incidence of our findings comforts the ICN community momentum to improve LFU cache management policy and its approximations. We demonstrate that in-network caching leads to content-wise fair network capacity sharing as long as bandwidth sharing is content-wise fair.

Finally, we contribute to the research effort aiming to help ICN Forwarding Information Base scale when confronted to the huge IoT era's namespace. We propose AFFORD, a novel view on routing in named-data networking that combines machine learning and stochastic forwarding. More specifically, we show that compressing the Forwarding Information Base into bitwise trie-indexed Artificial Neural Networks accelerates next hop lookup and reduces Forwarding Information Base's size by orders of magnitude.

KEYWORDS: Information-Centric Networks; Performance evaluation; Cache management; Transport protocol; Forwarding; Machine learning.



TELECOM ParisTech

EDiTE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY