Revisiting Resource Utilization in The Internet: Architectural Considerations and Challenges

Somaya Arianfar

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 209/2014

Revisiting Resource Utilization in The Internet: Architectural Considerations and Challenges

Somaya Arianfar

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall S4 of the school on 15 January 2015 at 1 pm.

Aalto University
School of Electrical Engineering
Department of Communications and Networking
Networking Technology

Supervising professor
Prof. Jörg Ott

Thesis advisor
Dr. Pasi Sarolahti

Preliminary examiners
Prof. Christian F. Tschudin, University of Basel, Switzerland
Dr. Volker Hilt, Alcatel Lucent, Germany

Opponent
Dr. Karen Sollins, MIT, USA

Aalto University publication series
DOCTORAL DISSERTATIONS 209/2014

ISBN 978-952-60-6017-0 (printed)
ISBN 978-952-60-6018-7 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-6018-7

Unigrafia Oy
Helsinki 2014

Finland

Publication orders (printed book):
somaya.arianfar@aalto.fi

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Somaya Arianfar

Name of the doctoral dissertation
Revisiting Resource Utilization in The Internet: Architectural Considerations and Challenges

Publisher School of Electrical Engineering

Unit Department of Communications and Networking

Series Aalto University publication series DOCTORAL DISSERTATIONS 209/2014

Field of research Networking Technology

Manuscript submitted 9 December 2013 **Date of the defence** 15 January 2015

Permission to publish granted (date) 13 November 2014 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

The Internet has been a success story for many years. Recently researchers have started to deal with new questions that challenge the effectiveness of the Internet architecture in response to the new demands, e.g. overwhelming traffic growth and latency optimizations. Various proposals ranging from new application level protocols to new network stacks are emerging to help the Internet to keep up with the demand.

In this dissertation we look at a few different proposals that deal with improving the speed and resource utilization in the Internet. We first discuss improving the resource utilization in the current Internet by minor changes such as adjusting various parameters in TCP. We then discuss a more radical form of resource utilization through combining the network and the available storage. Combining these two resources, which have traditionally been considered separate, could provide many new speed improvement opportunities. We discuss relaxing the barrier between the storage and the network in the context of Information Centric Networking (ICN), which in itself is an alternative proposals to the current TCP/IP style Internet. With the help of ICN, we propose different forms of in-network caching below the application layer.

We argue that, although useful, the new models of utilizing network resource could show to have their own challenges. We namely discuss the resource management and privacy challenges that are introduced with ICN in general and within our proposed solutions in particular. The lack of end-host bindings and the existence of network routable data names in different data chunks make the congestion control, reliability, and privacy in ICN rather different from TCP/IP. We discuss some of these differences and propose solutions that can help addressing each issue in our particular form of ICN-based mechanisms.

Keywords Information Centric Networking, ICN, Internet Architecture

ISBN (printed) 978-952-60-6017-0	**ISBN (pdf)** 978-952-60-6018-7	
ISSN-L 1799-4934	**ISSN (printed)** 1799-4934	**ISSN (pdf)** 1799-4942
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2014
Pages 152	**urn** http://urn.fi/URN:ISBN:978-952-60-6018-7	

Preface

This thesis has been a long journey of continuous questioning, learning, and pushing my limits. I have had the chance to meet and work with many wonderful people, who influenced the way I think and the way I work. It is impossible to name everyone, but I will forever be in debt to all of them.

I am grateful to my supervisor Jörg Ott who has taught the importance of patience in order to be both a good teacher and a good researcher. I will never forget all those times, when over and over again, Jörg tried to teach me how to do research and how to express my ideas in a more clear form.

I would also like to thank Pekka Nikander for originally shaping and motivating my research interests. Pekka showed me the importance of teamwork, and how everyone in a team matters.

I express my gratitude to Pasi Sarolahti who has been a great advisor and an even greater colleague. Pasi helped me to bring to life many of my ideas by discussing them. He gave me the time and the courage to be an explorer.

I am thankful to Prof. Scott Shenker for giving me the opportunity to work with him and his amazing group in Berkeley, California. Scott's level of energy, hard work, and positive attitude made my visit to Berkeley one of my best life experiences.

I like to thank my colleagues at Ericsson research and Cisco systems with whom I got the chance to see the world differently. I would also like to thank my colleagues in different EU projects specially to my co-authors in the resulting publications; Petri Jokela, Andras Zahemszky, Christian Esteve Rothenberg, Teemu Koponen, Barath Raghavan, Nikos Fotiou, Mikko Särelä, George Polyzos, Lars Eggert, and Walter Wong.

I extend my gratitude to my pre-examiners Dr. Volker Hilt and Prof. Christian Tschudin for their helpful comments and Mark Stapp who has

read and commented on my thesis.

I will also like to acknowledge Academy of Finland's Future Internet Graduate School, EU FP7 projects PSIRP and PURSUIT, Tekes funded FI-SHOK, Ericsson, and Aalto University who provided funding for my research.

Finally I would like to express my deepest gratitude to all the amazing individuals in my life just for being there; my mother who has taught me to fight and never accept failure; to my loving brother and sister, Bardia and Roya, for always being there for me, and to my dearest Ahmad for letting me be myself.

Boston, November 27, 2014,

Somaya Arianfar

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Somaya Arianfar, Pasi Sarolahti, Jörg Ott. Marooned Magic Numbers – An Adaptive Congestion Control Architecture. In *IEEE International Conference on Network Protocols (ICNP)*, Austin, Texas, pp. 1-11, October 2012.

II Somaya Arianfar, Pekka Nikander, Jörg Ott. On Content-Centric Router Design and Implications. In *ACM Re-architecting the Internet Workshop (ReArch)*, Philadelphia, pp. 25-31 , December 2010.

III Somaya Arianfar, Pasi Sarolahti, Jörg Ott. Reducing Server and Network Load with Shared Buffering. In *ACM Capacity Sharing Workshop (CSWS)*, Nice, France, pp. 33-38 , December 2012.

IV Somaya Arianfar, Pasi Sarolahti, Jörg Ott. Deadline-based resource management for Information-centric Networks. In *ACM Information Centric Networking Workshop (ICN)*, Hong Kong, pp. 49-54 , August 2013.

V Somaya Arianfar, Teemu Koponen, Barath Raghavan, Scott Shenker. On Preserving Privacy in Content-Oriented Networks. In *ACM Information Centric Networking Workshop (ICN)*, Toronto, Canada, pp. 19-24 , August 2011.

Author's Contribution

Publication I: "Marooned Magic Numbers – An Adaptive Congestion Control Architecture"

I have initiated the ideas on Pathlets, and increasing TCPs initial window size in discussion with other team members in early 2012. I have simulated, and evaluated these ideas in ns-3. I also participated in the process of writing the paper.

Publication II: "On Content-Centric Router Design and Implications"

I have discussed the basic idea of packet level caching in Information Centric Networks with Pekka Nikander in 2009. Later on, I have simulated, and evaluated these ideas in ns-3. I also took part in writing the paper.

Publication III: "Reducing Server and Network Load with Shared Buffering"

I have been a member of the team discussing the ideas on shared buffering. I have simulated, and evaluated these ideas in ns-3. I also participated in writing the paper.

Publication IV: "Deadline-based resource management for Information-centric Networks"

The original idea for this paper came from my discussions with the co-authors of this paper regarding the possibility of using deadlines for better control of network congestion and supporting reliability. I have simulated,

and evaluated these ideas in ns-3. I also participated in the process of writing the paper.

Publication V: "On Preserving Privacy in Content-Oriented Networks"

This paper was born from the discussion I had with Scott Shenker in Berkeley. I have been part of the team discussing and writing the ideas in this paper.

List of abbreviations

ACK	ACKnowledgment
ALF	Application Level Framing
ALTO	Application Level Traffic Optimization
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
AS	Autonomous System
BGP	Border Gateway Protocol
CCWND	Conceptual Congestion control WiNDow
CDN	Content Delivery Network
CM	Congestion Manager
CPU	Central Processing Unit
DNS	Domain Name System
DONA	Data-Oriented Network Architecture
DRAM	Dynamic Random-Access Memory
DSL	Digital Subscriber Line
ECN	Explicit Congestion Notification
FCT	Flow Completion Time
FIFO	First-In-First-Out
HTTP	HyperText Transfer Protocol
I/O	Input/Output

ICN	Information Centric Networking
ICP	Interest Control Protocol
ICW	Initial Congestion Window
IP	Internet Protocol
ISP	Internet Service Provider
LIPSIN	LIne speed Publish/Subscribe Inter-Networking
LRU	Least Recently Used
MFU	Most Frequently Used
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NDN	Named-Data Networking
NetInf	Network of Information
OSPF	Open Shortest Path First
PId	Packet Identifier
PSIRP	Publish-Subscribe Internet Routing Protocol
PTA	Pathlet-based Transport Architecture
PURSUIT	PURSUing a pub/sub InterneT
QoS	Quality of Service
RCP	Rate Control Protocol
Re-ECN	Re-feedback of Explicit Congestion Notification
RTT	Round Trip Time
SRAM	Static Random-Access Memory
SYN	SYNchronize
TCP	Transmission Control Protocol
TOE	Transport Offload Engine
TRIAD	Translating Relaying Internet Architecture
URI	Uniform Resource Identifier
XCP	eXplicit Congestion Control Protocol

1. Introduction

"In the dim background of our mind, we know what we ought to be doing but somehow we cannot start."

(William James)

The modern Internet was born in 1983 when TCP/IP replaced the ARPANET [39, 139]. Since then the Internet has been in use to provide different services and transmit various types of traffic. The amount of Internet traffic has increased tremendously from 1 Exabytes per year in 2001 to 31 Exabytes per month in 2011, and it is expected to grow even more in the future [87, 43]. This overwhelming traffic growth has had many implications including the increasing demand for better infrastructures and services. To support this demand, today, huge datacenters are in use to store and manage big amounts of data [81, 15], and cloud computing is becoming more popular in order to benefit from these big datacenters [22]. CDNs are other examples of new infrastructures that have been introduced to help dissemination of large amounts of data in the Internet [96, 108]. The traffic growth has even affected the underlying network and high-speed routers are in high demand to efficiently handle the increasing traffic [7].

During all these years the TCP/IP protocol suite has been successfully used to deal with all kinds of demands, and has remained unchanged. Although sometimes there has been concerns regarding inefficiency of the current Internet protocol suite (e.g. in [42]), there has never been an absolute need to go through the hassle of redesigning the TCP/IP protocol suite. Only recently, the research community has started to consider the fact that finally the traffic demand on the network, its growth rate and its diversity is reaching the point that may enforce an actual change to the current Internet protocol suite. Many new proposals such as TRIAD [73], DONA [93], PSIRP [1], NDN [8], and Architecting for Innovation [94]

challenge the TCP/IP model from different aspects including its primitive host-centric design [1, 8]. Although motivated from slightly different angles, all these proposals agree on the general need for a change, and they proceed to explore their view of the required change(s).

In this thesis, we explore two closely related criteria, which reflect the need for change in the current Internet protocol suite. The first criterion is **speed**: in a network with an overwhelming traffic growth, it is important to process and transmit the traffic fast enough, so that the increasing demand does not result in unmanageable backlogs. Scaling up the transmission and processing capacity in the network is one solution to adapt to the traffic growth. However, in this dissertation we mostly concentrate on the cases in which scaling-up the hardware resources is not an ultimate solution because of either economical or technical difficulties. Instead we discuss scenarios in which increasing the speed requires software and architectural changes to the current protocol suite, and this brings us to our second criterion: **resource utilization**. We argue that improving the speed, in addition to improving the capacity, also requires being able to get the most from all available network resources. Therefore, we seek improved in-network resource utilization, either achieved through a minor change in a protocol implementation or achieved through radical changes in the network architecture. We discuss both these criteria in this thesis, and describe our solutions to some of the problems that will be mentioned later on.

1.1 Problem statement

In this dissertation, we approach the speed as a fundamental issue in the Internet. We discuss that changing the design of the networking stack protocols as part of the speed improvement requirements could become inevitable in near future. We then argue from the protocol design perspective emphasizing speed could also be interpreted as requiring more efficient resource utilizations in the network. In this thesis we investigate how to efficiently use the available storage, processing, and communication resources in the network for speed improvement purposes. Part of this work focuses on combining the communication and storage in the network, enabling a model in which data is written once into the in-network memory, but can be read multiple times to alleviate the effects of an increasing traffic demand and improve speed. There are two fundamental

questions that form our basic research problems in this thesis:

What kind of protocols and abstractions could get the most out of the available network resources? This thesis investigates combining storage and communication as the fundamental approach in the network design. In particular, we look at different methods to efficiently use the available bandwidth and the in-network storage, both to improve the speed and reduce the load. In this thesis, we explore caching and data retrieval at finer granularities than what the current application-layer methods use today.

What are the challenges raised in regard to using new protocols, which change the resource usage model in the Internet? Using in-network storage in more efficient forms and combining it with communication is not that straightforward. This thesis investigates the challenges introduced by the new protocols and abstractions that help combining storage and communication. In particular, we investigate the resource management and privacy issues that exist within our proposed mechanisms. Both these issues are the result of the specific form of abstractions and protocol mechanisms that we require in our proposed models.

1.2 Context and methodology

In this thesis, two different set of networking protocols are used to motivate the speed and efficiency challenges and give an overview regarding the related literature. First, we use the TCP/IP protocol stack to describe the existing challenges and limitations for resource utilization and speed improvements. Second, most of our work focuses on Information-Centric Networks (ICN) [88, 8, 1], which is proposed as a solution to the availability, speed, and security problems that are believed to exist with the TCP/IP model. Using these two models, we have designed and simulated different mechanisms that help to utilize the network resources and deal with the introduced challenges.

Part of our work presented in this thesis uses the TCP/IP model as the basis for our experiments, and investigates how bandwidth and in-network storage might be used more efficiently. Designing different mechanisms in the context of the TCP/IP protocol suite makes them more applicable to the current Internet. However, a significant part of our work presented in this dissertation has originated in the context of different collaborative projects [1, 10] related to revisiting the Internet Ar-

chitecture. This has been done as part of the EU FP7 PSIRP [1] and PURSUIT [10] projects with the vision of a publish/subscribe-based [56] Information-Centric Networking (ICN). The clean-slate approach of these projects have been to apply a name-oriented publish/subscribe design to all different modules in the network protocol stack.

In the context of ICN, in one part of this dissertation we use the networking model proposed in [1] to investigate the possibility of in-router packet caching and data retrieval. For this purpose, we examine the challenges that might exist with designing a fast enough packet-caching mechanism in ICN environments. In another part we explore the resource management challenges that are created in ICN environments where storage and communication could be combined and packet caching is possible. We finally investigate the privacy issues that ICN models introduce as a price of their new abstraction models.

Our methodology for designing and evaluating our work relies on simulations and back of the envelope calculations. Most of the results discussed in this thesis are based on our ns-3 [78] simulations. For our TCP/IP based solutions, small pieces of code has been added to the existing ns-3 TCP/IP implementation. We have then evaluated the credibility of our results based on running different simulations with varying number of parameters in ns-3. For our ICN related solutions, we have implemented a new ICN stack in ns-3 that replaces the existing implementation in the IP layer and above. We have then used this publicly available code [1] for running different simulations.

1.3 Contributions

The research reported in this dissertation has been published in six original publications. The specific contributions of the original publications are as follows:

- Proposing a pathlet-based congestion control and transfer initialization model (Publication I): The transmission speed in the current Internet partly depends on the speed of the transport protocol. The transport control block in today's Internet is very much dependent on the TCP/IP model of unicast flows. In Publication I we explain and asses a mechanism for creating a more generalized form of transport control blocks

[1] The code is available at: http://users.piuha.net/blackhawk/contug/

in the stack, and for dynamically setting the initial congestion control values, both in hope of increasing the transmission speed.

- Demonstrating the feasibility of having caching routers: In Publication II we introduce a specific router model for the proposed Information-Centric Network architecture in PSIRP [1], and assess the benefits, and feasibility of having such store-and-forward information-centric routers.

- Introducing byte-stream caching in the network to reduce the server load and increase the speed: Publication III presents a novel mechanism for adding the byte-stream caching inside the network and benefiting from that in order to offload part of the servers operation to the network. In this proposal we apply different ICN concepts to the current TCP/IP model and introduce new mechanisms whenever applicable.

- Investigating the resource management challenges that exist in a network with store-and-forward caching routers: In [21] we examine an ICN network which implements the packet caching mechanism introduced in Publication II. We explore the congestion control problem that is created in this environment. The congestion control solution proposed in [21] benefits from a similar model to TCP but with minimal changes that makes it adaptable to router caching. This model is then analyzed through ns-3 [78] simulations.

- Proposing a new resource management mechanism in a network with store-and-forward caching routers: Publication IV discusses a novel resource management approach for the environments in which storage and communication are combined together, i.e. environments discussed in Publication II and Publication III. The mechanism explored in Publication IV uses packet deadlines for effective network resource management. We go into the details of deadline-based packet scheduling in the routers and how such model could benefit both the network and the applications.

- Exploring privacy issues caused by naming abstractions that alleviates Information-Centric designs and router caching : In Publication V we first describe the new form of privacy attacks that ICN networks enable. We then present a specific method of overcoming these privacy issues,

and discuss the tradeoffs that concern our solution.

1.4 Structure of the Thesis

This thesis is structured as follows: chapter 2 explores different areas motivating our research. In chapter 2 we examine different cases that emphasize the need for having an architectural solution for the speed and resource utilization issues in current Internet. We discuss the resource utilization model in the TCP/IP stack and point out some of its deficiencies and limitations. Chapter 3 describes ICN as a solution for many resource utilization problems described in chapter 2, and discusses the costs associated with it. In chapter 4 we then present different mechanisms that we have designed and analyzed for better resource utilization in the network. We discuss the challenges and costs associated to our solutions and address some of them in chapter 4. We finally conclude this thesis with a discussion in chapter 5.

2. Speed in the current Internet

"If a problem has no solution, it may not be a problem, but a fact not to be solved, but to be coped with over time."

(Shimon Peres (Peres's law))

2.1 Speed Matters

When roads were made, their main purpose was providing connectivity between different points. At that time if people wanted to travel somewhere, they first had to check if there was any road to that place or not. Nowadays, as roads have become an embedded concept in our everyday life, the question of connectivity does not seem that relevant anymore. The new questions are related to optimization: Which route is less costly to take? Which route is faster?

Similar to the roads, the Internet has formed a unique communication network that connects many nodes across the world. The Internet has been around long enough to raise optimization questions regarding its speed [67, 12, 14]: Could the Internet become any faster?. A high speed Internet is the one with high bit rate and low delay. The speed then depends on low-level physical properties mainly link capacities and transmission distance. The higher the link capacities and the closer the transmission distances are, the better the speed is.

However, speed is not all about physical properties. At the end what matters the most is the end-users' experience. Speed affects the end-users' experience and her level of satisfaction, while using different services. A study by Google [2] shows that half a second increase in showing the search results ends up to a 20% drop in traffic demands from an unsatisfied user. Therefore, today there are also an increasing number of

high-level network proposals, i.e. application and transport level protocols, which try to make the Internet faster from the end users' perspective. Many of these protocols aim at better utilization of the physical infrastructure instead of changing it. One example is Google's SPDY, which achieves the particular goal of decreasing the web page load time by using software methods such as compression [29].

Someone once said: speed is not just a feature, it's the feature [80]. Today, in addition to being an optimization question, speeding the Internet up is also becoming a crucial requirement for keeping up with the increasing traffic demand. As the amount of traffic grows the network could slow down because there would not be enough resources available for timely traffic handling. In the most extreme scenario, a dramatic slow down in the speed could even be interpreted as network unavailability. Thus, keeping the Internet available requires speeding it up, for example through increasing the bit rate or reducing the buffering delay.

For many people having faster Internet means having higher capacity links. However, in reality speed is affected not just by link capacity but also by the network design and the software that uses it. In this context speeding the Internet up does not necessarily mean increasing the link capacities, but it could also be interpreted as reducing the design-related delays introduced by the communication networks and their internal components. To understand how we could improve the speed in the Internet, here we describe the most important elements that affect the speed in the overall network.

2.1.1 The transmission delay

To speed up the transmission operation in the Internet, one needs to first identify the basic elements that contribute to the overall transmission delay. We start from the transmission delay for packets as network protocols' prominent transfer units. The transmission delay for a packet is the result of the sender's and receiver's processing overhead, the link bandwidth, the transmission link's propagation delay, and the traversed routers' processing time [72, 113]. In high data rates, the sender and receiver's processing overhead in creating, sending, and receiving packets are one of the main reasons for performance degradation and increased delay. In addition to that, as shown in Eq. 2.1, the network bandwidth and the number of bits that could be sent over a link plus routers per packet processing speed and the link propagation delays also play impor-

tant roles in determining the transmission delay.

$$\begin{aligned} Transmission\ delay\ for\ a\ packet = sender's\ and\ receiver's\ processing\ delay \\ + total\ path\ propagation\ delay + total\ router\ processing\ delay \\ + \frac{packet\ size}{bottleneck\ bandwidth} \end{aligned} \tag{2.1}$$

In order to speed up the network, one needs to seek solutions that help to reduce the delay caused by every single element in Eq. 2.1. Sometimes, though, one element turns to become the most significant speed bottleneck. For instance, when the link capacities where much lower than today's gigabits per second the bottleneck link bandwidth was the single most important delay component in Eq. 2.1. However, with the increasing link capacities, the propagation delay and other processing related delays appear to gain more importance in determining the speed. For example, results in [28] show that today when comparing the HTTP page load times, increasing the bandwidth is less important than reducing the latency. Figures 2.1 and 2.2 illustrate some of the results reported in [28]. Figure 2.1 shows the effects of increasing the bandwidth and figure 2.2 illustrates the effects of reducing the RTT on improving a HTTP page load time. These figures suggest that for every 20 ms RTT reduction, there is a linear improvement in HTTP page load times. A similar trend does not exist with the bandwidth improvement. Doubling the bandwidth from 5 Mbps to 10 Mbps does not have any significant effect on the page load times. In the next section, we describe some of the physical properties and their improvement trends affecting the delay components in Eq. 2.1. In section 2.1.3, we then explain the relevant design and implementation trends that affect the resource utilization and speed in the network.

2.1.2 Physical properties

The delay components described in Eq. 2.1 are affected by two obvious physical properties: *bandwidth*, and *latency* , as well as two other properties that might be less obvious: *processing speed*, and *memory capacity*. Bandwidth defines the size or the width of the data transfer channel and therefore, affects the data rate and processing speed in the source and receiver, the routers, and the bottleneck link. Latency, on the other hand, defines the total propagation delay on the path, and also affects the time that it takes to read the data from the memory either at the source or

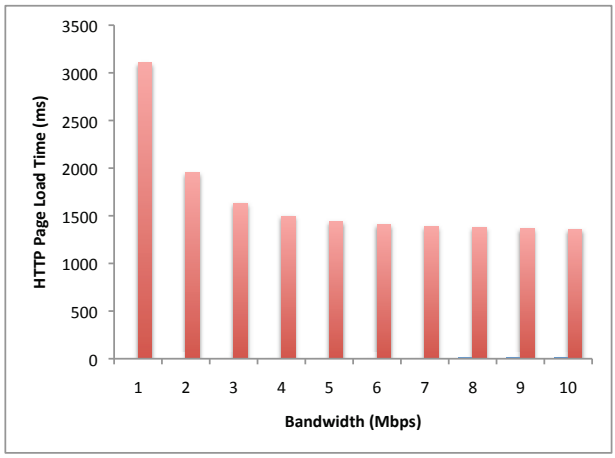

Figure 2.1. Improvement of HTTP page load time with increasing bandwidth and constant RTT of 60 ms (data from [28])

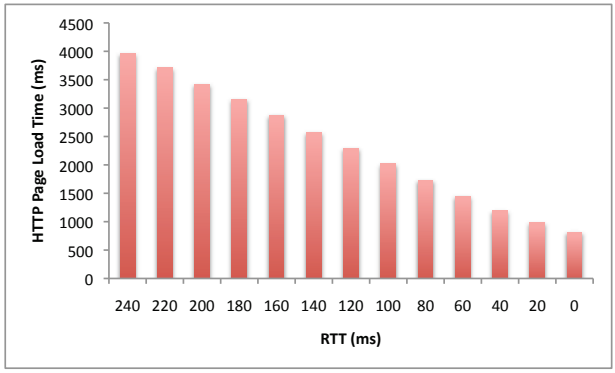

Figure 2.2. Improvement of HTTP page load time with decreasing RTT and constant bandwidth of 5Mbps (data from [28])

receiver or in the routers. Less obvious physical properties affecting the delay include the processing speed and memory capacity. The existing processing power on different routers and end-points determines how fast the data could be processed and transmitted in the network. There is a dramatic speed difference between a network element that for example uses a CPU with a clock rate of 100 MHz versus another instance that uses a 5.5 GHz CPU [84]. Similar to the processing speed, the memory capacity is another important factor affecting the delay equation. Memory capacity could either be used for queuing and thus increase the delay [67], or it could be used for caching and therefore reduce the propagation delay [32].

For many years, there have been discussions regarding the speed and performance improvement ratios of the physical properties listed above [72, 113, 125]. These discussions are centered around the fact that the speed and performance of different elements of communication systems do not remain the same but they change and they change at different rates.

In 1965, Gordon Moore described a trend in which the number of transistors on integrated circuits doubled approximately every 18 months [114]. This has become famous as *Moore's law*. Many components of the communication system have followed a similar trend of improving 4 times per 3 years as described by Moore's law. However, not all the components of the communication system follow the same ratios as Moore's law. In addition to that Moore's law is not guaranteed to hold forever, especially when different components reach their improvement limits.

Starting from the improvement ratios of *bandwidth* and *latency*, the statistics suggest that one is improving much faster than the other [125, 72, 23, 79]. In a 2004 paper [125], Patterson argues that across different technologies bandwidth has been improving much faster than latency. This trend has also been observed and argued in [72, 113]. We use the numbers mentioned in the Patterson's work [125] to illustrate a summary of relative bandwidth and latency improvement ratios across different technologies in figure 2.3. The figure emphasizes that bandwidth improvements across different technologies are higher than latency improvements. For instance, the network bandwidth in particular seems to have followed Gilder's law [11] for quite a few years. The network bandwidth has improved 3 times every 4 years between 1995 and 2005 [113]. The bandwidth growth rate has slowed down a little bit since then but it

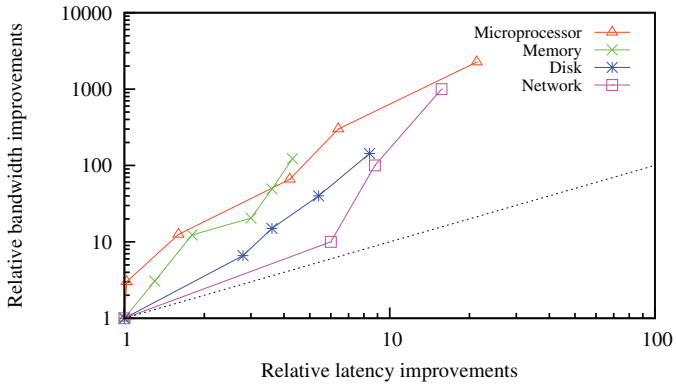

Figure 2.3. Log-log scale bandwidth vs. latency improvement ratios (data from [125])

has kept its exponential growth. Latency on the other hand has lagged behind the bandwidth [125]. For many years now, the network latency has only improved less than 10% a year [113, 125] through applying different techniques like using new form of encodings as in DSL networks [156]. We observe the same trends of latency lagging bandwidth on the memory side. During the past few years the DRAM access latency has not improved much while the DRAM access bandwidth has kept on increasing.

Moore's law affects both the bandwidth and the latency. However, Moor's law has greater and more positive effect on bandwidth than latency . For example, increasing the speed and the number of transistors per chip affects both the latency and the bandwidth, but in different ways. Having higher number of fast transistors per chip helps to improve the bandwidth. Faster transistors help the latency as well, but at the same time, having higher number of transistors per chip results to increasing the average distance between different transistors on the chip. Increasing the average distance could then increase the latency. As across different technologies, increasing the distance adds to the latency. It is simply because we are not able to achieve a speed higher than the speed of light over long distance paths [72]. In a similar manner, many other components that help to improve the bandwidth hurt latency at the same time. For example, adding buffers to the routers is desirable as it is useful to more efficiently benefit from the available bandwidth but it also increases the latency, creating the phenomena that today is identified as *bufferbloat* problem [67].

Storage and *memory capacity* are other important components affecting the speed. Storage and memory capacity have kept their high improve-

ment ratios for many years. An old rule of thumb states that the storage capacity increases hundred times per decade [72]. Nowadays, it seems that the improvement ratio of DRAMs, as a particular memory type, have slowed down and it has become questionable if DRAM capacity could be improved anymore [79]. But anyhow DRAM capacity is still doubling every 2-3 years [79], while other new memory technologies are also being introduced from time to time[79].

Similar to the memory capacity trends, the *processing speed* has also improved a lot in the past few years. A single processor's speed has doubled every 18 months between 1986 and 2002 [79, 23]. During these years, performance doubled because the system performance and improvements of a single processor was not dependent on the limitations of the external entities, such as bus I/O speed. However, now this trend has slowed down [23]. Nowadays any performance improvements is dependent on the external factors. The memory wall is one external factor that describes the growing gap between the *speed of CPU* and the off-chip memory's *access latency* [163]: having a slow memory creates a memory wall by making the processors wait for accessing the memory. One would think that having multi-core architectures with multiple CPUs and memory banks could be helpful in this case. However, another external factor to performance improvements is the power wall problem that is created by the increasing heat generated from the transistors [23]. For instance, the increasing number of on chip processors could soon reach its limits as a result of the power wall problem [110, 23, 4].

The varying trends between performance ratio improvement of different system components emphasizes a major phenomenon. The limiting factor for increasing the speed of data transmission operation is shifting from the bandwidth to the latency, for example as mentioned above large average distance in multi-processor architectures creates added latency. With the growing amount of available bandwidth and memory, it seems that we are not too far from the future that Gray predicted in 2000: stop worrying about the bandwidth and memory capacity, and cache everything because latency is the new bottleneck [72].

2.1.3 Design and implementation trends

Increasing the speed is not a new concern. Most design and implementation solutions in the filed of computer hardware, software, and networking have tried to improve the speed before, and many of them had to tackle

latency as one of the most difficult system bottlenecks. However, since as an independent component, latency in itself cannot be improved so much [125], many solutions attack the latency problem by seeking alternative ways of operation and utilizing other available resources such as bandwidth and memory. Here, we take a brief look at some of these solutions.

Node-related trends

At the lowest level in communication systems, different forms of redesigning hardware have been used to improve the speed. Improving the speed of DRAMs and multi-processor designs are good examples of such methods. In the case of DRAM increasing the memory bandwidth has been used to increase the speed and to overcome some aspects of the latency problems.The logic is simple, if a single memory access is costly in terms of latency, it would help if every single memory access is done with reading/writing bigger blocks of data [44]. In case of multi-processors increasing the number of processing components has been used to increase the bandwidth. Therefore, increasing the parallelization and data rate can reduce the waiting time for the processing operations to be complete [23].

In upper layers of the networking stack, combining the hardware and software efforts together appears to be a design trend for some solutions. Transport Offload Engine (TOE) [41] is one example of coupling software and hardware efforts that takes part of the TCP processing to the network interface card and in this way avoids going through the CPU for every single operation. For instance, TOE could be used to offload TCP's checksum computation to the network interface card, and in this way reduce the end-hosts processing delay.

The pure software design and implementation efforts have also targeted increasing the speed in communication systems, when feasible. Many software implementation efforts are concentrated on reducing the delay introduced by sending and receiving end-hosts. This means most of the efforts are centered around improving the transport protocol implementations and packet I/O. NetMap [136] is one of the recent efforts that achieves fast packet I/O by reducing some of the packet processing steps in the source and receiver stack. NetMap [136] for instance reduces the delay caused by the required memory copies to generate a packet. NetMap [136] achieves this by using shared buffers between the kernel and user-space modules.

Network-related trends

Most of the mechanisms mentioned above redefine some mechanisms and utilize the available resources to alleviate increase the speed in a single node. A *resource* in this context and in the remainder of this thesis refers to one the following components: *bandwidth, processing power, and memory*. Obviously, the more resources are available and the more freedom a system has in using them, the easier it is to improve the speed and overcome the latency bottleneck.

However, it is not always up to a node-specific hardware or software solution to easily utilizes different resources and address the speed improvement issues. The overall system architecture and design also plays an important role in determining the speed, and even causing limitations for improving the speed. A network protocol on its own can limit the possibilities of improving the speed. For example in case of TCP, no matter how efficient its implementation is, TCP by design mandates its own delays. TCP's added delay is often the result of the time that a protocol spends in the initiation and also waiting state before sending out a packet. This waiting time on its own is often dependent on the link latency and single packet delay. TCP's small initial congestion window size and its intertwined congestion control and reliability, all contribute to its added delay.

Recent efforts try to overcome the delay caused by the control flow logic through either changing the existing protocols or designing new protocols. TCP Quick start [62] is one of the recent mechanisms that try to improve the speed by introducing incremental changes to the existing protocols. RCP [50] and XCP [91] transport protocols are examples of the new protocols that have been designed with the goal of improving the speed. We will discuss the control flow delay in more details later in this chapter.

When talking about the Internet, designing and implementing the speed optimization solutions are not only dependent on isolated physical properties or protocol details, but they are also dependent on the interactions between different mechanisms and protocols. For instance, in the lowest levels multi-processor designs are not affective, if a router/ end -host software is not optimized for using multi-processors [79]. There are also other examples that emphasize that isolated optimizations may not necessarily achieve their expected outcomes without the overall system coordination. One particular example of this required coordination is the network usage model enforced by various transport protocols. These specific protocols could affect the outcome of all different kinds of isolated optimizations in

the network, as in the following example.

All-optical packet switching [1] has been introduced in early 2000 [113] to increase the processing speed in the routers (as one of the main delay components in Eq. 2.1). However, one of the biggest problems in the all optical packet switching world is the fact that rather big buffers in the routers are required to keep the link utilization high in the current Internet model. The requirement for buffers comes from the fact that most traffic in the Internet uses TCP and TCP's sawtooth behavior mandates having buffers in the orders of $link\ bandwidth \times Round\ Trip\ Time$ [20, 160]. Figures 2.4 and 2.5 show schematic views of this requirement at the time this buffer size was suggested for a single TCP flow [160]: the bottleneck queue drains with a matching speed with a TCP flow. The queue size increases as TCP keeps on sending data and increases its congestion window. The queue starts to drain when TCP sender pauses and reduces its congestion window. Buffers as big as the congestion window are required to keep the bottleneck link busy all the time. However, the optical switching components and especially optical RAM's cannot easily support big buffers. Therefore, it is difficult to use them with the TCP traffic at large scale. New optimizations to the TCP and its behavior or even new networking models [113] might be required to eliminate the software limitations that stand on the way of high speed optical switching.

In the next section, we look at delay components and the type of resource tradeoffs that exist in the Internet as a whole. We specifically look at the most common mechanisms for file transfer operations in the Internet, and describe how an architecture similar to the current Internet could limit the resource utilization opportunities.

2.2 Delay in the Internet

From the application's perspective the transmission delay in the Internet depends on at least two metrics: first, from where a file is transmitted (affecting the delay components in Eq. 2.1), and second, the speed of the transmission related protocols. Both these metrics are directly affected by the architectural design and the implementation of different protocols.

[1] Note that all-optical packet switching is different from today's optical circuit switching and optical burst switching. The closest concept to all-optical packet switching in the current networks is optical packet networking. However, unlike in all-optical packet switching, in today's optical packet networking signals are converted to electrical before switching and processing.

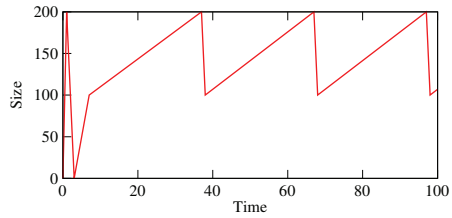

Figure 2.4. Schematic view of TCP sawtooth behavior

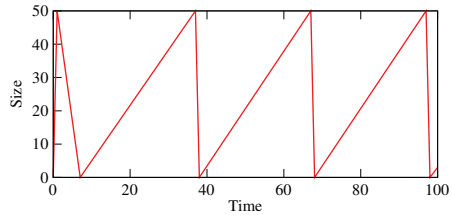

Figure 2.5. Schematic view of the queue size matching the TCP sawtooth behavior

Alleviating the delay then depends on how efficient the architecture and the implementation of different protocols are, and how easy they are to improve. To understand these better, in this section we try to answer the following questions in the context of the TCP/IP protocol suite:

1. What defines the distance that a file has to traverse in the network and how could it be improved?

2. What defines the control flow delay and how could it be reduced?

To be able to answer the above questions we first identify the basics protocol mechanisms that are used during a file transmission operation. Figure 2.6 illustrates a simplified version of sample web-based file transmission operation in the Internet. In the the first phase of this operation a client application requests a DNS server to resolve the domain name part of a web *URI* to an IP address. In next steps, after the IP address is obtained, one application end starts to communicate to the other application end residing on another node. IP-based routing and forwarding is done to navigate the packets within the network. Different protocols and policies are used for filling up the routing tables and making the routing decisions. All these operations affect speed, enforce a specific form of resource usage, and add their own delay to the overall user-perceived delay as we describe in the following.

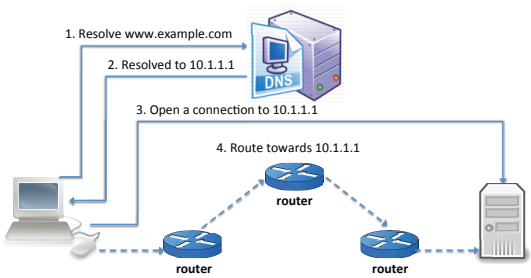

Figure 2.6. An example file retrieval operation in the Internet

2.2.1 Distance

The file transmission delay depends on the distance from which the file is served. In the TCP/IP model, the applications need to identify which nodes they want to connect to before they are able to operate on the network. Thus, an applications either needs to know the node that is of interest to it, or it needs to retrieve this information through a *lookup* operation. The choice of the end node that serves the file determines how far away a file is located in the network. The *routing and forwarding* module at the IP level then specifies the path to reach the file location. To be able to understand the resource usage implications of these mechanisms and the effect that they have on the user-perceived delay, we take a closer look at the *lookup* and *routing and forwarding* functionalities.

File Location: Application level Lookup

In the current Internet, the most prominent method of node lookup is DNS. The Domain Name System (DNS) is a hierarchical naming system that associates various pieces of information with domain names. DNS is mainly used to translate (resolve) the domain names into the IP addresses needed for identifying different nodes. More than one IP address could be associated to a domain name. The DNS hierarchy is divided into zones and each zone is served by a name server. The top of the hierarchy is served by the root name servers. Each domain has at least one DNS server that publishes information about that domain. Domain names consist of different hierarchically structured parts that are used to find out which

name server should be contacted when resolving them. The transmission delay that is experienced by the application depends on both the distance of the DNS servers used for resolving a name as well as the location of the node that is resolved by the DNS servers.

Usually an end-host's web browser is configured to use a default DNS name server and resolve domain names to IP addresses. The result of this operation ultimately defines which node and IP address should be contacted by the application. DNS name servers are not by default designed to take into account the transmission delay and the latency that are associated to each IP address/node. Therefore, it is probable that an IP address returned by a name server results in high transmission costs, for example, because it is located far away from the client application. In this case the application and lower layer protocols and mechanisms simply do not have any control on choosing or changing the returned IP address afterwards.

Today, many DNS level solutions try to optimize the DNS name resolution and lookup. For instance intelligent DNS methods that are designed for CDNs take into account the node's location for finding the *closest* IP address to the requesting node. The *closest* IP address could be defined based on different metrics such as requester's IP address, e.g., [6]. CDNs perform DNS redirections using a hierarchy of DNS servers to translate file request in that CDN's customer network to a nearby CDN server [155]. In this way, CDNs manage to reduce the distance that a file item has to transfer to reach the client.

More radical forms of naming and mapping is also suggested in data-oriented and flat-name based resolution models such as DONA [93]. DONA uses a route-by-name approach for name resolution in which the file names are kept in Resolution Handler nodes across the network. A request to resolve a file name is then routed through the Resolution Handlers to resolve the name to the nearest node that contains a copy of that file item [93]. In this model, unlike in the simple DNS, the closest possible Resolution Handlers are approached and additionally the name resolution often results in finding the closest IP address.

Other less radical replacements for DNS include using BitTorrent trackers [130], and peer-to-peer file sharing mechanisms [75, 74]. Most of these application-specific location identifiers fail in finding the closest nodes in terms of latency and delay. This happens because these application level solution do not have any lower level information about the physical net-

work topology. However, there are also some mechanisms that could provide network related information for these kinds of application level solutions. Application Level Traffic Optimization (ALTO) [135] is one example that defines an API through which topology and infrastructure hints are requested by the application layer and delivered by the network layer.

File Path: Routing and forwarding

In addition to the file location, the choice of the path also affects the overall transmission delay. The choice of the path in an IP-based network depends on the IP layer routing and forwarding modules. IP routing and forwarding provides support for transferring packets that have pre-identified source and destination addresses in their header. Within the Internet, a collection of IP addresses or IP prefixes that are under the control of one operator are referred to as Autonomous System (AS) [77]. There are different types of routing protocols that are used within each AS or connect different ASes together. For instance, OSPF [115] is a link state routing protocol [83] that is used within a single AS. OSPF works by constructing a connectivity graph of the network in each node, showing which nodes are connected to which other ones. The routing table is then calculated by finding the shortest path in the connectivity graph. The shortest path calculation is based on the link costs across each path which includes the available bandwidth.

BGP [166] is another routing protocol in which path and network policies are used to connect different ASes. These policies define the connectivity information that is advertised around the network. Policies define which paths and prefixes should be advertised further in the network. Routing tables are filled based on constant metrics such as bandwidth.

One of the main benefits of the IP routing is the fact that the cost functions used in IP provide a *stretch* which is close to optimal for finding the shortest possible path between two different nodes. The path *stretch factor* is measured by the ratio between the length of the path traversed by a packet and the length of the shortest path between its source and its destination. The stretch factor usually defines how efficiently network resources such as link capacities are used. At the same time the stretch factor could also be important in defining the propagation delay associated with a path. This is especially important when comparing the IP routing to other forms of routing based on flat labels [35, 149].

In addition to being able to find paths with small stretch factors, IP rout-

ing protocols also share another important property: the cost functions used by different protocols do not adapt or are slow to adapt to the dynamic conditions such as increased traffic. It means the choice of a route is pretty stable, even with dynamically changing transmission costs. In most IP routing protocols, the routing table is filled with at least one possible route choice at a time, and this choice is not necessarily optimized for latency and delay in all situations. If the network condition and transmission costs change, this change might not be reflected in the routing table. This is either because routing updates are not propagated quickly enough or because the routing cost function does not include that specific condition such as increased traffic. Therefore, the route calculation model that is used by different IP routing protocols remain relatively rigid towards leveraging other resources (routes). Depending on the context this might be considered a positive or a negative property. It could be positive as the routing choices remain rather stable in the network, and it could be negative as the network resources might not be used in an optimal way. Today traffic engineering methods are used to address some of the resource optimization issues that could not be addressed by IP routing(e.g. [164]). Although traffic engineering methods are commonly in use today, they usually only respond to substantial events in the network. Additionally, they usually also introduce an extra level of complexity into the network.

Many of the new network architecture proposals that require a radical change to the TCP/IP protocol suite target the rigidness of the IP routing protocols. Proposals such as NDN [8] aim at new routing protocols, which are rather flexible towards dynamic conditions such as increased load. Some of these proposed routing protocols even combine a DNS-style lookup infrastructure to the routing module [8, 88]. This combination is likely to provide better resource utilization opportunities and improve speed. However, as NDN and most similar solutions remain to be research proposals for now, their success in optimizing network resources with the help of flexible routing has yet to be evaluated.

2.2.2 Control flow delay

The overall transmission delay is not only affected by the file location and the path to reach the file but it is also affected by how quickly a protocol is able to transmit the whole file. This is dependent both on the file size and the operational delay caused by the the control logic of different protocols.

The control flow delay is partly caused by out-of-band operations such as DNS lookups and partly by the application and the transport protocol. The transport added delay is especially significant for large files, and is of interest to us in this section.

In TCP the delay is associated to the connection initiation procedure and reliability, ordered byte delivery, as well as congestion control algorithm. TCP's 3-way handshake is required to negotiate the required parameters to setup a connection between two end hosts. A 3-step tear down procedure is also required in TCP to end a connection. Connection setup and tear down are needed to help reliability assurance in TCP, but with the tradeoff of added delay. This added delay is not significant for big file transfers but it could cause problem for short file transfers. If a file could be transferred in less than 2 or 3 round trip times, dedicating one whole round trip time for connection initiation is too costly. In these cases using a transport protocol that does not need connection initiation seems a better choice than using TCP. There are, although, some new mechanisms such as TCP Fast Open [131] that go around this problem by transferring data in the SYN packets exchanged during the connection initiation phase. Defining reliability as part of TCP can also create delay during a file transfer operation. This delay depends on the Retransmission Timeout that determines how long a TCP sender has to wait before it resends the lost packets. Long Retransmission Timeouts could add unwanted delay to a file transfer operation.

The other part of the control delay in TCP depends on the time that a receiver has to wait to be able to receive a range of packets in order. Out-of-order delivery is not supported in TCP. For example, no matter how fast the 200th byte of a stream arrives at the receiver, it is not delivered to the receiving application before bytes 0 to 199 have also arrived there. This adds an unnecessary delay to the file transfer operations that are not sensitive to loss or out-of-order delivery. One solution to this problem could be using TCP minion [119] instead of unmodified TCP. TCP minion adds a few socket options to the normal TCP implementation that allows out-of-order send and delivery in TCP.

Some of the added control delay issues in TCP is addressed by newer transport protocols such as QUIC [140]. QUIC does not require a 3-way handshake before sending data, and also tries to reduce the possibility of congestion and packet loss by pacing the packets. Additionally, to alleviate the packet loss and the time spent to decide to re-send the lost packet,

QUIC supports forward error correction in the packets.

In addition to the delay caused by the reliability considerations and ordered delivery, significant parts of the transport delay depends on the congestion control algorithm that is used by that protocol. A good or a bad congestion control strategy could make a big difference in estimating the available bandwidth and determining the speed of data transmission. For example, if the available bandwidth is 1 Mbps and the congestion control module is already aware of that, then it can right away start sending data at the rate of 1 Mbps. Obviously data transmission would be much slower if the congestion control operation starts conservatively and only gradually probes the bandwidth. In the next part we take a look at congestion control in the current Internet.

TCP's Congestion Control

The congestion control algorithm in TCP uses data packets and acknowledgments to implicitly collect information. TCP uses this information to learn about the resource availability in the network and make decisions on how much of these resources to use. TCP treats the network as a "black box", with all the intelligence located at the end hosts. In this model the congestion control module does not receive explicit resource availability information from the network and therefore it operates based on different assumptions about the network. For instance, a TCP end point assumes that any packet loss in a connection is the result of congestion. In the following, we explain some of the design and implementation assumptions that shape TCP's congestion control model.

TCP's congestion control algorithm was designed a long time ago [85] when the network was all wired and 10Mbps links were just introduced. This means some of TCP's original resource probing design choices may not necessarily match the recent resource availability trends in the network. Some of these design choices hinder TCP's performance especially regarding speed. For instance, TCP's inability to quickly benefit from increased network bandwidth is one of the main concerns for many researchers [91]. The primary reasons for this inefficiency is that the congestion control module is rather conservative and slow in probing the bandwidth. TCP uses send and receive windows to define the boundaries for transmission speed. The window size shows the amount of data that could be in transit in the network. TCP uses the sent data and their acknowledgments as a sign of bandwidth availability in the network. The

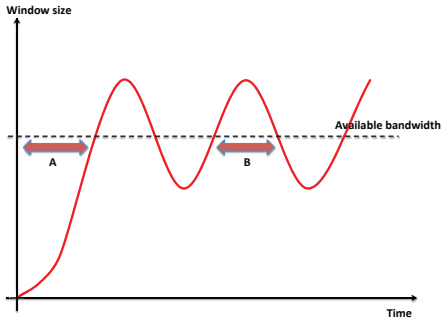

Figure 2.7. The congestion control related added delay

congestion window size at maximum reflects the bandwidth delay product where bandwidth specifies the bottleneck link capacity on the path and delay is the *Round Trip Time (RTT)* between the sender and receiver. TCP's congestion control algorithm is slow as it starts with a very small window size of 3-10 MSS (Maximum Segment Size [127]) for bandwidth probing. The congestion control algorithm then follows an additive increase, multiplicative decrease model in which every acknowledgment results in a maximum window increase of one MSS.

In TCP the congestion control is done on per flow basis. Each flow has its own share of available bandwidth and seeks optimal ways to probe the network about that. The optimal window size for each flow equals that flow's share of bandwidth times the delay. The available bandwidth for each flow can change dynamically depending on the status of other flows that share the same bottleneck link. Therefore, each flow has to constantly probe the network to find out about the available bandwidth. This is a slow operation especially in high latency environments, because every acknowledgment (the implicit probing signal) only arrives to the sender after a *Round Trip Time (RTT)*. As can be seen in figure 2.7 TCP spends a lot of time probing the available bandwidth with this method. In the figure the TCP sender spends a significant amount of time in phase *A* just to get an initial estimation of the available bandwidth. The algorithm then spends a lot of time in consequent phase *B*s trying to estimate if the available bandwidth has increased or decreased, and thus, keep the traffic rate at balance with the available bandwidth.

Researchers have come up with many different mechanisms to reduce file transmission delay in the context of TCP's congestion control algo-

rithm. There are at least two different speed improvement angles that target TCP 's congestion control operation: the first one is sharing bandwidth availability information among multiple flows [24, 13], and the second one is increasing the size of the initial congestion window [14, 52]. Some of these mechanisms are described below.

TCP creates per flow basis boundaries between the resource availability information that it gains from the network. Assume there are two consecutive flows that share the same sender and receiver, there is a high chance these two flows also share a common set of network resources. However, with normal TCP each of these two flows have to go through the resource probing operation independently. If one flow is already aware that the congestion window could be increased to 32KB it does not share this information with the other flow that is just going to start resource probing. This kind of information sharing, if possible, could accelerate the speed that different flows estimate the network resources and start to use it at maximum rate. Mechanisms such as congestion manager [24] use cross-flow information sharing to speed up the bandwidth estimation procedure and reduce harmful competitions to gain network resources.

Cross-flow information sharing, although good for resource optimization and speed, has remained an abstract concept. Meanwhile TCP's congestion control related added delay is still an open issue. A 2009 study by Qian et al. [129] shows that some TCP flows have been violating some of TCP's original assumptions such as initial window size of 3 [18] probably to overcome TCP's slowness. In this regard some researchers have argued that since most of the web objects transferred over TCP are as small as 16 KB [14, 133], then increasing TCP's initial congestion window size could indeed reduce the effects of latency and increase the speed [14, 52]. The latest versions of Linux Kernel already implements initial congestion window size of 10 (Ĩ5KB), reducing the time spent in phase A in figure 2.7. However, the long term effects of increasing the initial window size form 3 to 10 remains unknown.

In addition to the general design choices and the initial parameter settings that could make TCP slow, there is also another congestion control metric that affects TCP performance, and that is the buffer added delay. This metric depends on the way that TCP utilizes buffering resources. If there is available buffer in the network TCP keeps on sending data until the bottleneck buffer gets full. Because of its logic, TCP needs a big enough buffer to prevents link under-utilization. The trouble is that be-

cause of the diverse nature of the network estimating the correct buffer size is not easy. Therefore, as memory gets cheaper more and more buffers is thrown at the switches and routers to avoid any possible chance of link under-utilization. However, as the buffer size grows the added queueing delay becomes troublesome. In a recent study Krishnan et al. [96] report that in the Google CDNs, latencies are relatively high, and in 40% of the cases round trip times exceed 400ms. This report also argues that adding more CDN nodes would not always help to reducing the delay because high latencies are often the result of excessive queuing rather than not having nearby copies of the content.

One extensively studied approach to address the added queuing delay is using Active Queue Management (AQM) methods in the switches and routers. With AQM methods such as Random Early Detection (RED) [64] the average queue size and other metrics are used to control the queue growth rate before the queue size reaches to the maximum buffer size. More recent AQM works [117, 123] use the packet-sojourn time in the queue to identify the possibility of excessive queue build up and signaling it. Similar problems have also been studied in the context of different datacenters [16, 17]. For example, the work done on Data center TCP (DCTCP) [16] tries to benefit from shallow buffer space in switches and early congestion marking, which in turn results to reduced delay.

Other solutions to control the buffer growth rate is implementing delay-based congestion control algorithm like TCP Vegas [31] could help. In TCP Vegas [31] packet RTTs are observed carefully and dramatic RTT increases are often prevented by reducing the sending rate just in time before a radical RTT increase happens.

Another solution for reducing the added buffer delay in the Internet is reducing the expected TCP buffer size as a whole, as suggested by [20]. In their work Appenzeller et al. [20] suggest that with the big number of flows that go through every router, a much smaller buffer size would be enough to keep the link utilization high compared to an optimal size of a single flow's bandwidth-delay product.

Congestion control beyond TCP

The network path between the client and server is opaque to TCP and it creates its own share of the resource management problems. TCP's congestion control has been built based on many assumptions about the network. From time to time, the network invisibility as well as the specific

TCP assumptions become problematic for resource optimization.

Many of the TCP assumptions could not be easily changed because the network is opaque. For instance, consider the case for the cross flow congestion management [24] that we mentioned earlier. While promising to provide a more optimized way of sharing the bandwidth availability information, these methods are not exactly successful in categorizing which flows should share the bandwidth information. The work done by Akella et al. [13] shows that even flows that share same end-points might not share the same bottleneck.

Aside from TCP specific optimization, there have been many proposals that have tried to replace TCP and come up with better forms of congestion control [50, 91]. Most of these proposals aim at optimized resource probing by getting direct help from the network. These protocols often require adding more capabilities to the routers to get direct bandwidth availability report from the routers and make the resource probing operation quicker [50, 91]. This protocols try to reduce the time spent both in phase A and *B*s in figure 2.7. For instance, XCP [91] requires all the routers to be able to calculate per flow fair share of traffic in different time periods and report back to the end-points, therefore, reducing the time spent both in initial bandwidth estimation (phase A in figure 2.7), and in continues bandwidth probing (phase Bs in figure 2.7) However, at least until now none of the congestion control modules that have counted on the wide-scale network support, have got any chance of being deployed. The reasoning is simple: most these new protocols such as XCP [91] and RCP [50] require significant changes in all the routers in the network, but deploying such changes is not easy.

2.2.3 Resource utilization

In addition to what we have explained above, distance and control flow delay could also be analyzed from a more general resource utilization perspective that is often ignored. It is clear that to reduce the delay it is important to choose and utilize the best available resources at any given moment in time. Unfortunately the possibility of choosing the best available resources at any stage during a file transfer operation is rather limited in the TCP/IP model. The limitations are caused by at least two different reasons as we will describe shortly:

- Resource binding happens before a file transfer operation starts.

- Protocols below the application layer are not designed to benefit from the available network resource other than for supporting connectivity.

As we have discussed before, end-point binding in TCP/IP happens before the file transmission starts, and it could not be changed during the file transfer. Even the path towards an end-point usually only changes if a link or a router IP address is not reachable through the same interface anymore. If bandwidth, memory, or processing power in a network path starts to get overloaded, the most probable reaction is just to slow down the transmission rate. There are limited chances to use other available network resources that are located nearby but are not part of the currently congested path or an overloaded end-point.

The limitation in switching to less loaded (re)sources during a file transmission operation is partly caused by the IP routing models and partly by the end-to-end logic forced at the transport level. We have already discussed the limitations caused by the IP routing, but the role of the end-to-end logic needs more explanation. From the early days of Internet, the transport protocol has been defined to support end-to-end operations [145]. Each transport connection is identified based on its end-point IP addresses and port numbers. There is a strong correlation between specific end-to-end identifiers and the way that many of the transport operations are defined. For instance, consider reliability as one of the most important functions in TCP. In TCP reliability is a function that strictly depends on a specific end-to-end connection. Bytes are only meaningful and deliverable if they are identified to be part of a specific end-to-end connection. This means changing a connection identifier and consequently changing the enforced end-point and path bindings during a file transmission operation, breaks TCP's reliability logic.

Except early bindings there is another limit in freely choosing the best available network resources in the TCP/IP model. This relates to the fact that in TCP/IP network resources mainly includes only links. Other available resources such as in-network memory and processing power do not have significant importance when talking about network resource availability. The reasoning is simple, TCP/IP is only about connectivity, and it has been designed at the time when bottleneck link bandwidth was the most valuable/ limiting network resource. In TCP/IP the available processing and memory resources in the routers are only used to achieve end-to-end connectivity through routing, forwarding, and buffering. When

talking about delay reduction and resource utilization these use cases are not necessarily enough. Assume that the network could provide lots of processing and memory resources, much more than what is needed for routing and forwarding operations. But at the same time upstream links are extremely busy to forward similar packets over different connections. In this case, TCP/IP can not do much with the available memory and processing resources to alleviate the load on the upstream link. It can only slow down the sending rate over the upstream link and use the available memory to buffer the packets. A different network design on the other hand, could have for example used the available memory and processing power to combine the similar packets into one, and save the bandwidth while controlling the overall file transmission delay. We discuss more details regarding such mechanisms in chapter 4.

Being able to switch to less loaded resources during a file transmission operation, and being able to use the available network resources more freely, if implemented properly both could result to the file transmission delay. However, many things need to be changed in the TCP/IP model before such capabilities could be added to the stack. In the next chapter, we discuss a new networking paradigm that might create less restrictions for increasing speed.

2.3 Summary

In this chapter we have discussed various elements affecting speed and the user-perceived delay in the current Internet. We have argued that both physical properties, and design and implementation choices are important in determining speed. Among all physical properties latency appears to be the new bottleneck in many parts of the system, and improving speed requires reducing the latency. However, since physical laws limit the possibility of directly improving the latency in many aspects, it becomes more important to reduce the delay through applying different design and implementation choices that alleviate the role of latency.

We have discussed the architecture- and design-related delay in the TCP/IP protocol suite. Distance and control flow delay were among the discussed metrics, which affect the delay depending on how many network operations need to be done and in which distance.

We have argued that alleviating the overall user-experienced delay in the Internet could be possible by optimizing the usage of different avail-

able resources in the network. If a certain resource is slow, using other available resources could help improving speed. If a certain design pattern enforces binding to certain resources and thus results in additional delay, then a new design that relaxes those resource bindings could reduce the delay. However, we discussed that many design and implementation choices, especially the ones made for the networking stack in the current Internet, limit the possibility of optimizing different resources to improve speed.

3. Towards the future: Information-Centric Networking (ICN)

"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."

(Albert Einstein)

3.1 Information-Centric Networking (ICN)

A significant share of Internet usage is about distributing and transferring content. However, the Internet architecture and the TCP/IP protocol suite have been designed to mainly support connectivity between end-hosts. The mismatching goals between the network and the applications using it can limit the possibilities to improve the efficiency of the content delivery operations. For example, as speed becomes an important metric in defining the application's efficiency in delivering the content, the underlying network architecture might remain rather indifferent to this metric and therefore, reduce the chances to improve application efficiency.

Information-Centric Networking (ICN) [88, 126] is a new networking proposal concentrating on content delivery. ICN aims at improving the availability, failure resilience, and security of content delivery within the network [88]. As can be seen in figure 3.1a an ICN API passes the content names to the underlying network, and the network directly routes and responds to the requests for specific content names. The same name could be used to store and access the content in the storage nodes across the network. As we will explain later, being able to access the network and the storage with the same API and with the same name that is understood by the applications could result in unique opportunities for better content delivery. This is rather different from the IP network [54, 53] in which only IP addresses are understood by the network, and the network API is

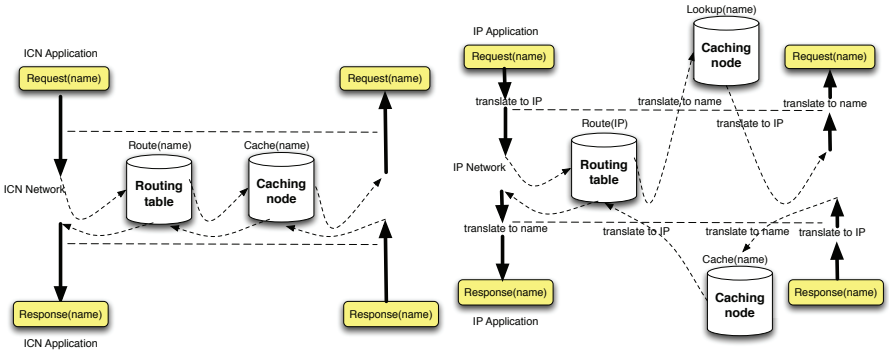

Figure 3.1. Schematic view of content access in an (a) ICN network on the left, (b) IP network on the right.

independent from any other API in the system, figure 3.1b.

Distributing and accessing the content is the main goal in ICN. ICN has more emphasize on accessing content using all available resources, and less emphasize on connectivity and fixed bindings to certain resources [118, 126]. Thus, compared to the TCP/IP model ICN supports more in-network optimization opportunities for content delivery. For example, as we will discuss later, ICN simplifies network wide application-independent caching that could be used to reduce network congestion, and consequently to improve the speed.

There are many different components in ICN and therefore many concerns regarding their design and implementation. However, our main interest are the principals and components that deal with the speed and resource utilization in the network. In this regard, here we describe the existing ICN concepts that could be used to target TCP/IP's two main resource utilization problems: early bindings, and the resource usage logic.

3.1.1 Bindings

Information Centric Networking is about named data items. The only enforced early binding in ICN is the binding between an information item and its name. This binding might be defined to be path/ end-point/ resource-independent. A name could be anything from an IP address to a hashed identifier [93, 9, 10] to a human readable identifier [88, 8]. The network infrastructure is expected to provide the routing and forwarding support for finding and routing these names in the network. ICN proposals like NDN [8], PURSUIT [10], and NetInf [9] study different forms of naming, name resolution, routing and various other concepts related to

the network infrastructure.

The main form of communication in ICN is consumer-oriented. The consumer of a data item requests that item by name and the network is expected to deliver that item to the consumer. There are different ways that this logic could be designed and implemented, for example through hop-by-hop lookup and forwarding of content names, or through looking up the name in a centralized database and finding the content location. Here, we describe some of these different design and implementation choices.

Naming

Different ICN proposals have different views on what is expected from a name. In most designs, data names that are passed through the API share the common property of being location-independent, but their structure varies from one design to another. The choice of a naming structure depends on different factors including human readability, and security. The data names that are revealed to the network could be human-readable or non-human-readable (or a combination of both). Human-readable names are easy to memorize for the users and do not need an additional resolution phase [8]. Human-readable names often have varying lengths, and sometimes they require the existence of a naming assignment authority that will handle various issues regarding who is authorized to create and/or modify a content item with a certain name. Non-human-readable names are usually of constant size and can be derived from the human-readable names that are used at the application level. Non-human-readable names, which are used within the network could for example be resolved through hashing the human-readable name or by using a directory service similar to DNS [10].

As mentioned earlier, an ICN name in contrast to an IP identifier is often considered to be location-independent [88, 10]. An API binding in ICN does not necessarily enforce connecting to a certain end-point. In this environment, many IP-based security solutions that authenticate, encrypt, and secure connections based on end-point identities become difficult (if not impossible) to use. Therefore, it is preferable if authentication, encryption, and security of a content transfer operation becomes embedded into the name or the data item itself [88, 150]. In this regard, different ICN proposals suggest either direct or indirect binding between the content of a data item and its name. In a direct binding, the name or part of the name is cryptographically derived from the data item itself [93, 10].

In an indirect binding, the name securely binds to something or someone who can vouch that the received data item is the correct one matching the requested name. For instance, if the data item contains some information about the data source then the receiver that suspects data authenticity can contact the specified source to make sure that the data is correct. One example of this kind of indirect binding is the binding to the public key of the item owner and signing the combination of a content item and its name [150].

Advertisement and Lookup

Similar to the DNS concept in the Internet, named item *advertisement* in ICN creates state in the network indicating where each named item could be found. Data *lookups* then utilize this state in order to locate the desired item. Advertisement and lookup handling may be logically *coupled* or *decoupled* to the routing protocol of the underlying architecture. Coupled advertisement/lookup and routing means that the advertisement/lookup messages directly shape and follow the routing table entries within the network [8]. Decoupled advertisement/lookup and routing means that there is an additional advertisement and lookup handling layer on top of the underlying network, independent of the network structure and the routing table entries [10].

The decoupled advertisement/lookup and routing enforces some forms of early binding. This model requires the consumer to first go through the advertisement and lookup phase, retrieve some location-specific identifier and then use that identifier to access the actual data item [10, 89]. In simple terms, the decoupled advertisement/lookup and routing method could even be considered an equivalent to just replicating content in close-by caches and servers in IP network, e.g. as in CDNs [122], but at broader scale and with many more optimization opportunities below the application level.

Routing and Forwarding

Different ICN proposals approach routing and forwarding from different angles. In some proposals like NDN [8], lookup, routing and forwarding are coupled. Names fill in the router entries and the named data items get routed and forwarded hop-by-hop from one router to another. Some other proposals like PURSUIT [10] do not enforce the lookup, routing and forwarding to be coupled. For instance, PURSUIT's LIPSIN model [89] allows an efficient form of fast forwarding with encoded source-routing

headers.

The important difference between the hop-by-hop methods and the LIPSIN model is that in LIPSIN the forwarding identifier is not necessarily the same as the lookup and routing identifier. This means in LIPSIN the lookup and routing operation could be done based on one form of named data identifiers but forwarding could be done based on some other form of identifiers. In this way while the lookup and routing operation can benefit from using complicated content names, the forwarding operation could still be done with simple identifiers and at line-speed. This of course has the drawback that the location/path/destination information should already be included in the packet header, before that packet could be forwarded by the network.

3.1.2 Resource Usage Abstractions

Many of the research efforts on ICN are concentrated on a specific form of naming, lookup, routing and forwarding [10, 8]. Hoping that improving each of these functionalities, either in isolation or combined, could result in significant gains for content delivery. However, from our point of view one of the less emphasized ICN strength points is the form of abstraction that it provides: the form of abstraction that unconventionally reveals more about the application level items of interest to the underlying protocols and not the other way around. Although not totally new, this form of abstraction should allow better forms of resources optimizations that suits the application interests, i.e. [42, 57].

The idea of making some information about the application level data visible to the underlying protocols was first discussed in Clark et al.'s work in [42]. There the authors argue that in order to improve the speed and efficiency of the send and receive operations inside the TCP/IP stack implementation, it would be beneficial to reveal the identity/boundary of the application level data units (*ALF*) to the underlying transport protocol. They investigate the speed improvements that this form of abstraction would bring to the end-host stack implementation, for example through allowing unordered processing of data units in the stack.

Although not explicitly mentioned anywhere, ICN appears to borrow an information sharing concept quite similar to the ALF. Application level data units and/or application level names are not only identifiable to the transport end-points but they are also visible to the routing and forwarding protocols as well as the nodes that reside within the network. Adding

this level of visibility to the network could potentially lead to making wiser choices in choosing the route. However, the more interesting aspect of this abstraction is the new opportunities that it could bring to the network. The network as a whole could become a resource that can freely do anything to quickly serve the application with the named data [126, 88].

With the new naming abstraction, there are at least two new functions that could be added to the network. The first functionality is the capability to directly utilize the available storage and memory within the network. If a forwarded piece of data seems to attract many interests, a transmitting network node can decide to keep a copy of that data item in its memory to directly serve future interests. Additionally, if an incoming piece of data could stay valid and useful for a really long time, then the available memory in the network could be used to benefit from this durability, e.g., by delayed forwarding of that data item in favor of the item that will soon expire. These all, if designed and implemented properly, could be used for better resource utilization as well as speeding up the content delivery. Many of the contributions of this thesis rely on this functionality.

The second functionality that could be added to the network is exploiting its available processing power to create items of interest on the fly. The ICN abstraction can provide the possibility of optimizing the available processing power. As an example take the case of network coding [105]. With network coding the simple forwarding of received packets could be replaced with more sophisticated operations. The forwarding node can start combining multiple packets, name them, and forward them altogether. This could be used to achieve better throughput in the network and improve the speed. When it comes to combining multiple packets together, the bigger the range of available packets is, the chances are higher to be able choose a better combination of packets for the network coding. We do not explore this functionality in more details in this thesis, and we certainly do not want to convince reader to believe in the magical power of ICN in creating all kinds of items on the fly. However, we hope it is clear that ICN abstraction eliminates a big number of restrictions that the TCP/IP model imposes on using different resources including the routers processing power, and thus ICN could create many opportunities for using these resources in more creative ways.

Going back to our discussions in the previous chapter: reducing the effects of latency requires the network to be able to leverage all different

resources more extensively. In this regard ICN abstraction seems to start a promising new trend in which not only bandwidth but also in-network storage and the available processing power could be leveraged. Next, we are going to have a look at some of the challenges that ICN faces.

3.2 ICN challenges: Congestion Control

Compared to the current Internet, ICN benefits from allowing a less restricted form of leveraging available resources. This is mainly because the application reveals to the network what it wants, and the network gets that to the application without binding it to any specific resources. The network can even use this information to optimize its resources and operations. For example, in some NDN-based proposals [8] the forwarding module can take action in stopping the interest packets to be sent to a specific interface, if the forwarding module notices that specific interface is overloaded with data [167].

However, with all its benefits, ICN is not necessarily good in resource management. This is because resource management is not only about knowing *what* are the resources used for and having *control* on limiting their usage, but it is also about having information on *when* a specific resource is used or will be used again, or *when* it is released. For example, even if the forwarding module is capable of stopping the traffic flow in case of resource overload, it still needs to know approximately *when* a resource is going to be overloaded or *when* it is going to be free. A resource in the ICN context is not only the bandwidth, but also the available in-network storage that could be easily used for caching. However, in this section we keep these resources as separate as possible, and only look at the bandwidth usage model under the general assumption that in-network caching is possible. Here, we mainly discuss the issues regarding congestion control and resource (bandwidth) management in ICN.

3.2.1 Resource Probing

Resource probing is done in order to determine the amount of available resources and the maximum amount of traffic that a resource can handle at any given moment in time. In networks in which resource availability information is not already known by the resource management module, resource probing becomes essential. As described in previous chapter, in

TCP/IP the resource probing operation is done only implicitly by the congestion control module and on per flow basis. Each flow has to constantly probe the network to find out about the available bandwidth. Each resource probing request takes one RTT to be complete, and the optimal share of each flow equals the bandwidth delay product at maximum. The probing operation relies on a few basic assumptions: first, the communication channel remains rather stable for the lifetime of a flow, and second, the change in RTT reflects the change in available bandwidth and the queuing in the network.

ICN deviates radically from the common set of assumptions that form the resource management logic in the IP network. First and foremost, the concept of a an end-to-end flow that connects a consumer to a certain source is almost nonexistent in many ICN proposals. This is because the API does not bind two specific endpoints together, instead anyone can respond to a request originated from a consumer. Thus, in some ICN proposals there is no transport protocol [88], and if there is, no end-to-end communication is defined there [21]. The other important difference between IP and ICN is that the communication channel does not necessarily remain the same for the duration of a data transfer operation and can oscillate quite radically. This means the change in RTT does not always reflect the change in bandwidth, and could be the result of a change in the data origin and/or the path that data traverses. Last but not least, in this new networking model there is no guarantee that the bandwidth delay product is a good indicator of the resource availability, as it is in TCP/IP. The bandwidth delay product becomes especially vague, if available caches could be used to time shift the traffic for more than a few 100 milliseconds.

Many early efforts in ICN replicated the resource probing problem and solutions used in the IP network. They try to do the resource probing on per-flow basis and provide a fair share of bandwidth for each flow. In most of these solutions the flow concept is usually created based on content identifiers [37, 143]. The network then can actively participate in the resource probing operation either by dividing the available bandwidth among different flows from the beginning as in HoBHIS [143], or by dividing the remaining bandwidth among the flows that seem to experience congestion as in HR-ICP [37]. Some solutions also explicitly define the resource allocation operation to be a function of a predictable RTT [143]. In these solutions the amount of available bandwidth is divided between

different flows before any overload happens in the network [143].

3.2.2 Congestion Handling

In an environment without end-to-end connectivity as in ICN, a hop-by-hop resource management model might become desirable. A simple option is having a back-pressure style resource management in place [116]. With back-pressure, whenever there is congestion, the congested router sends a message to its upstream routers to stop forwarding data. The upstream router repeats the same thing when it realizes there are no resources available. The chain continues until the feedback reaches the data source. Except having slow feedback loops toward the sender, this method also requires per flow[1] states to prevent false congestion propagation towards the traffic flows that do not go through the congested part of the path. Therefore, the flow-dependent back-pressure logic is not directly applicable to a flow-less[2] ICN model. In order to use back-pressure in ICN an equivalent abstraction of TCP/IP flow identification in the targeted ICN environment. For example, one could use content identifiers as flow identifiers in an ICN network and then apply back-pressure there [143].

Implementing resource management at the end hosts is another option considered in some ICN proposals. In this case, a resource management module resides at the end-host and waits to receive resource availability feedback about the network, before it makes any decisions. In the TCP/IP world it is simple because a feedback message/bits either arrives within the duration of an RTT or the feedback message goes missing as a sign of congestion. However, this logic is not easily applicable in ICN. As discussed earlier, ICN often supports flexible resource bindings and allows the source and the route to change quite often. This kind of flexibility comes at the cost of reducing the overall predictability of the path itself and the estimated RTT. This means the period of time that an end-point needs to wait for a feedback is difficult to predict. Therefore, if one wants to handle congestion through an end-point residing module, then she first needs to address the unpredictability issues.

ICP [36] and CCTCP [144] are among many proposals that apply receiver-

[1] A flow is usually used to identify the relation between a traffic stream and its source and destinations. A flow, therefore, could be identified by its source and destination, or by a unique stream identifier.

[2] We call ICN flow-less because a traffic stream might not be identified by its ultimate source and destination, or by a stream identifier that is unique among many sources and receivers at the same time.

driven resource management and congestion handling in the ICN environments. ICP [36] relies on implicit congestion feedbacks based on the expiry of the retransmission timeouts. Retransmission timeouts in ICP are calculated based on the maximum RTT samples of previous data packets, irrespective of their source of origin. CCTCP [144] is another proposal, which also uses the expiry of the retransmission timeouts as a sign of congestion, but it tries to achieve a more accurate form of resource and RTT estimation by predicting who will respond to a packet request. Most of these works often concentrate on one aspect of ICN and RTT flexibility and leave out the other aspects. For example, the results of both ICP and CCTCP are mainly applicable to the scenarios in which fast resource probing is not important, and/or the range of packets served by each source remains stable for the duration of a file transmission operation. More dynamic scenarios are not the main concern in these proposals. For example, it is questionable how much one can predict the origin of the next packet, if the content in a router cache dynamically changes during a file transmission operation [21].

3.3 ICN Challenges: Privacy

As discussed in section 3.1, ICN benefits from a certain level of transparency between the applications and the network, and provides many resource optimization and speed up opportunities both for the application and for the network. ICN implicitly assumes that the users and applications are happy and eager to share their interests with the network, and the network is a trustworthy infrastructure that not only uses user's interests for good, but it is also able to guarantee no one else misuses that information. This seems to be a naive assumption. Just imagine if during London unrests in 2011, instead of speculating about filtering Facebook and Twitter [3], David Cameron could have only asked UK ISPs to immediately filter out every request or data that contains the word *London*. Giving more information to the network means giving more freedom to the network to control who uses the network resources and how. This could, therefore, result in endangering users privacy.

Privacy is one of those concepts that has no clear definition, and as Jarvis Thomson puts it "nobody seems to have any very clear idea what it is." [148]. But in most contexts revealing any kind of information by the user could mean giving up privacy in some way, minor or major. Any

kind of communication creates its own privacy threats for the participating party. The privacy threats in the TCP/IP network are not necessarily the same as in ICN. For instance, in an IP network where every packet has to contain the sender and receiver's address, IP addresses could be used to track the geographical origin of a packet. This is not true for most ICN proposals where "IP addresses" are not fundamental to the network operation. In this section we provide an overview of different weaknesses that can threaten the users privacy in ICN networks.

3.3.1 Revealing information

TCP/IP provides isolation between the network and application. This has its own efficiency drawbacks but it means the amount of information revealed between the application and the network remains limited and manageable. Encryption and other security measures could be applied on the packets so that IP addresses or in other words the location information remain the only piece of information that can connect a user to a specific transmission at any point in the network. If a user wants anonymity she only has to influence the IP addresses that get inserted into the packet header. ToR [49] for example can help the users to make packets untraceable to a certain location, by interfering with the IP addresses and the routes that each packet takes .

ICN, on the other hand, puts more emphasis on including other information, rather than the global location addressing, into a *data name* inserted to each packet. Although, this avoids revealing the location information just by looking at the packet header anywhere in the network, it could disclose other information that is embedded into the packet header. Here we list some of the application-related information that are included in each ICN data item transmitted over the network.

3.3.2 Communication-related information

Data names are fundamental in ICN networks. Every packet has to contain a certain type of identifier that is understood by the network, and reveals the data name that is of interest to the user. Depending on its structure and format, this identifier could reveal different things about a user. A ICN identifier could be an exact match to what the user have asked for, or it could just be an ambiguous translation of what user wants.

The less ambiguous the relation between a network identifier and the

actual user interest is, the easier it is to interpret this relation. A network proposal that enforces human-readable identifiers in the packets, makes it easier for a random observer to make sense of the data that is transmitted over the network. An ICN proposal that enforces universal one-to-one relationships between a human-readable name and its equivalent network understandable identifier is easier to crack compared to a model without the criteria of universal uniqueness. It is because a universal one-to-one relationship once revealed could not be concealed again. Therefore, when deciding about the format and structure of identifiers in ICN networks, it is important to consider their privacy properties.

3.3.3 Security-related information

A mentioned earlier, security is one of the main motivations behind many Information-Centric networking proposals [8, 10]. The major goal is securing the content instead of the container to overcome the problem of misplaced trust on communication channels in the TCP/IP world [88]. Therefore, in a ICN environment protection and trust supposedly is embedded into the data item itself, rather than being a property of the connections over which it travels. In most ICN proposals, the private content could be encrypted with a content specific key [8, 9, 10].

One of the most important security enablers in ICN proposals is adding the content owner's signature somewhere in the data item [93, 150]. This signature allows for authenticating the content in a network that does not support end-to-end authentication [98].

Privacy of a content publisher could then be threatened, depending on the way that a signature is validated. If there is always a universal authorization party that always associates a publishers real identity to a signature then there remains no way to support anonymous publishing in the network. This is even worse if the publishers real identity has to be directly reflected in every data item, for instance in the data name.

3.3.4 Privacy control points

The information that is revealed in every ICN data item does not always create the same level of privacy threats. The privacy level also depends on the control points that each infrastructure creates and the type of services and functionalities that it supports. In a network in which lookup and forwarding are necessarily controlled by the same player, it is easier

to profile different users and at the same time deny their access to the network.

In many clean slate ICN networks such as the ones proposed in PURSUIT [10] or NDN [8] the basic API is limited to the data name that is of interest to the user application. In these proposals the network itself is also designed in a way that a user would not receive the data item that she has not subscribed for. The user has to subscribe for a data item but she does not have any control over how the network will deliver this item to her. If the network is organized in a way that it keeps track of user location on every hop in the network, then everyone with access to one or some of these tracking records could learn about users location. Because of the API limitations, the user herself cannot do much. The user can neither prevent the network from keeping these location tracks nor she can make the location information anonymized or ambiguous.

The universal caching and storage feasibility in ICN networks can result in its own privacy issues. A study by Lauinger et al. [100] suggest that as a side effect of caching, users privacy could be endangered by revealing their access patterns. Other similar side effects are yet to be discovered in ICN networks.

3.4 Summary

In this chapter, we briefly discussed the general ideas behind the Information-Centric Networking concept. This chapter emphasizes the ICN elements that more directly influence speed and resource sharing compared to the other ICN elements. Limited resource-binding and flexible routing/node assignment help ICN to better support resource optimization in the network. This is achieved through binding and routing based on specific content names, and creating a content name-based unifying interface between the application and storage, processing, and forwarding nodes in the network. These properties make ICN interesting for us as an alternative solution in future to overcome the latency limitations and speed improvement challenges in the network.

ICN's flexibility in retrieving content creates its own challenges in the network. Congestion control and privacy are two of these challenges that are throughly discussed in this chapter. Congestion control and managing the resource usage is a main concern in many ICN proposals that relax fixed bindings between a content transfer operation and specific set of

resources. Privacy is an issue in ICN networks that rely on the user being eager to share information about his/her item of the interest with the network.

There are other challenges to the ICN that we have not discussed in this chapter. These challenges include designing high-speed content routers [157], and providing support for scalable inter-domain routing [47, 132]. These remain open questions that affect the wide-scale acceptance and deployability of ICN in near future.

4. Network resource utilization and its challenges

"In any moment of decision, the best thing you can do is the right thing, the next best thing is the wrong thing, and the worst thing you can do is nothing."

(Theodore Roosevelt)

4.1 Bandwidth Optimization

In previous chapters, we have discussed that TCP although used quite commonly, it is not necessarily the most efficient protocol in optimizing the available network bandwidth as quickly as possible. In Publication I we re-examine some of TCP's abstractions, and propose a solution to improve TCP's efficiency in bandwidth probing.

4.1.1 Information sharing

Fast data transfer over the network requires bandwidth availability. However, bandwidth availability is not the only requirement for fast data transfer. A data source also needs to *know* about this available bandwidth. In TCP, this knowledge comes from bandwidth probing. Every single TCP flow begins its own bandwidth probing operation when it starts, and finishes the bandwidth probing when it ends. The bandwidth probing starts with a relatively low estimate for the available bandwidth, and as the flow continues this estimate slowly starts to get closer to the real value of the available bandwidth.

TCP assumes there is no information about the available bandwidth when a flow starts. That is the reason that every flow has to start its own bandwidth probing instance. This assumption might not always be true. There might be some other existing flows that share the same path as this

new flow, and they have already estimated the available bandwidth on that path. What makes things complicated is that even if the bandwidth availability information already exist in one node, it is not possible to share this information across different flows. In Publication I, we discuss the benefits of having a new bandwidth probing module that allows sharing the bandwidth availability information across different flows and over different time spans, similar to the earlier work in Congestion Manager (CM) [24]. In this way, a new flow can just use the bandwidth availability information that is collected by other flows, and speed up its operation. This information could be retrieved from the flows that are currently in progress, or it could be retrieved from the existing state of the flows that have recently ended. Of course, the bandwidth availability information from the old flows needs to follow some aging patterns to be useful for the new flows. For example, the available bandwidth estimation for each path could be halved per last estimated RTT on that path.

In Publication I, we extend the CM [24] idea to provide support for information sharing between a wider range of flows. In CM the bandwidth availability information could only be shared among the flows that share the same end-points, because those are the ones likely to share the same bottleneck. This limits for number of flows that can benefit from CM style information sharing. For instance, looking at Figure 4.1, for the flows that start from node J and end at any of the nodes E, F, or G the bottleneck link would probably be the slow cellular access link near the node J. However, a CM-based resource probing instance on node J could only share the bottleneck information among the flows that end at the same node, e.g., at F. Separate resource probing instances are then needed to share the same information among the flows that end at E or G. The particular problem that Publication I addresses is to find a better abstraction for capturing and sharing the bandwidth availability information between different flows, and identifying the bandwidth bottlenecks. This could help to share the resource probing information among a larger number of flows compared to the Congestion Manager.

Pathlets

Based on the TCP/IP abstractions, the lowest possible granularity for which a common resource probing module could be proposed, is an end-to-end flow. A TCP end point that hosts a congestion control module could only associate the network resources to a flow, based on the other end's ad-

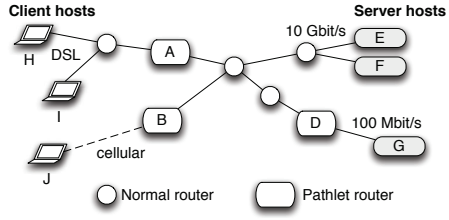

Figure 4.1. A sample instance of the network model

dress prefix. This is a limiting factor for designing a common bandwidth probing module, as in the current Internet not many flows share the same source destination pairs.

In Publication I we suggest to keep the congestion control states per path segments. The congestion control state then remains useful over time and across different flows. We introduce the concept of congestion control *pathlets*. In our proposal, the network path between the sender and the receiver is split into pathlets, which are separated by specific *Pathlet Routers*. Pathlet routers are identified by unique IDs. Pathlets are then indicated by the ID pair of the pathlet routers at their edges. Pathlets are identified at the beginning of or during a connection and their resource availability information is updated and used by different flows. For example, for connections between the server host *G* and client host *H* in Figure 4.1, the sender learns that the connection consists of pathlets GD, DA, and AH. In the same scenario, a connection between the server host *G* and client host *J* would consist of pathlets GD, DB, and BJ. In this case all the connections that go from G to H or J, can at least share the bandwidth availability information over the pathlet GD.

Our Pathlet-based Transport Architecture (PTA) mode, suggested in Publication I, supports two distinct operations: the pathlet discovery to identify which pathlets are traversed by an opened transport connection; and data collection, which happens during the normal communication. We use a "Pathlet header", as shown in Figure 4.2, to collect information about the network path both during the initial pathlet discovery, and during normal data transfer operation. During the pathlet discovery, every traversed pathlet router adds its identifier to the first free slot in the pathlet discovery section in the pathlet header. In this way, when a packet reaches the receiver, it contains information about the pathlets traversed on a path. This information will be echoed back to the source. Data collection occurs after the connection setup phase and pathlet discovery. In addition to the high level path-specific bandwidth estimation, PTA also

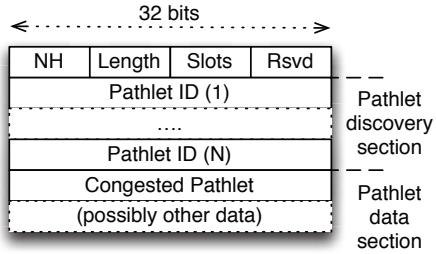

Figure 4.2. The Pathlet header

provides the oppertunity to locate the congestion on the path. Basically, if congestion happens, the pathlet routers that are closest to the congestion point identify themselves in the congested pathlet field in the pathlet header. This information is then sent to the source allowing the source to see (and react to) where in the network congestion happens. However, this approach of data collection requires that congestion is explicitly understood within the network. For this purpose, every end-point and router that benefits from PTA should be able to mark and interpret ECN (Explicit Congestion Notification) [134] bits in packet header. A pathlet router realizes if it is the closest pathlet router to the congestion point, if there is an ECN congestion mark in the packet header, but the congestion pathlet field in the header is not marked. The first pathlet router that sees the ECN mark then inserts its own identifier into the packet header. This would the allow the source to approximately identify the congestion location in the network.

Once the pathlet-based abstraction is in place and the required information is collected, various advanced algorithms could be used for different congestion control and resource optimization purposes. In Publication I, however, we stick to the simple case when the pathlet-related information sharing could be used for changing different default values such as the slow start threshold, minimum RTT, and the MTU size. In particular we investigate the initial ICW setting and how it could be changed dynamically to help faster transmissions.

4.1.2 The ICW setting

The initial bandwidth estimation value in TCP is defined in the ICW size, which affects the speed of operation especially for short flows. We have already discussed that reducing the number of RTT rounds that a protocol spends for a file transmission operation can increase the speed. In TCP, one way of reducing the number of RTT rounds is increasing the ICW size,

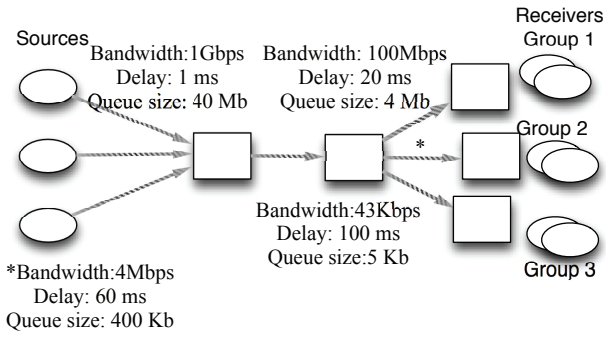

Figure 4.3. The simulation topology

especially useful to quickly finish short flows.

TCP's initial bandwidth estimation and the ICW size for every flow has to be defined conservatively to avoid sudden congestion and too many packet losses. This setting is constant and cannot be changed, even if there is a lot of bandwidth available. In Publication I, we argue that improving the speed and efficiency would be much easier in TCP, if one could at least change TCP's default ICW setting dynamically based on the available network bandwidth. There, we use our suggested PTA model for estimating the ICW settings for new flows. In this way, a new TCP flow does not necessarily have to go through using a constant ICW setting for its initial bandwidth estimation, if there is information available about the pathlets that the new flow traverses. Instead, the new flow can just use the bandwidth availability information that is collected by other flows, and speed up its operation.

To evaluate the possibility of dynamic ICW setting with PTA, we ran some ns-3 simulations. For these simulations we use the topology shown in Figure 4.3, in which the bottleneck links are the last links next to the router that connects to the receiver groups. Each group of receivers are connected to the source through slightly different bottleneck links, with bandwidths of 100 Mbps, 4 Mbps, and 43 Kbps. The rest of the links have a bandwidth of 1 Gbps.

We use two different traffic profiles in our simulations, simply calling them traffic profile 1 and traffic profile 2. Traffic profile 1 follows a Pareto model with a long tail for short flows (Pareto mean 40 KB, shape 1.5), as motivated by other studies on increasing initial congestion window size [14, 52]. Traffic profile 2 has longer flows (Pareto mean 200 KB, shape 1.5). The latter profile emphasizes the effect of larger initial congestion window because, for short flows, the initial congestion window does not al-

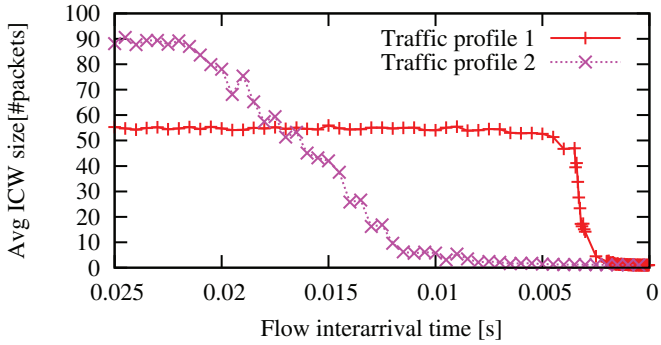

Figure 4.4. Average ICW size with 1000 TCP flows between servers and clients with 100 Mbps bottleneck

ways have significant effect. Flows arrive in the network based on Poisson distribution. The inter-arrival times between different flows are exponentially distributed.

As a starting point we show the average ICW size that the PTA model achieves under various conditions. Figure 4.4 illustrates the average window size that our PTA-based method achieves with different traffic profiles. As can be seen in the figure, the PTA-based method adjusts the ICW size based on the congestion on the link. Traffic profile 2 with longer flows naturally causes more load on the network than traffic profile 1 with shorter flows. The graph shows how this difference is reflected in the initial window selection made by the PTA.

Goodput defined by the file size divided by the flow completion time, is one metric to show the performance improvements achieved by PTA in terms of bytes per second. Figures 4.5 and 4.6 show with low congestion, different initial window sizes results in different performance. But as the congestion increases the performance differences vanish. The graphs, again, show that initial window of 10 packets is better for high-speed links, while initial window of 3 packets works for low-speed links. PTA is better in both cases, because on high-speed links it can pick even higher values than 10, and in the low-speed case it stays at initial window of 1 or 2 packets.

It is also worth mentioning that these results emphasize that even with PTA the overall goodput remains lower than the nominal link bandwidth. This is because the goodput covers the whole duration of TCP connection, including initial SYN handshake and waiting times for the ACKs. This becomes troublesome especially with short flows and longer relative round-trip time. One still needs to consider these aspects when thinking about improving the speed and efficiency in the network.

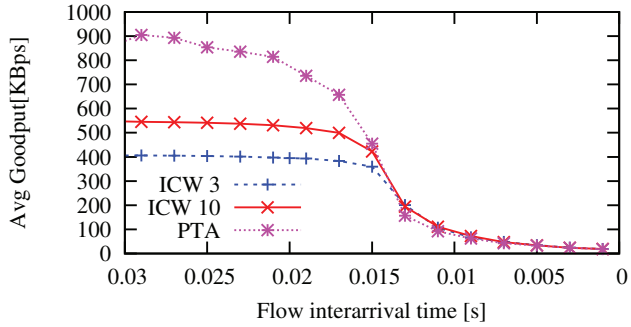

Figure 4.5. Average goodput with Multiple TCP flows of traffic profile 2 between servers and clients with 100 Mbps bottleneck

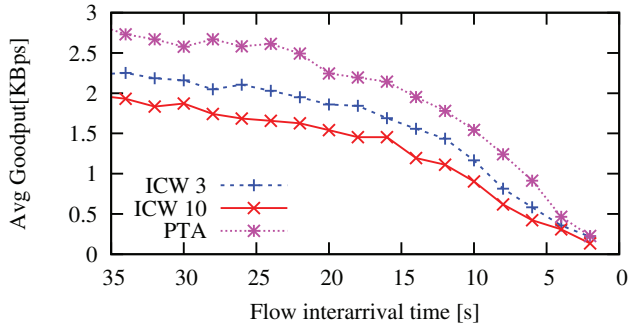

Figure 4.6. Average goodput with Multiple TCP flows of traffic profile 2 between servers and clients with 43 Kbps bottleneck

4.2 In-network Storage Optimization

Our discussions above partly focuses on reducing the RTT rounds that a protocols spends for a file transmission operation. There are also other ways of reducing the RTT-dependent portion of the overall user-experienced delay, e.g. propagation and queuing delay. In this regard we think benefiting from other available network resources such as using in-network storage for caching could be useful in reducing the user-perceived delay. Our work in Publication II and Publication III focuses on in-network caching.

Caching is one of the oldest network mechanisms used to reduce bandwidth consumption and latency, and to increase speed [60, 137]. It is commonly believed that caching is a costly extra service that only pays off if there is a high probability that the cached objects are going to be re-used [68, 162]. This mindset makes it difficult to argue in favor of supporting wide-spread in-network caching [61]. However, based on our earlier discussions in chapter 2, we think as the price of memory and storage drops the costs associated with caching could also reduce to the level that it is better to cache than not to cache . In Publication II and Publica-

tion III we discuss the caching style of in-network storage optimization. Here, we summarize some of the important points shaping our thinking in Publication II and Publication III.

4.2.1 Object Granularity

Application level proxy caches are the most common form of caches. HTTP proxies as well as caching solutions designed for peer-to-peer applications are examples of these application-specific mechanisms [66, 165, 90, 102, 147]. However, there are at least two problems with application specific caching: first, it is difficult to design and manage different caching nodes for different applications, and second, application level objects do not always provide the best granularity for storage and re-use. In both our proposals in Publication II and Publication III we consider caching below the application layer, and use the ICN [88] and ALF [42] style of identifier transparency to define finer granularity objects for caching.

Packet granularity
When thinking about caching below the application layer, packets do not even seem significant enough to be considered. This is because application level caching might just be enough by eliminating most of the redundant transfers [61]. However, in Publication II we take the ICN view [88, 10, 146] into account that if packet level objects are identifiable anyway, then the router and transport protocols might as well use those objects to improve their own operations. For example, the transport protocol and its dependent modules can benefit from these identified objects to share the common bytes across different flows.

In Publication II, inline with many other ICN works, we identify every packet by means of a content-specific identifier. This content-specific identifier is shared across the transport end-points as well as the routers. These identifiers allow the packets to meaningfully reside in some memory and stay inside the network independent of any transport level concept such as flows. Different protocols that can access these identified packets can then benefit from them, e.g., for replacing a lost packet, or eliminating redundant transfers of the same packet over a congested link.

Byte-stream granularity
In Publication II we assume that routers are interested in content-centric routing and forwarding and therefore, it makes sense that each packet

is identified independently. Nevertheless, if routers are not able to use the packet specific content identifier for routing and forwarding, other object granularities could be preferable for caching. In Publication III we suggest byte-stream identification to be used inside the network and for caching. A byte stream is identified by the means of using a seed identifier plus a range. The seed identifier is often an application specific identifier that represents the beginning of a file/stream and the range defines which part of that file/stream is being presented.

Similar to the named packets, every router that sees a named byte stream can cache and reuse these bytes. With byte stream identification the cacheable objects still remain independent from both applications and TCP style end-to-end flows.The biggest advantage of this model is that byte stream granularity allows size flexibility and partial caching of higher granularity objects. But unlike packet caching, the flexibility for partial caching is achieved without any need to define restrict boundaries for the cached items. Depending on the situation, bytes belonging to an object could be put together or separated easily inside the cache. Therefore, byte stream caching brings a new dimension to caching below the application layer: no size restricted object identification.

4.2.2 Pull-based transport protocols

Adding object identifiers and caching capabilities below the application layer requires defining new methods of accessing cached items. Therefore, in both Publication II and Publication III we introduce pull-based transport protocols which can specifically ask for data items with different granularities. The most important commonality between these protocols is their binding to a location-independent identifier. In both these protocols, the data transmission operation is identified through binding to content identifiers instead of location identifiers. This is a radical change compared to most transport protocols that are in use today.

In Publication II we use a pull-based content-centric transport protocol. In this protocol, the requester knows the packet identifiers beforehand (possibly through querying someone) or can generate them locally. The requester then requests each of the packets (logically) separately; in practice, several requests can be generated in parallel, following a logic somewhat similar to the TCP. As each caching node within the network understands the request and response packets, it can easily cache the responses (data packets) and reply to data requests. If a request packet is

answered from the cache, the request need not be forwarded. In this way, a router cache can independently reduce the load on the upward path, without necessarily requiring any inter-cache co-ordination effort. It is also expected to reduce the content retrieval time that the transport protocol achieves.

In Publication III we introduce a pull-based transport protocol that is adapted to requesting identified byte-streams. In this protocol, the receiver can request a window of bytes identified with the content-based stream identifier and the byte range. This is done using a REQUEST packet. Any stream caching node on the route between the server and the client can interpret this request and reply to it without having to consult the application layer. Only the server would do the latter to, e.g., fetch the required data from the source (e.g., a file). Data is carried in DATA packets. If intermediate nodes are able to supply part of the data on behalf of the sender, they update the offset and byte range in the REQUEST packet accordingly.

Packets with the same content identifier and byte range are inter-changeable, regardless of the application that has triggered them. Due to the stream caching nature, also overlapping fractions of packets with partially intersecting byte ranges may be stored, as packet boundaries are no longer an issue.

4.2.3 Costs and policies

Reducing the transmission costs is one of the main reasons for using the in-network storage and caching. However, if not designed or used properly, the caching operation could become useless or, in the worst case, it can even add to the cost itself. Thus, the design of a proper caching mechanism plays an important role in defining the system costs. Often in these systems the caching success itself is then evaluated based on the cache hit rates and the latency [152, 151].

When the storage space is limited and there are many items to cache, it is necessary to define a proper policy to keep the cache useful. In this case it is important to track the content popularity trend, and choose the right storage and replacement policy to keep the cache hit rate as high as possible, either in a single cache or in a set of collaborative caches [120, 101, 45, 153]. The success of a caching policy depends on the traffic (cacheable content) and access (request diversity) pattern of the cache. For example, in one local setting replacing the Most Recently Used (MRU) item could

result in the highest hit rate, while in another setting replacing Most Frequently Used (MFU) item could increase the hit rate. In these cases the cache designer should carefully investigate the traffic and access patterns in a local setting, before defining and implementing a specific policy for a cache.

Although the cache hit rate is an important factor, the system costs are not always reflected in an absolute hit rate value, and policies are not always defined to increase that absolute value. Complexity is another important factor in choosing a caching policy. For example, Least Recently Used (LRU) caching policy is widely used today, because it is simple. There possibly are many other algorithms that specifically consider different items popularity and achieve higher hit-rates, but none of them is as widely used as LRU. The reasoning is simple, the benefits of using popularity-based schemes are not often big enough to justify the added complexity of monitoring the popularity.

The added latency is another factor that affects the choice of caching policies. In some environments like multiprocessors, caching is mainly used to overcome the latency issues that exist between different levels of memory and the processors [5, 168]. In such environments, in addition to the hit rate, the success of a caching policy depends on the latency that it adds to the system. If a specific policy can achieve a high hit rate but it is slow in doing that, it becomes less useful for the latency-sensitive environments. In these cases, a cache designer has to pay extra attention to the simplicity and the speed of a caching policy.

In Publication II and Publication III we discuss caching policies in two different scenarios in which increasing the hit rate by careful investigation of traffic and access pattern is not our main concern. Instead Publication II and Publication III look at some other factors affecting the system costs from the speed and load sharing point of view: this is either because the cache space is big enough to reduce the concerns regarding the hit rates (as in Publication II), or it is because the caching system can evaluate its hit rate in real time and affect the traffic pattern towards the cache (as in Publication III).

Random caching

We have previously discussed that router processing can add to the latency and reduce the speed in the network. Even in current routers in which the basic functionality is limited to routing, forwarding, speed is an

issue. In such routers, adding yet another cycle consuming functionality seems to be out of question. This is because of the possibility of added latency and the reduced speed implications that a new function such as caching can impose to the routers. However, in Publication II we introduce packet level caching in routers as a optional functionality that takes into account the speed limitations.

Similar to the caching in multiprocessor environments, the packet level caching described in Publication II cares most about simplicity and speed. The reading and writing operations to memory have to be done quickly enough so that it does not add to the latency. In Publication II we address this issue by introducing random caching. A packet is only written to the memory with a random probability and it is only read from the router memory if there are enough available resources to do this operation in line speed.

The random caching method suggested in Publication II benefits from a specific form of addressing and memory hierarchy to control the added latency in each router. Figure 4.7 shows one sample cache structure proposed in Publication II. There, we rely on the fact that once a packet has been written to the memory, it remains there until it is overwritten by another packet. Each incoming packet with some probability is written to the available packet cell in the DRAM packet store, and its index information is updated at the index table. Depending on the structure and the randomness of the Packet Identifiers (PIds), the index address corresponding to a PId can be defined from a range of the PId bits, e.g. bits 0...23. The SRAM entry for an index needs to store a non-overlapping range of the PId bits, e.g. bits 24...51. With this,the router can check from the SRAM if it has any packet with the bits 0..51 matching those of a given PId. Then, if that packet exist, it could be retrieved from the DRAM. For storing a packet in the cache, different range of bits, S_i s, and packet access times could be used to compare different entries and choose the best one to write a new packet in place of an old one.

In Publication II the random caching functionality in one router is extended to be supported by a chain of the routers on a path. These routers could implicitly help each other to reduce the load on on one cache. Each packet has an equal chance of being written to the cache memory in different routers and it could be read from any of them. Each router then could choose to serve a packet or not depending on its load or some other metric. In Publication II we have used some simulation examples to describe the

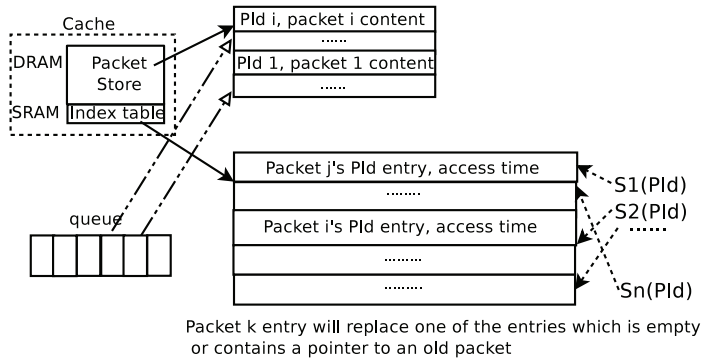

Figure 4.7. The proposed router cache structure in Publication II

usage of router chains in our random caching model. Our simulation settings consists of an ns-3 implementation of a native content-centric protocol stack, using in-packet Bloom Filters [89] for forwarding. The basic scenario includes one of the branches shown in Fig. 4.8, with 8 routers in the path. Each router is considered a possible cache source and is identified with its hop number from the receiver. Each router caches the packets passing by with the probability d. In our experiments we have varied the percentage of packets cached at each router, choosing the cached packets randomly with the probability d of $1, 1/2, 1/3, 1/4,$ and $1/8$.

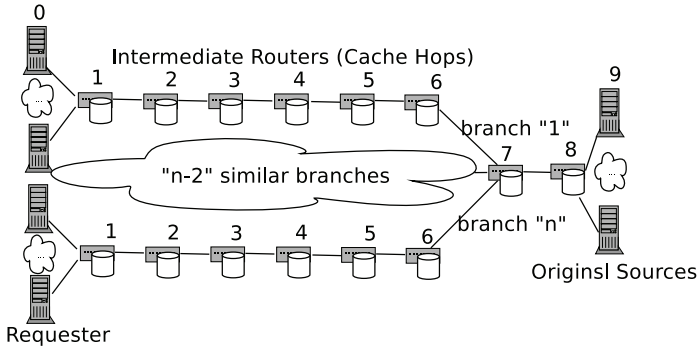

Figure 4.8. Example topology with requesters on the left, original sources on the right, and potential caches along the path.

In one set of the experiments, we have transferred an application level content item once and applied the random caching method on the path. We have then started requesting that same item for the second time. We have used a pull-based transport protocol for requesting named packets. As a result, requests for cached packets arrived at the first router on the path, and if not served, continued upward.

The results, shown in Fig. 4.9, indicate that the overall mean efficiency

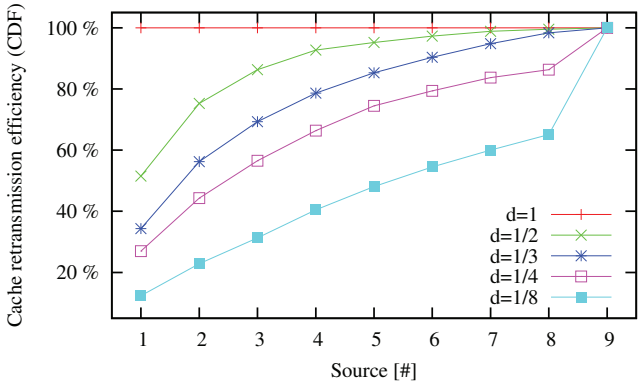

Figure 4.9. A sample packet retrieval percentage from different cache hops

can be quite high, even with partial caching at each router. With full caching, the first cache essentially covers all the requests, while with 50% random caching, 4 caches achieve $\sim 95\%$ hit rate. For the relatively low caching fraction of 12.5%, the hit rate across 8 hops reached only $\sim 60\%$, indicating that some sort of simple inter-cache coordination might be useful at such low caching rates. For example, similar to the work in [104] the list of cached items in each node could be announced to its neighbors.

The random caching policy discussed in Publication II relies on the fact that most retransmissions of the same packets happen during a period of a few seconds. Therefore, the size of the cache could often be big enough to cover all the packet retransmission requests that happen during that period. Based on the results shown in [19] 10 seconds of packet caching is enough to capture 60% of the redundancies in the network traffic. Additionally, studies done by Rossi et al. [141] and Psaras et al. [128] emphasize that in different ICN settings a reasonably sized packet/chunk cache with random caching can achieve a reasonable hit rate anyways. This is because in a random caching scheme the most popular content is transferred through the cache most often and thus, it has the highest chance of being written to the cache. Therefore, as mentioned earlier the focus of our work in Publication II has been towards simplicity and affordability of the cache, and not towards the cache sizing or different cache replacement policies, and hit rate.

Anyhow, it is not always possible to define big enough caches that achieve a good hit rate in all kinds of scenarios. In Publication III we discuss one such case in which in-network caching is used to reduce the high server load. But similar to Publication II instead of investigating which

algorithm achieves the highest hit rate, we focus on a different criteria, namely: controlling the amount of traffic that is considered for caching.

Resource-aware caching

When there is limited space compared to the amount of data that needs to be cached, there is also the chance that the cache becomes useless because of too many re-writes in a short period of time. Cache pollution is also another possibility. In cache pollution data items that are never going to be used again fill in the cache and prevent the more useful data items from being cached. However, controlling the amount of the traffic that enters the cache could limit the chances of re-writing the cache too often or even polluting the cache. In Publication III we propose to control the cache traffic by marking the most valuable items beforehand. In a router caching case, this marking could significantly reduce the number of items considered for caching compared to the number of items that enter the router every second. In Publication III we rely on the simple observation that the server is the first entity to notice the potential redundancy. Therefore, if in-network caching is needed, the server itself can assist the network in deciding which items are worth caching.The server identifies the most valuable contents for caching by giving corresponding hints to the network: the server sets *cache-me* bits in the packet headers.The stream caching nodes on the path from the server to different clients will only consider caching those bytes that are marked with the cache-me bit.

Putting the servers in the position of marking different items for caching and using downstream network resources obviously raises its own issues. Every server could simply set the cache-me bits for every item, which would lead to cache pollution and the caching nodes would be back to second-guessing which items to store. Therefore, we need a mechanism that penalizes for excess cache-me requests and gives incentives for the senders to be selective about when to send them. To address this issue we have used a method similar to *ECN* [134] in which the sender is notified about overload, and *re-ECN* [33] in which whoever sends excess traffic to the network is held accountable for that. In Publication III we suggest to use an *overload* bit that is always initialized to zero by the sender. If the rate of incoming cache-me bits at a stream cache exceeds an threshold beyond which the caching node considers itself overloaded by the cache-me requests, it sets the overload bit in the packet. The state of the overload bit is echoed back to the sender through setting the *overload echo* bit in

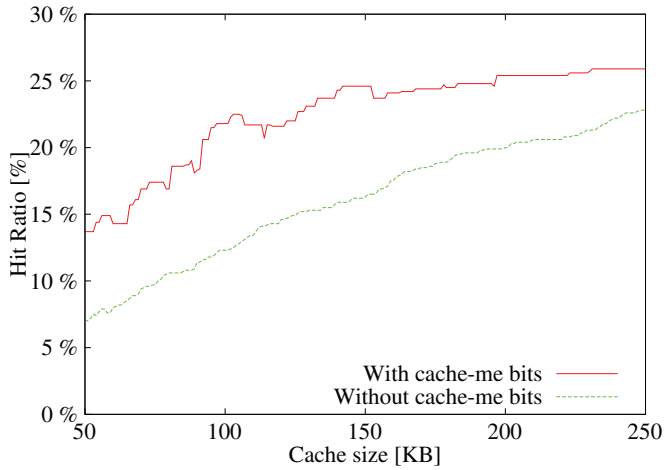

Figure 4.10. The hit rate difference without and with the cache-me bits

a different packet. As the sender receives overload echoes, it reduces its allowed rate of outgoing cache-me bits. In addition, there can be a policer near the sender (e.g., run by a network operator) that monitors the overload echoes and cache-me bits for each source, and clears any excess cache-me bits that the sender might try to send.

To show the validity of our design we ran some initial simulations using ns-3, for a case with 6 traffic sources and 1000 receivers routed through a common cache (shared buffer). Between the sources and the shared buffer there also is a policer that monitors the rate of cache-me bits per source against the overload signals, and eliminates excess cache-me bits based on this information. The senders have 100 files that have zipf popularity and are requested according to Poisson distribution with mean 50. The file sizes follow Pareto (mean:40 KB, shape: 1.5) distribution. When there is no overload, the cache-me bit is set purely based on the popularity of each file, so that the popular files are more likely to be transmitted with the cache-me bits compared to the non-popular files. Inside the network the cache-me bits are erased after an item gets cached. In case of overload as indicated by the network the senders stop setting the cache-me bits in the packet headers.

In the simplest scenario, figure. 4.10 shows that adding the cache-me bits can increase the cache hit rate. Figure. 4.10 shows with limited cache capacity using the cache-me bits can double the hit rate compared to the case where cache-me bits are not in use. The huge hit rate difference starts to fade as the cache capacity grows.

In Publication III we have also examined the role of the policer in case

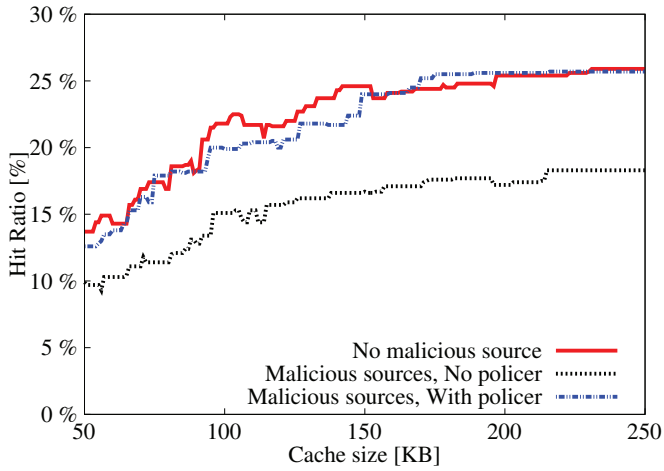

Figure 4.11. The request hit rate distribution with 1000 files and variable size cache.

malicious sources do not react to the overload bits set by congested caches. Figure 4.11 shows the effect of malicious hosts on the data hit rate on requests arriving at a caching node. By malicious host we mean the hosts that set the *store me* bits randomly and do not respond to the *overload* bits. The graph shows that the presence of malicious host significantly hurts the request hit rate at the shared cache, but the presence of policer helps to repair the situation.

4.3 Congestion control

The in-network caching models that we have proposed in Publication II and Publication III are most useful if there is no strict binding between a receiver and a specific source, and the receiver can receive the data freely from the best available sources. This freedom and unpredictability in resource assignments makes it rather challenging to make a reasonable association between the packet arrivals (loss) and the overall resource availability (unavailability) in the network. This could result in added difficulties for congestion control.

In this section we first explain how a TCP-like method might be adapted to deal with the congestion in ICN environments. This identifies the challenges that one have to deal with when adapting an end-to-end congestion control mechanism to be used in an environment with the possibility of packet caching and without strict end-to-end bindings. We then go through the details of one specific method that could help to address some

of the congestion-control issues in such environments.

4.3.1 TCP-like congestion control

In order to deal with congestion in ICN the first solution that came to our mind was designing a TCP-like congestion control mechanism. Therefore, in our earlier work that has been partly documented in [21], we have designed and simulated, *ConTug*. ConTug was one of the earliest works that explored congestion control issues in an ICN environment. Therefore, it is important to discuss its design and the related findings here.

In ConTug we rely on an ICN network in which there are no specific bindings between different sources and receivers. However, a source-routing-based channel identifier is always inserted into the packet header that specifies the packet route [89] (as illustrated in figure 4.12). The sender sends request packets to the network and the network routes them based on the source routing header, and then finds and forwards the proper response to that request. On the route that a packet takes, any node can act as an in-network cache responding to or storing that packet. Proper rate control with implicit resource probing is difficult in this environment, as any router along the path can store any response and respond to any request. However, ConTug introduces a TCP-like module, which resides at the end-host and makes rate adjustment decisions based on the observed packet arrivals and packet losses.

ConTug is a receiver-driven transport protocol that probes resources and handles congestion in a similar fashion as TCP. To achieve the congestion control functionality in ConTug we introduce a *Conceptual Congestion Control Window (CCWND)* to estimate the available network resources. Unlike TCP, in ConTug CCWND resides at the receiver, and the receiver adjusts its packet request rate based on the CCWND size. Another major difference between CCWND and a TCP window is that CCWND does not actually define any lower or upper bounds on the sequence number of the requested/ sent packets. CCWND only defines the number of on-the-fly requests that the receiver can send out to the network, without worrying about the sequence number of each request. The logic is adapted simply because there could be many sources that contribute to a content transfer operation, and thus, packets might be sent back to the receiver in any orders depending on their availability in different sources (caches). In this scenario re-ordering does not necessarily mean congestion or resource unavailability.

The operational logic for adjusting the CCWND size is a bit more complicated compared to TCP. That is because of the specific challenges attributed to environments with no binding to a particular source, and with the possibility of reactive caching as in Publication II. As an example of CCWND adjustment challenges, in figure 4.12 consider the case when the receiver *R1* has just started to request packets belonging to a content item *C1* and some range of these packets start to get cached in the nearby router *A*. Before *R1* finishes requesting and receiving the packets that it needs, the receiver *R2* starts to request the same packets belonging to *C1*. In this case, from the *R2* point of view, at first many of the requested data packets start to appear in the nearby cache *A* and cause *R2*'s perception of network resource availability to increase to near 1Mb/s with the minimum RTT of for example 20ms. However, because not all packets that belong to *C1* has been already transferred to the the cache, *A* can not respond to the new packet requests made by *R2*. In this case, *R2* needs to retrieve the rest of the packets from a further away source behind the 64Kb/s bottleneck with the minimum RTT of 100ms. In this case in *R2*, quick reception of packets could have resulted to a huge CCWND, which does not reflect the bottleneck near the source and its added delay. This could create sudden and unnecessary congestion on the bottleneck link. This effect could get worse, considering that while *R1* and *R2* are both requesting the same range of packets, their perception of the network resource availability could easily oscillate depending on who gets a specific packet first and leaves a copy of it in the nearby cache. In this case, reactive packet caching continuously changes the order and proportion of packets that are being served from different caches. Therefore, the RTT and the CCWND size could slowly oscillate between low and high values that are not a real reflection of the real-time resource availability in the network. We call this the *source unpredictability problem* and we discuss its side-effects in more details in [21].

In ConTug, we define multiple CCWNDs per transmission. Each CCWND is then expected to reflect the resource availability towards a specific source. At first this might seem a contradictory assumption, because our primary assumption was that there is no binding to a specific source in ICN. However, we argue that even if there is no specific binding between a specific source and receiver, the receiver could still cluster the *received* data packets and associate them to a specific source. In [21] for simplicity, we cluster the data packets based on the forwarding channel identi-

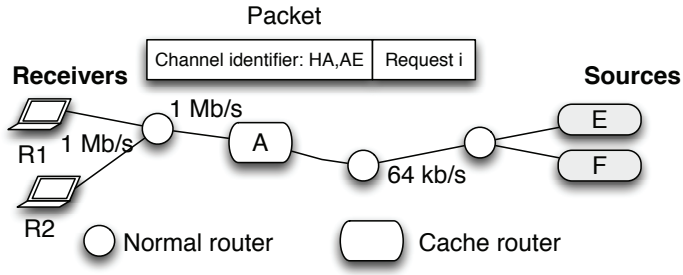

Figure 4.12. An example caching-enabled network setting

fiers included in the headers [89]. The receiver then creates a distinct $CCWND_i$ for each identified segment source i. The sum over different $CCWND_i$ yields the overall CCWND. The number of requests that are sent out to the networks is not specific to each $CCWND_i$, but it is defined by the the overall size of the CCWND.

The receiver starts requesting segments with one conceptual window, $CCWND_1$. Then, it enters the slow start (later congestion avoidance) ramp-up phase for that window, increasing its size on successful responses. Whenever there is a response from a new source, ConTug creates a new $CCWND_i$. Each response from the source i triggers an increase on the correspondent $CCWND_i$, thereby increasing the overall CCWND size, allowing more outstanding requests to be sent to the network.

RTT-based congestion indication

In ConTug the request rate is decreased upon the reception of a congestion indication. We mainly use RTT-based timeouts as the sign of congestion. Our first choice is using the forwarding channel timeouts (Eq. 4.1), which are triggered by packets that never reach the receiver. In Eq. 4.1 we use the maximum observed RTT to make sure that a loss has actually happened. A channel timeout will cause every single $CCWND_i$ size to reduce to half. Every request, which is not answered by the channel timeout, will be repeatedly sent to the network until it gets answered.

$$Channel\ Timeout = C \times MAX_{i \in \{0,n\}} RTT_i \tag{4.1}$$

Using the above methods, ConTug aims at adapting the rate based on the available bandwidth. However, ConTug channel timeouts (alone) are inadequate and would yield performance results even worse than TCP. This is because in ConTug reception of the data packets at the receiver side is considered a sign of resource availability, either this resource is

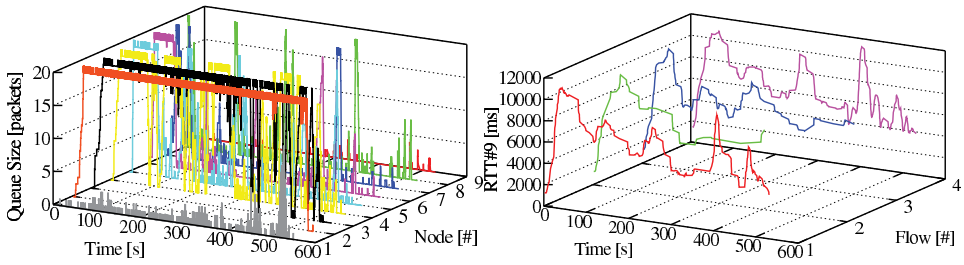

Figure 4.13. Four concurrent ConTug transfers in a scenario *with* caches when only sign of congestion are channel timeouts (a) Queue occupancies in different intermediary nodes on the left, (b) RTT changes on responses arriving from the *original source* (node #9) on the right.

bandwidth or it is the in-router cache memory. A packet might be received from any range of nodes starting from a first hop cache to an original source which is located behind one or many bottleneck links. At the same time, timeouts are only triggered based on the RTT of the furthest away source or the source located behind the most congested link. In this scenario the number of packets that are received at the receiver in a timeout period "T" from all available sources could be multiple times more than the number of packets that are received from only one source over a TCP connection in the same period. Therefore, the CCWND size could increase much faster than a TCP window, and cause more severe congestion effects.

Our ns-3 simulations in a network setting similar to the figure 4.8 confirm this claim. In this high-delay network setup different links have 100 Kbps bandwidth and their delay is set to be 20ms. There is unlimited caching capacity in all the nodes on the path, and the caches are empty at the beginning. Figures 4.13a, and 4.13b then show the queue size and the observed RTT for the packets that arrive from the furthest away source. One can clearly see that this simple initial algorithm is not able to control the RTT, and queue size and, as a result, the amount of the congestion in the network can remain drastically high (note: in this setup the maximum queue size is 20 packets). The queue size remains at its maximum value in first 2 hops from the receiver, and naturally result in many timeouts and packet drops.

The one difference that might make these many timeouts and losses more tolerable in ConTug compared to the TCP is the caching adaptation in the network. In TCP, congestion could cause packet drops and retransmissions of the same packet over and over on the same set of links. In

ConTug caching to some level can reduce the number of retransmissions from the original source. Instead packets which are dropped from the queue could be served from the caches that have a copy of them. For Instance, figure 4.14 shows in the above scenario most received packets are served from the first 2 hop caches. This happens even though caches were initially empty and during the simulations they are filled with copies of the packets that have reached the cache, and got forwarded/ dropped afterwards. Most packets that ultimately get served from the cache are the ones, which have actually been served from an original server before, but got dropped before reaching the receiver. In this way, as long as there is enough space, caches can compensate for exhaustive retransmissions.

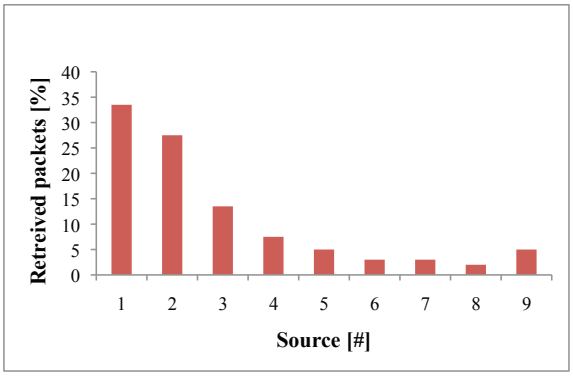

Figure 4.14. The proportion of packets received from each source for one transfer

Nevertheless, the unmanaged congestion results as shown in figures 4.13a, and 4.13b suggest that we might need to use other indicators for congestion and resource unavailability in the network. Therefore, in ConTug we use two other signs of resource unavailability, first a source specific timeout, and second, the increase in the source-specific RTT. Source specific timeouts (Eq. 4.2) identify less availability towards a specific source, either it is because there is no data available in that source or it is because of the congestion. This kind of timeouts happens when no packet from the source i arrives during the timeout period. In this case the $CCWND_i$ reduces to half.

$$Source\ Timeout = C \times RTT_i \tag{4.2}$$

In ConTug we also exploit an RTT-based resource estimation approach similar to the one in TCP Vegas [31]. This is to alleviate the consequences, when reception of the packets from nearby caches results to misleading estimates about the available bandwidth further away down the path.

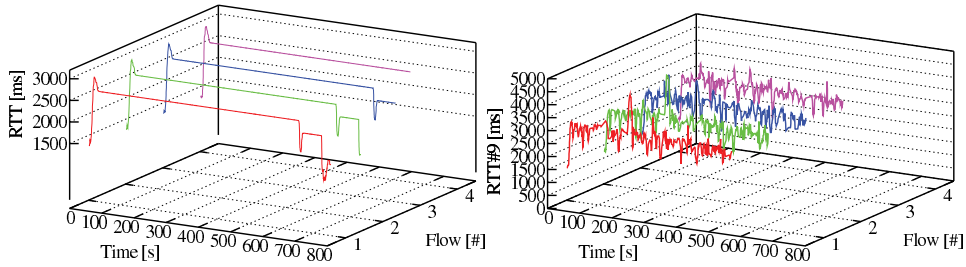

Figure 4.15. RTT variations in a scenario with four concurrent ConTug transfers; showing the case on responses arriving from the *original source* (node #9) (a) without caching on the left, (b) *with* caches on the right.

We use the increased RTT estimation as a sign of congestion that is reflected in the decreased packet arrival rate. Similar to TCP Vegas [31] the difference between the expected rate and actual rate is counted as an indication for the amount of resource unavailability in the network, and defines the proper CCWND size.

In ConTug, the expected rate at each time is calculated based on Eq. 4.3.

$$Expected\ Rate_i = \frac{CCWND_i \times Segment\ Size}{BASERTT_i} \tag{4.3}$$

The $BASERTT_i$ is the minimum RTT that receiver sees in the batch of samples coming from source i. The actual rate is then calculated based on Eq. 4.4.

$$Actual\ Rate_i = \frac{CCWND_i \times Segment\ Size}{RTT_i} \tag{4.4}$$

At the beginning, if the difference between the expected and actual rate is more than a value γ, the algorithm enters the congestion avoidance phase. In the congestion avoidance phase, if the difference is less than an α value the $CCWND_i$ will increase and if it is bigger than a β value then $CCWND_i$ will decrease linearly.

Adding the new resource unavailability indicators to our design makes it a better fit for congestion control in models with reactive packet caching and without any support for source binding. To initially confirm this claim we compare the window and RTT changes in both caching and non-caching environments. Figure. 4.15b illustrates the RTT changes of 4 simultaneous receivers; there we use RTT_9 (RTT of packets served from source #9) as indicative variables for our algorithm. Figure. 4.15b shows that the RTT estimates for source #9 are a bit higher on the caching-enabled channel compared to the case shown in figure. 4.15a. However, RTTs remain rather stable and reasonably controlled compared to figure. 4.13b. Figures. 4.16a and 4.16b also show that the queue occupancies

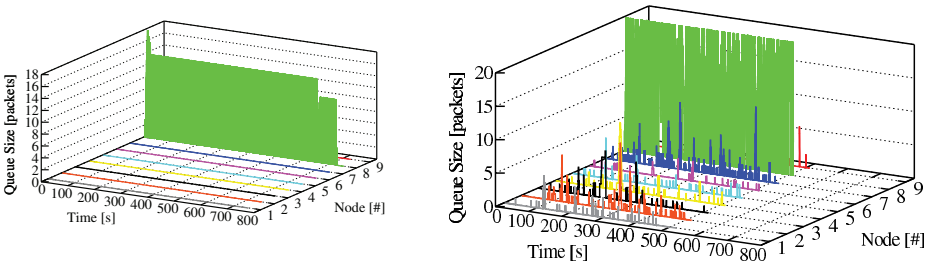

Figure 4.16. Queue occupancies in different intermediary nodes with four concurrent ConTug transfers; showing the case (a) without caching on the left, (b) *with* caches on the right.

are better controlled with using the new congestion indications. In this model the major queue occupancy happens at the bottleneck queue residing at node #8 both in the caching and non-caching scenario.

Although RTT measurements and queue size variations show positive and rather stable trends in our evaluations, CCWND size and rate adjustments do not follow the same behavior. The plot in figure 4.17 shows that in our simulations the estimated $CCWND$ size does not become stable. For instance, considering the case for $CCWND_9$ in 4.17 the size oscillates for the duration of content transfer. This is because with random caching present on different routers on the channel, some random packets are found in sources closer to the receiver than #9, resulting in changing CCWND for the source #9. The same unstable window adjustment applies to $CCWND_i$ for other sources, and as a result to the overall $CCWND$ estimate. Therefore, even with our new congestion control countermeasures rate or window adjustment stability is still hard to achieve, which is mainly because of the randomness and unpredictability of caching and packets availability in different places in the network. One needs to take into account other countermeasures, if a stable rate estimate is desirable. Otherwise, she needs to change her exceptions about the outcomes of a congestion control algorithm in an ICN environment with the capability of in-router caching.

In addition to congestion control related evaluations, our findings in [21] show positive results regarding ConTug's efficiency in improving the speed of a content transfer operation. The average Flow Completion Times (FCT) shown in Table 4.1 is a sign that because of exploiting in-network caching opportunities, on similar routes, ConTug could finish the file transmission operations faster than the simple TCP. More details can be found in [21].

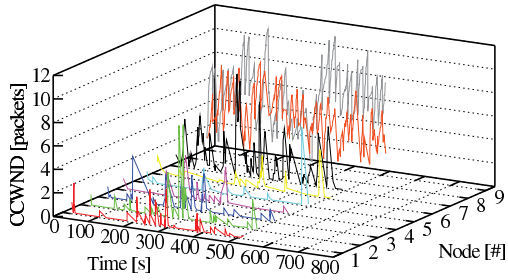

Figure 4.17. $CCWND_i$ variations for a sample flow in a scenario with four concurrent ConTug transfers

Flows(#)	ConTug's Mean FCT	TCP's Mean FCT
4	511.34	756
8	613.42	1512
16	782.81	3024
32	883.34	6048

Table 4.1. Mean FCT of ConTug and TCP flows requesting same content in caching enabled network

Although ConTug suggests one particular mechanism for controlling the congestion, there are still some open issues regarding the feasibility and efficiency of our proposed solution in relation to the *source unpredictability problem*. In the next section we explore some of these open issues and discuss an alternative approach for controlling the resource usage in the network.

4.3.2 Deadline-based congestion control

Our work in [21] emphasizes that congestion control in ICN networks requires a different treatment compared to the TCP/IP. In [21] we overcome some of the (re)source unpredictability issues in ICN by making weak correlations between the data packets and their sources, and by using RTT-based congestion indications. However, alternative solutions are needed in case making such correlations between the data packets and their sources are not feasible, or RTT-based congestion indications are not enough, or even if a better form of congestion prevention and resource management is required. One such alternative solution would be to require the network itself to take part in controlling its resource usage. If the network is involved in the congestion control then it can prevent oversubscriptions to its resources. It can even act intelligently and communi-

cate its resource availability information to the applications/end-hosts.

Most of the classic works that focus on network involvement in the congestion control operation [91, 50, 116] operate based on specific identification of a flow, and RTT measurements for that flow. The concept of the "flow" and its related RTT measurements are required for fair division of the network resources and preventing congestion [91, 50], and/or to properly feedback the network resource (un)availability status to the data source [116]. However, as discussed earlier finding an equivalent concept to a TCP flow or even proper RTT estimations for a content transfer operation is not that easy in ICN. This is because neither the end-to-end bindings nor the content to location bindings are enforced in ICN, see chapter 3.

In Publication IV we propose a new congestion control model that has little dependency if any, on correct RTT estimations and flow identifications. In the proposed model, we benefit from the fact that the network is more knowledgable about what it is transferring and how each transfer is done. Therefore, in our model the network can take advantage of this knowledge and deal with the congestion rather differently compared to the TCP/IP.

Deadlines

In Publication IV we introduce packet *delivery deadline* or *lifetime* as a prominent component used for congestion control and network resource management. In our model each packet is provided with an application specific deadline. For example, a real-time streaming or a web search application might request a packet deadline of 100 milliseconds, whereas for batch file transfer significantly longer deadlines (in order of seconds) would be sufficient. The data receiver specifies the packet deadline based on how long it is ready to wait for a response. The deadline setting then allows the data receiver to know how long it must wait before declaring the packet lost, and re-request the data if needed.

With the additional knowledge of packet deadlines, the network nodes can perform advanced scheduling algorithms for more efficient resource management. Before acting on a data request (e.g., an Interest packet in CCN [88]), the router checks the current load on the return path, and calculates whether it is possible to deliver the response to the data request within the assigned deadline, considering the currently scheduled data transmissions. If it seems that the data cannot be delivered by the

deadline, the data request is dropped immediately. This way the router proactively avoids using the upstream network resources, and helps in reducing load on the return path. For example, if the router already sees that forwarding the currently queued packets will take more than 200 milliseconds, there would be no use in forwarding packets have have deadlines equal to or below the 200 milliseconds [1].

In our proposed model in Publication IV, the network is not just a passive entity that has to use its resources to blindly and quickly forward anything. The network can check if the resources are available for serving a specific set of bytes within a specific time period. Setting deadlines that are visible to the network allows the network to control the resource usage within a well-defined time period and overcome the problem of tackling uncertain RTTs. Applications/transport protocols can also rely on deadlines instead of RTTs to choose when they have to re-request something or choose the proper request rate.

Scheduling

One of the biggest issues that can rise from our application-specific deadline setting is the issue of handling different set of deadlines in one network node. The problem arises as every single packet could conceptually have a different deadline setting, and for optimal resource usage the network nodes have to be able to schedule all these packets in an efficient manner. In Publication IV we suggest a simple deadline-based scheduling method that could help network resource management. In our suggested model, when a packet arrives at a node, the node checks its deadline carried in the packet header. If the packet is not expired, it is scheduled for forwarding.

The queue management in routers can leverage the deadline information in packets for better resource utilization. As summarized in Figure 4.18 in our specific model the router queue space is divided into N different subqueues to which packets are assigned based on their deadlines. In the figure, T indicates the starting time of the first queue that contains packets with deadline between time T and T+K, the second queue contains packets that have deadline between T+K and T+2K, and so on. At any moment, the current time is between T and T+K, and when the time reaches T+K, the first queue, and any possible unsent packets in the

[1]There is no flow-based QoS style guarantee in this model. The oversubscriptions are only controlled restrictedly to prevent congestion collapse in the network

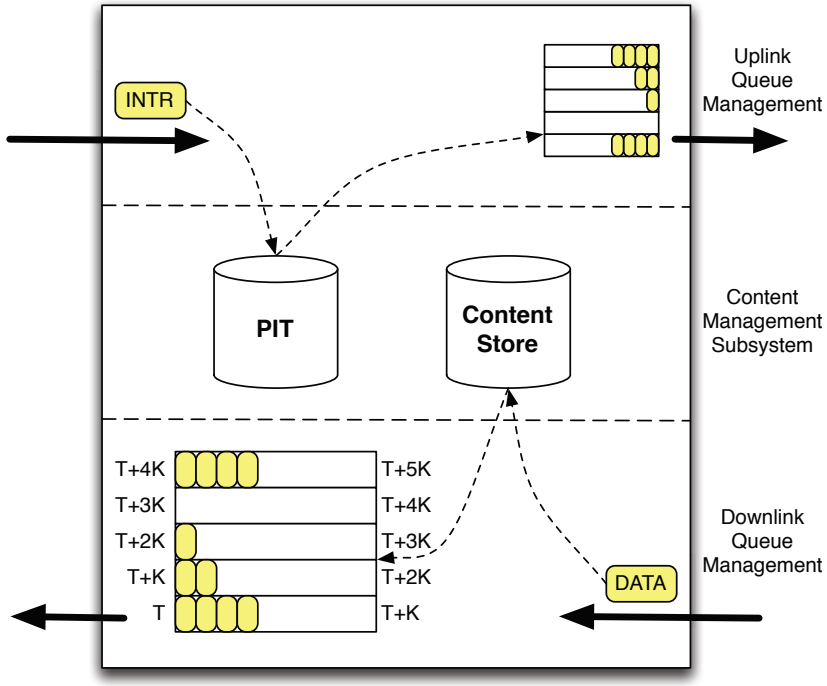

Figure 4.18. A sample queuing model for deadlines

queue are discarded, because the remaining packets have missed their deadline. The number of packets that could fit in each queue is limited based on the link capacity and the duration of K. Over-subscription is then controlled based on the number of the packets that could fit in to a subqueue.

We have done an initial evaluation for this proposal by using the ndnSIM simulation framework for ns-3[2], and conducting simulations on a dumbbell topology. Bandwidth of the bottleneck link in our simulation model is 1 Mbps, while other links have the capacity of 1 Gbps. The round-trip delay between data sources and receivers ranges from a few milliseconds to 100 ms, excluding the possible effects of queuing.

Figures 4.19 and 4.20 compare the goodput in our deadline-based scheduling heuristics and a simple FIFO. The FIFO scheduling results in devastating effect on the content with short deadlines, because as the queues build up on higher load, these packets have no chance of surviving. The deadline-based scheduling gives more capacity to the packets with short deadlines, but less capacity to the traffic with longer deadlines. We can

[2] `http://ndnsim.net/`

also see that under high load, although deadline-based queues manage to prevent over-subscription and control the congestion, the FIFO performance suffers with the single queue models, leading to unpredictable behavior.

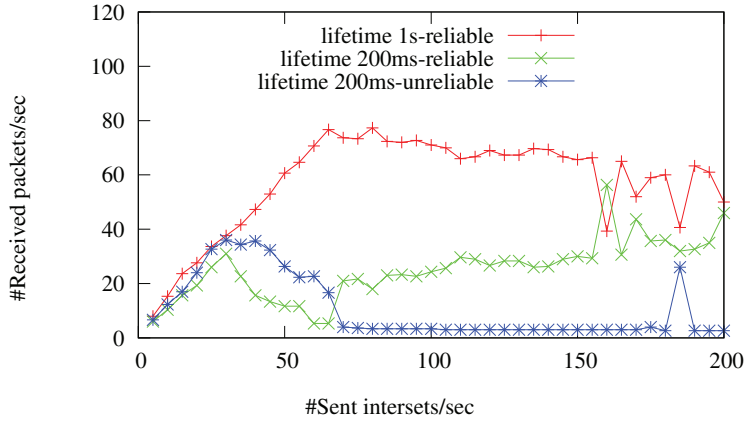

Figure 4.19. Number of received packets with FIFO style scheduling

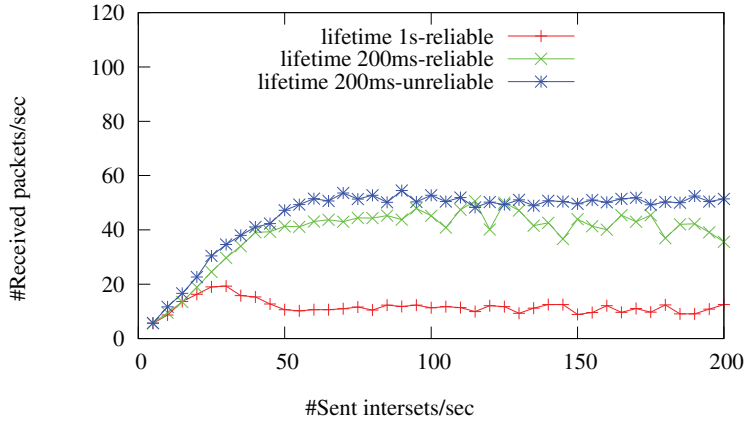

Figure 4.20. Number of received packets with time scheduled queues

Finally, it is important to distinguish the earlier queue management proposals and QoS mechanisms [70, 124] from the approach described above. The queues in the queue stack do not reflect priorities of traffic, and are not flow-aware: each packet is processed independently, and preventing over-subscription is achieved within the network only through the knowledge of the packet deadlines. Plus, the deadlines and the queue management model are used for in-network congestion control without requiring the routers to know the flow/ packet RTTs as it is the case in ICN.

4.4 Privacy

In section 3.3 we have discussed that using ICN model although might improve the content delivery experience by the user, but it could also cost the users' privacy. Common privacy issues that are associated with the IP network might still exist in ICN. For example, tracking user's location is not necessarily impossible in all ICN proposal [46]. But in addition to that, things get more complicated from the privacy perspective as the network and the available storage within the network form a new combination, accessible through the same interface. Privacy attacks that were previously considered only relevant to the storage nodes suddenly become relevant to the network as well, e.g. not being able to identify which data item has been requested. In Publication V we investigate the specific problem of name privacy in ICN, and propose a solution to address the privacy issues.

4.4.1 Privacy attacks

In Publication V, we assume that all parties attach to a public network in which all content requests (fetches) and content deliveries, can be observed by the adversary. We focus on two types of attacks in this scenario.

First, a *name-watchlist attack*, in which the adversary has a list T of content names that it wishes to filter or eliminate. It then monitors the links in the network performing real-time filtering; if a content fetch matches against T the adversary may squelch the request and/or record the user that requested that data. In addition, the adversary may attempt to delete the data with names in this target list T. The watchlist attack can be thwarted by query and data anonymity—if it is difficult for an adversary to determine whether a fetch or a piece of stored content matches against T, then it is difficult for the adversary to effectively interfere with the dissemination of this content.

Second, a *content-analysis attack*, in which the adversary does not use a precompiled watchlist, but instead inspects the data to see if it should have been flagged (it contains the wrong keywords, etc.). This attack can be thwarted by providing plausible deniability for users (which means that they can plausibly claim that they did not ask for bad data).

4.4.2 Privacy-preserving names

Since both the user and the adversary share the same information in our setting or any ICN setting that we know of, in Publication V we propose a mechanism, which creates some form of computational asymmetry that allows users to retrieve content efficiently but that makes it computationally expensive for the adversary to:

- Identify that the name being requested refers to flagged content. This makes name-watchlist attacks hard to mount on a large scale.

- Identify that the content retrieved should have been flagged. This makes large scale content-analysis attacks difficult.

For the purpose of creating computational asymmetry we hide the names and the content the adversary wishes to blacklist or discover by mixing the *target* content's constituent data blocks with the blocks of normal content or *cover file*. A user can then fetch the content by judiciously selecting mixed data blocks to reconstruct the desired content, while the adversary is forced to perform significant computation to determine the true name of and content in those mixed blocks.

In our proposed solution, an interested user first needs to use a back channel to retrieve a metadata file that enlists the name of the chunks belonging to a target file and its cover files. The metadata file can change dynamically, so that the combination of the cover files associated to a target file changes over time. The publisher of a target file needs to be aware of these changes and it needs to publish a reasonable set of chunks that at least partially matches the combination of the target file and the cover file chunks. In our solution, the original name for each chunk could be generated by applying a hash function H to different chunks. The name $n(t, i)$ for block t_i is $n(t, i) = H(H(t), i)$ where H is a well-known cryptographic hash function. The same applies to cover blocks c_i, taking the name $n(c, i) = H(H(c), i)$. This naming convention applies to all cover and target files; in the presentation of our notation we only referred explicitly to a single cover and target file, but our process applies to the entire set of target files and cover text cover files. The user can use the original name of different chunks, retrieved from the metadata file, to generate new names for new chunks that are a combination of the original ones.

The names of composite chunks are then computed by taking the hash of the names of the constituent blocks: for example, the name of the pair (t_2, c_7) is $H(n(t,2), n(c,7))$ which is given by $H(H(H(t),2), H(H(c),7))$. Requesting and retrieving a combination of multiple chunks works as a simple form of encoding, without requiring any agreement between the user and the publisher. It is the responsibility of the publisher to generate many combination of different chunks that belong to the the original and the cover file. It is the responsibility of the user to request different combinatory names, and re-build the original file by decoding (decomposing) the retrieved combined chunks.

4.4.3 Performance

In our proposed solution in Publication V the ICN style routing and content matching could be done without creating any permanent unique matching between a content item and its requested name. In this model combination of different names could result in retrieving the same object by a receiver. The computational asymmetry is created by making the relationship between name and content less predictable compared to what most ICN designs assume today.

In Publication V we discuss different aspects of computational asymmetry that could be created between the data requester and the adversary as well as the data publisher and the adversary. We base our discussions on the assumption that the content publisher creates chunks by mixing blocks of k tuples of the n target and m cover blocks. We then argue that in our model the cost to the user in requesting and later determining the constituent blocks of a chunk is $O(1)$, since the user explicitly selects chunks based upon the blocks it desires. For the adversary, the cost depends on its resources (time and storage). The adversary can either pre-compute and store all possible chunk names and their constituent block-names, or, if it has limited resources, the cost for it to decode each chunk is $O((n+m)^k)$ at best, since for each comparison the adversary needs to calculate all possible chunk names.

We also discuss that, while the work required of the content requester(user) is constant, content publishers must produce all the chunks in advance, and thus must perform $O((n+m)^k)$ work to generate, name, and publish the chunks for a given file. While this is more work than users must perform, especially in the general case in which the adversary does not pre-compute names, the amount of work the publisher must do is likely

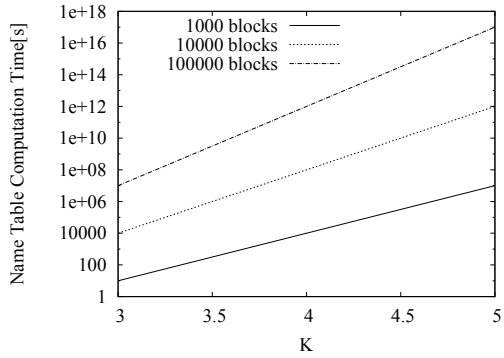

Figure 4.21. Having more blocks or increasing k makes the adversary's cost grow exponentially (linear in log scale) while the user's costs increase linearly.

less than the adversary since the content publisher knows exactly which flagged and cover files to consider during chunk generation, figure 4.21. More importantly, there is no time constraint for the content publisher, since chunk generation does not need to be done in real time. We also suggest that a more sophisticated way of preserving-privacy is when publisher is not forced to publish all possible combinations of target and cover blocks and instead publishes only a proportion of possible chunks it announces. This approach is much less resource-intensive for the publisher as it can announce a huge set of cover blocks to keep the adversary busy but not publish/match all the combinations itself.

The privacy benefits in our solution, however, come with their own performance limiting costs. Making the relationship between a content item and its corresponding name transient, makes it difficult to benefit from the in-network caching. If every requester can ask for the same object with a different name, then the cache might become useless because it is rare that the same name is requested again.

Another performance side effect of our proposal is the possible waste of bandwidth and the added delay that it can cause for the content retrieval. A specific form of performance deficiency would result, if a content publisher only publishes random mix of chunks and leaves it to the requester to ask as many chunks as possible to finally find the correct set of existing chunk mixes. This property although provides a good computational asymmetry between the publisher and the adversary, can result in an undesirable amount of wasted time and bandwidth. One needs to think about better solutions if s/he want to achieve both the privacy and improved performance at the same time.

4.5 Summary

In the first part of this chapter we have explored various solutions that relate to better resource optimization in the network. First, we have explored information sharing among different TCP flows to learn about different segments of a path and their congestion state. Then, we have suggested two different methods to benefit from the available in-network storage. One suggestion is to provide support for packet caching in the routers and the other is byte-stream caching. Both these caching methods are supported below the application layer and as part of the network/ transport layer functionality. A pull-based transport protocol is discussed to benefit from these in-network caches.

In the second part of this chapter, we have discussed the challenges that are created with the introduction of our specific caching models and their requirements. These challenges are mainly caused by accessing data by name below the application layer, and lack of binding between specific source and receiver during a content transfer operation. We have explored the congestion control, and privacy issues in such environment and proposed some solutions for each problem. We have discussed a deadline-based congestion control approach, which helps to manage the network resources in a highly flexible network environment with oscillating routes and oscillating sources, as it is in ICN.

There are many loose ends to our work, especially in the area of Information Centric Networks. Some of these issues that we have not addressed include: considering the fairness metrics in resource management, and defining privacy methods that work with different styles of naming other than the discussed hash-based flat names. In a more general level, we unfortunately lack a realistic traffic pattern that could have been used to evaluate our claims, especially when talking about caching and resource management in ICN. This means implementing and using each of our suggested methods still requires a much more detailed assessment of each solution. Such a detailed assessment would be helped, if there is an (experimental) implementation of an ICN model in use.

5. Conclusions

"The only reason for time is so that everything doesn't happen at once."

(Albert Einstein)

In this dissertation improving speed is the main motivation for our discussions. We have argued that in the near future delay and specifically physical latency could turn to be the major speed bottleneck in the network. Therefore, the network and different protocols should take measures to alleviate latency and its effects, and reduce delay. One countermeasure against latency effects is designing new mechanisms that reduce the number of latency-bounded operations in different protocols. For example, we have suggested a possible change to the TCP logic in order to speed-up its initial bandwidth estimation.

We have discussed latency effects could also be alleviated through supporting flexibility and better utilization of the available network resources such as the bandwidth, storage, and processing power. We have talked about ICN as one of the alternative network solutions that could help with speed improvement through flexibility and resource utilization. In this thesis we have namely discussed the usage of in-network storage as one form of simplified resource utilization in ICN. In our work, ICN abstraction could be used to cache fragments of the data in the network and below the application layer.

Any system that moves towards more flexibility to improve speed naturally loses its simplicity. In this thesis, we have for example argued that, although the idea of flexible resource binding as proposed in ICN could provide the network with better opportunities for resource utilization, it could also make it difficult to control the network resource usage. The possibility of changing the route and the source during a content transmission operation, contrasts the design assumption of most current conges-

tion control modules. Most congestion control modules assume the source and the route are fixed during a connection. Therefore, new forms of congestion control mechanisms are needed to address the requirements of a non-source-binding environments. We have proposed two different congestion control solutions that suit the source and route unpredictability in ICN. One solution relies on making weak associations between different data sources and the receiver, and manages the congestion at the receiver side. The other solution uses packet lifetimes. The network then can use the packet lifetime information to control the over-subscription to its resources.

Our work also discusses some of the privacy concerns regarding accessing data by name in the network. There we argue that this feature makes users more vulnerable to some privacy attacks, such as name-watchlist attack. We propose a specific naming solution that can alleviate some of the privacy vulnerabilities regarding accessing data by name. There are also other engineering challenges that tend to exist with most ICN proposals and we have not touched them in this thesis. These challenges include: scalable inter-domain routing, proper API design, handling real-time application scenarios, and etc. Nevertheless, when one considers the possibility of implementing and using ICN in near future, it becomes important to address all these engineering challenges.

Some argue that the speed and latency improvements achieved by widespread caching are not significant enough to justify the costs of adapting ICN to the current network [68, 61]. That might be true if ICN is summarized to be only about widespread network caching (as it might appear to be the case in most of this thesis). However, in a broader scale we see the power of ICN to be in the flexibility and freedom that it brings for resource utilization in the overall system and not just for in-network caching. For example, using ICN concepts the available processing power in the network could be utilized by the network itself in order to reduce the effects of latency and delay. For instance, the processing power in one router could be used to multiplex different copies of the same data item to one copy at one end of the link, and demultiplex and deliver those copies to different processes/ interfaces at the other end of the link. This could reduce the average experienced delay for each single copy. Although we did not discuss these ideas in this thesis, we argue that it is not fair to vote for or against usefulness of ICN only based on the criteria of widespread network caching and bringing the content closer to the users. In short,

to us the flexibility and resource utilization opportunities that ICN provides are important for improving speed, either used for caching or used for something else like switching to less loaded servers during a file transfer. Researchers need to concentrate more on rather unexplored aspects of ICN flexibility and speed improvements, especially in new contexts such as in data centers or between different virtual machines.

Regardless of the fact that ICN as a whole might never get implemented, we believe over time some of the ICN features could get embedded into the segments of the current network. That would be the time when the challenges and solutions discussed in this thesis could actually be investigated and evaluated. Before that time, most of our discussions remain at a general level with only initial evaluations.

Finally, we remind the reader that we discussed ICN only as one possible mean to improve speed, but even without ICN the motivation to change the network mechanisms and protocols does not change. The requirements for speeding up the internet is only going to increase in the future [80], and it is important to be prepared for the time our assumptions about the network and how it should work does not match our exceeding exceptions from it. Until that day we can only continue exploring new models and mechanisms in hope of finding a model that would better support our increasing demands.

Bibliography

[1] Conceptual architecture of psirp including subcomponent descriptions. Deliverable D2.2, PSIRP project, August 2008.

[2] Speed Matters. http://googleresearch.blogspot.com/2009/06/speed-matters.html, 2009.

[3] Cameron's clampdown on social media: Can he really rein in rioters by turning off twitter? http://www.dailymail.co.uk/news/article-2025106/UK-riots-David-Camerons-clampdown-social-media-tools-like-Facebook-Twitter.html, 2011.

[4] The 2012 update to the international technology roadmap for semiconductors. http://www.itrs.net/Links/2012ITRS/2012Tables/, 2012.

[5] ARM Cortex-R series processors. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.cortexr/index.html, 2012.

[6] Intelligent user mapping in the cloud. https://blogs.akamai.com/2013/03/intelligent-user-mapping-in-the-cloud.html, 2013.

[7] Juniper t4000 router. http://www.juniper.net/us/en/dm/t4000/, 2013.

[8] Named Data Networking (NDN). http://www.named-data.net/, 2013.

[9] Network of information. http://www.netinf.org/, 2013.

[10] PURSUIT-Pursuing a Pub/Sub Internet. http://www.fp7-pursuit.eu/PursuitWeb/, 2013.

[11] Fiber keeps its promise: Get ready. bandwidth will triple each year for the next 25 years. Forbes, 7 April 1997.

[12] Akamai. State of the internet. 4th Quarter 2012 Report, 2012.

[13] A. Akella, S. Seshan, and H. Balakrishnan. The impact of false sharing on shared congestion management. In *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on*, pages 84–94, 2003.

[14] Mohammad Al-Fares, Khaled Elmeleegy, Benjamin Reed, and Igor Gashinsky. Overclocking the yahoo!: Cdn for faster web page loads. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 569–584, New York, NY, USA, 2011. ACM.

[15] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM.

[16] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pages 63–74, New York, NY, USA, 2010. ACM.

[17] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. Less is more: trading a little bandwidth for ultra-low latency in the data center. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 19–19, Berkeley, CA, USA, 2012. USENIX Association.

[18] M. ALLMAN, S. FLOYD, and C. PARTRIDGE. Increasing TCP's Initial Window. RFC 3390, 2002.

[19] Ashok Anand, Chitra Muthukrishnan, Aditya Akella, and Ramachandran Ramjee. Redundancy in network traffic: findings and implications. In *SIGMETRICS*, 2009.

[20] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 281–292, New York, NY, USA, 2004. ACM.

[21] Somaya Arianfar, Lars Eggert, Pekka Nikander, Jörg Ott, and Walter Wong. Contug: A receiver-driven transport protocol for content-centric networks. Technical report, PURSUIT publish-subscribe project, 2011.

[22] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[23] Krste Asanovic, Rastislav Bodik, James Demmel, Tony Keaveny, Kurt Keutzer, John Kubiatowicz, Nelson Morgan, David Patterson, Koushik Sen, John Wawrzynek, David Wessel, and Katherine Yelick. A view of the parallel computing landscape. *Commun. ACM*, 52(10):56–67, October 2009.

[24] Hari Balakrishnan, Hariharan S. Rahul, and Srinivasan Seshan. An integrated congestion management architecture for internet hosts. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 175–187, New York, NY, USA, 1999. ACM.

[25] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wirel. Netw.*, 1(4):469–481, December 1995.

[26] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. *SIGCOMM Comput. Commun. Rev.*, 32(4):205–217, August 2002.

[27] Dhiman Barman and Ibrahim Matta. Effectiveness of loss labeling in improving tcp performance in wired/wireless networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, ICNP '02, pages 2–11, Washington, DC, USA, 2002. IEEE Computer Society.

[28] Mike Belshe. More Bandwidth Doesn't Matter (Much). Google White paper, 2010.

[29] Mike Belshe and Roberto Peon. SPDY protocol. Internet Draft draft-mbelshe-httpbis-spdy-00 (Work in Progress), 2012.

[30] Marjory S. Blumenthal and David D. Clark. Rethinking the design of the internet: the end-to-end arguments vs. the brave new world. *ACM Trans. Internet Technol.*, 1(1):70–109, August 2001.

[31] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. Tcp vegas: new techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*, SIGCOMM '94, pages 24–35, New York, NY, USA, 1994. ACM.

[32] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. 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 vol.1, 1999.

[33] Robert Briscoe. *Re-feedback: Freedom with Accountability for Causing Congestion in a Connectionless Internetwork*. PhD thesis, UCL, 2009.

[34] M. Busari and C. Williamson. On the sensitivity of web proxy cache performance to workload characteristics. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1225–1234 vol.3, 2001.

[35] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: network routing inspired by dhts. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 351–362, New York, NY, USA, 2006. ACM.

[36] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. Icp: Design and evaluation of an interest control protocol for content-centric networking. In *IEEE NOMEN (INFOCOM WKSHPS)*, 2012.

[37] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *Proc. of ACM SIGCOMM workshop on Information-Centric Networks (ICN)*, Helsinki, Finland, 2012.

[38] B. Carpenter. Architectural Principles of the Internet. RFC 1958, June 1996.

[39] C. Stephen Carr, Stephen D. Crocker, and Vinton G. Cerf. Host-host communication protocol in the arpa network. In *Proceedings of the May 5-7, 1970, spring joint computer conference*, AFIPS '70 (Spring), pages 589–597, New York, NY, USA, 1970. ACM.

[40] Vinton Cerf, Yogen Dalal, and Carl Sunshine. SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM. RFC 675, December 1974.

[41] J.S. Chase, A.J. Gallatin, and K.G. Yocum. End system optimizations for high-speed tcp. *Communications Magazine, IEEE*, 39(4):68–74, 2001.

[42] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. *SIGCOMM Comput. Commun. Rev.*, 20(4):200–208, August 1990.

[43] K. G. Coffman and A. M. Odlyzko. Handbook of massive data sets. chapter Internet growth: is there a "Moore's law" for data traffic?, pages 47–93. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[44] V. Cuppu, B. Jacob, B. Davis, and T. Mudge. A performance comparison of contemporary dram architectures. In *Computer Architecture, 1999. Proceedings of the 26th International Symposium on*, pages 222–233, 1999.

[45] Asit Dan and Don Towsley. An approximate analysis of the lru and fifo buffer replacement schemes. *SIGMETRICS Perform. Eval. Rev.*, 18(1):143–152, April 1990.

[46] Steve DiBenedetto, Paolo Gasti, Gene Tsudik, and Ersin Uzun. Andana: Anonymous named data networking application. *CoRR*, abs/1112.2205, 2011.

[47] Steven DiBenedetto, Christos Papadopoulos, and Daniel Massey. Routing policies in named data networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 38–43, New York, NY, USA, 2011. ACM.

[48] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2. RFC 5246, August 2008.

[49] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[50] Nandita Dukkipati. *Rate control protocol (rcp): congestion control to make flows complete quickly*. PhD thesis, Stanford, CA, USA, 2008.

[51] Nandita Dukkipati and Nick McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36:59–62, January 2006.

[52] Nandita Dukkipati, Tiziana Refice, Yuchung Cheng, Jerry Chu, Tom Herbert, Amit Agarwal, Arvind Jain, and Natalia Sutin. An argument for increasing tcp's initial congestion window. *SIGCOMM Comput. Commun. Rev.*, 40(3):26–33, June 2010.

[53] R. Braden (ed.). Requirements for internet hosts: Application and support. RFC 1123, October 1989.

[54] R. Braden (ed.). Requirements for internet hosts: Communication layers. RFC 1122, October 1989.

[55] R. Essiambre, G. Kramer, P.J. Winzer, G.J. Foschini, and B. Goebel. Capacity limits of optical fiber networks. *Lightwave Technology, Journal of*, 28(4):662–701, 2010.

[56] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[57] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[58] Kevin Fall and Steve McCanne. You don't know jack about network performance. *Queue*, 3(4):54–59, May 2005.

[59] Bin Fan, John C. S. Lui, and Dah-Ming Chiu. The design trade-offs of bittorrent-like file sharing protocols. *IEEE/ACM Trans. Netw.*, 17(2):365–376, April 2009.

[60] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, June 2000.

[61] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, K.C. Ng, Vyas Sekar, and Scott Shenker. 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, 2013. ACM.

[62] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Quick-Start for TCP and IP. RFC 4782, January 2007. Experimental.

[63] Sally Floyd. Tcp and explicit congestion notification. *SIGCOMM Comput. Commun. Rev.*, 24(5):8–23, October 1994.

[64] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, August 1993.

[65] Annie P. Foong, Thomas R. Huff, Herbert H. Hum, Jaidev P. Patwardhan, and Greg J. Regnier. Tcp performance re-visited. In *In IEEE International Symposium on Performance of Systems and Software*, pages 70–79, 2003.

[66] Armando Fox and Eric A. Brewer. Reducing www latency and bandwidth requirements by real-time distillation. In *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, pages 1445–1456, Amsterdam, The Netherlands, The Netherlands, 1996. Elsevier Science Publishers B. V.

[67] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the internet. *Queue*, 9(11):40, 2011.

[68] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[69] George Gilder. Fiber keeps its promise: Get ready. bandwidth will triple each year for the next 25. In *Forbes*, 1997.

[70] S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *INFOCOM '94. IEEE*, pages 636–646 vol.2, 1994.

[71] Jim Gray. The cost of messages. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, PODC '88, pages 1–7, New York, NY, USA, 1988. ACM.

[72] Jim Gray and Prashant Shenoy. Rules of thumb in data engineering. In *IN PROC. 16TH INTERNAT. CONFERENCE ON DATA ENGINEERING*, pages 3–12, 2000.

[73] Mark Gritter and David R. Cheriton. An architecture for content routing support in the internet. In *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, pages 4–4, Berkeley, CA, USA, 2001. USENIX Association.

[74] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 314–329, New York, NY, USA, 2003. ACM.

[75] P. Krishna Gummadi, Stefan Saroiu, and Steven D. Gribble. A measurement study of napster and gnutella as examples of peer-to-peer file sharing systems. *SIGCOMM Comput. Commun. Rev.*, 32(1):82–82, January 2002.

[76] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[77] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930, March 1996.

[78] Thomas R. Henderson, Mathieu Lacage, George F. Riley, Craig Dowell, and Joseph B. Kopena. Network simulations with the ns-3 simulator. SIGCOMM'08 Demos, August 2008. Code available: http://www.nsnam.org/releases/ns-3.1.tar.bz2.

[79] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[80] Urs Hoelzle. The google gospel of speed. Jan 2012.

[81] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009.

[82] Chi-Yao Hong, Matthew Caesar, and P. Brighten Godfrey. Finishing flows quickly with preemptive scheduling. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 127–138, New York, NY, USA, 2012. ACM.

[83] Christian Huitema. *Routing in the Internet*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[84] IBM. *IBM zEnterprise EC12 Technical Guide*, 2013.

[85] V. Jacobson. Congestion avoidance and control. In *Symposium proceedings on Communications architectures and protocols*, SIGCOMM '88, pages 314–329, New York, NY, USA, 1988. ACM.

[86] Van Jacobson, Marc Mosko, Diana Smetters, and J J Garcia-Luna-Aceves. Content-centric networking: Whitepaper describing future assurable global networks. Response to DARPA RFI SN07-12, 2007.

[87] Van Jacobson, Marc Mosko, Diana Smetters, and J J Garcia-Luna-Aceves. Cisco visual networking index: Forecast and methodology, 2011-2016. White paper, 2012.

[88] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[89] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. Lipsin: line speed publish/subscribe inter-networking. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 195–206, New York, NY, USA, 2009. ACM.

[90] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[91] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02*, pages 89–102, 2002.

[92] Leonard Kleinrock. *Message delay in communication nets with storage*. PhD thesis, Massachusetts Institute of Technology, 1963.

[93] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *SIGCOMM '07*, pages 181–192, 2007.

[94] Teemu Koponen, Scott Shenker, Hari Balakrishnan, Nick Feamster, Igor Ganichev, Ali Ghodsi, P. Brighten Godfrey, Nick McKeown, Guru Parulkar, Barath Raghavan, Jennifer Rexford, Somaya Arianfar, and Dmitriy Kuptsov. Architecting for innovation. *SIGCOMM Comput. Commun. Rev.*, 41(3):24–36.

[95] Swastik Kopparty, Srikanth V. Krishnamurthy, Michalis Faloutsos, and Satish K. Tripathi. Split tcp for mobile ad hoc networks. In *GLOBECOM*, 2002.

[96] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize cdn performance. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 190–201, New York, NY, USA, 2009. ACM.

[97] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, November 2000.

[98] D. Lagutin, K. Visala, A. Zahemszky, T. Burbridge, and G.F. Marias. Roles and security in a publish/subscribe network architecture. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 68–74, 2010.

[99] Nikolaos Laoutaris, Michael Sirivianos, Xiaoyuan Yang, and Pablo Rodriguez. Inter-datacenter bulk transfers with netstitcher. *SIGCOMM Comput. Commun. Rev.*, 41(4):74–85, August 2011.

[100] Tobias Lauinger, Nikolaos Laoutaris, Pablo Rodriguez, Thorsten Strufe, Ernst Biersack, and Engin Kirda. Privacy risks in named data networking: what is the cost of performance? *SIGCOMM Comput. Commun. Rev.*, 42(5):54–57, September 2012.

[101] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies. In *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '99, pages 134–143, New York, NY, USA, 1999. ACM.

[102] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-shaul, and Aviv Shavit. Are file swapping networks cacheable? characterizing p2p traffic. In *In Proc. of the 7th Int. WWW Caching Workshop*, 2002.

[103] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. Brief history of the internet. *Internet Society*, 2003.

[104] Jun Li, Hao Wu, Bin Liu, Jianyuan Lu, Yi Wang, Xin Wang, YanYong Zhang, and Lijun Dong. Popularity-driven coordinated caching in named data networking. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, ANCS '12, pages 15–26, New York, NY, USA, 2012. ACM.

[105] S.-Y.R. Li, R.W. Yeung, and Ning Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, 2003.

[106] J. C. R. Licklider and Welden E. Clark. On-line man-computer communication. In *Proceedings of the May 1-3, 1962, spring joint computer conference*, AIEE-IRE '62 (Spring), pages 113–128, New York, NY, USA, 1962. ACM.

[107] J. C. R. Licklider and Robert W. Taylor. The computer as a communication device. *Science and Technology*, 76:21–31, 1968.

[108] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 359–370, New York, NY, USA, 2012. ACM.

[109] John Markoff. An internet pioneer ponders the next revolution. *The New York Times*, December 1999.

[110] Sally A. McKee. Reflections on the memory wall. In *Proceedings of the 1st conference on Computing frontiers*, CF '04, pages 162–, New York, NY, USA, 2004. ACM.

[111] Aravind Menon and Willy Zwaenepoel. Optimizing tcp receive performance. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 85–98, Berkeley, CA, USA, 2008. USENIX Association.

[112] Jeffrey C. Mogul. Tcp offload is a dumb idea whose time has come. In *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9*, HOTOS'03, pages 5–5, Berkeley, CA, USA, 2003. USENIX Association.

[113] Pablo Molinero-fernandez. *Circuit switching in the Internet*. PhD thesis, Stanford, 2003.

[114] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[115] J. Moy. OSPF Version 2. RFC 2328, April 1998.

[116] P. Newman. Backward explicit congestion notification for atm local area networks. In *Global Telecommunications Conference, 1993, including a Communications Theory Mini-Conference. Technical Program Conference Record, IEEE in Houston. GLOBECOM '93., IEEE*, pages 719–723 vol.2, 1993.

[117] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Queue*, 10(5):20:20–20:34, May 2012.

[118] Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer, Matvey Arye, Steven Y. Ko, Jennifer Rexford, and Michael J. Freedman. Serval: an end-host stack for service-centric networking. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 7–7, Berkeley, CA, USA, 2012. USENIX Association.

[119] Michael F. Nowlan, Nabin Tiwari, Janardhan Iyengar, Syed Obaid Aminy, and Bryan Fordy. Fitting square pegs through round pipes: unordered delivery wire-compatible with tcp and tls. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 28–28, Berkeley, CA, USA, 2012. USENIX Association.

[120] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum. The lru-k page replacement algorithm for database disk buffering. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, pages 297–306, New York, NY, USA, 1993. ACM.

[121] K. Pagiamtzis and A. Sheikholeslami. Content-addressable memory (cam) circuits and architectures: a tutorial and survey. *Solid-State Circuits, IEEE Journal of*, 41(3):712 – 727, march 2006.

[122] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Commun. ACM*, 49(1):101–106, January 2006.

[123] R. Pan, P. Natarajan, F. Baker, B.V. Steeg, M. Prabhu, V. Subramanian, and C. Piglione. Pie: A lightweight control scheme to address the bufferbloat problem. Internet Draft https://tools.ietf.org/html/draft-pan-aqm-pie-02 (Work in Progress), 2014.

[124] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, June 1993.

[125] David A. Patterson. Latency lags bandwith. *Commun. ACM*, 47(10):71–75, October 2004.

[126] K. Pentikousis, B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. Davies, D. Gellert, P. Mahadevan, S. Spyrou, and A. Molinaro. ICN baseline scenarios and evaluation methodology. Internet Draft draft-pentikousis-icn-scenarios (Work in Progress), 2013.

[127] J. Postel. The TCP Maximum Segment Size and Related Topics. RFC 879, November 1983.

[128] Ioannis Psaras, Wei Koong Chai, and George Pavlou. 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, 2012. ACM.

[129] Feng Qian, Alexandre Gerber, Zhuoqing Morley Mao, Subhabrata Sen, Oliver Spatscheck, and Walter Willinger. Tcp revisited: a fresh look at tcp in the wild. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 76–89, New York, NY, USA, 2009. ACM.

[130] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 34(4):367–378, August 2004.

[131] Sivasankar Radhakrishnan, Yuchung Cheng, Jerry Chu, Arvind Jain, and Barath Raghavan. Tcp fast open. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 21:1–21:12, New York, NY, USA, 2011. ACM.

[132] Jarno Rajahalme, Mikko Särelä, Kari Visala, and Janne Riihijärvi. On name-based inter-domain routing. *Comput. Netw.*, 55(4):975–986, March 2011.

[133] Sreeram Ramachandran. Web metrics: Size and number of resources. `https://developers.google.com/speed/articles/web-metrics`, 2010.

[134] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, 2001.

[135] I. Rimac, V. Hilt, M. Tomsu, V. Gurbani, and E. Marocco. A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem. RFC 6029, October 2010.

[136] Luigi Rizzo. Netmap: a novel framework for fast packet i/o. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference*, USENIX ATC'12, pages 9–9, Berkeley, CA, USA, 2012. USENIX Association.

[137] Luigi Rizzo and Lorenzo Vicisano. Replacement policies for a proxy cache. *IEEE/ACM Trans. Netw.*, 8(2):158–170, April 2000.

[138] L. G. Roberts. The evolution of packet switching. *Proceedings of the IEEE*, 66(11):1307–1313, November 1978.

[139] Lawrence G. Roberts and Barry D. Wessler. Computer network development to achieve resource sharing. In *Proceedings of the May 5-7, 1970, spring joint computer conference*, AFIPS '70 (Spring), pages 543–549, New York, NY, USA, 1970. ACM.

[140] Jim Roskind. Quic: Design document and specification rationale. `https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/edit#`, 2013.

[141] D. Rossi and G. Rossini. On sizing ccn content stores by exploiting topological information. In *IEEE NOMEN (INFOCOM WKSHPS)*, 2012.

[142] Christian Esteve Rothenberg, Petri Jokela, Pekka Nikander, Mikko Sarela, and Jukka Ylitalo. Self-routing denial-of-service resistant capabilities using in-packet bloom filters. In *Proceedings of the 2009 European Conference on Computer Network Defense*, EC2ND '09, pages 46–51, Washington, DC, USA, 2009. IEEE Computer Society.

[143] N. Rozhnova and S. Fdida. An effective hop-by-hop interest shaping mechanism for ccn communications. In *IEEE NOMEN'12 Workshop*, pages 322–327, 2012.

[144] Lorenzo Saino, Cosmin Cocora, and George Pavlou. Cctcp:a scalable receiver-driven congestion control protocol for contentcentric networking. In *IEEE ICC*, 2013.

[145] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984.

[146] M. Särelä, T. Rinta-aho, and S. Tarkoma. RTFM: Publish/subscribe internetworking architecture. ICT Mobile Summit, 2008.

[147] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. In *Proceedings of the 5th symposium on Operating systems design and implementation*, OSDI '02, pages 315–327, New York, NY, USA, 2002. ACM.

[148] Ferdinand David Schoeman, editor. *Philosophical Dimensions of Privacy-An Anthology*. Cambridge University Press, 1984.

[149] Ankit Singla, P. Brighten Godfrey, Kevin Fall, Gianluca Iannaccone, and Sylvia Ratnasamy. Scalable routing on flat names. In *Proceedings of the 6th International COnference*, Co-NEXT '10, pages 20:1–20:12, New York, NY, USA, 2010. ACM.

[150] D. K. Smetters and V. Jacobson. Securing network content. Tr-2009-1, PARC, October 2009.

[151] Alan J Smith. Design of cpu cache memories. Technical report, Berkeley, CA, USA, 1987.

[152] Alan Jay Smith. Cache memories. *ACM Computing Surveys*, 14:473–530, 1982.

[153] K. So and R.N. Rechtschaffen. Cache operations by mru change. *Computers, IEEE Transactions on*, 37(6):700–709, 1988.

[154] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *in Usenix Conference Proceedings*, pages 191–202, 1988.

[155] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and Fabián E. Bustamante. Drafting behind akamai (travelocity-based detouring). In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 435–446, New York, NY, USA, 2006. ACM.

[156] John E. Trombly, John D. Foulkes, and David K. Worthington. Audio and full duplex digital data carrier system. US Patent 4,330,687, 1982.

[157] Matteo Varvello, Diego Perino, and Jairo Esteban. Caesar: a content router for high speed forwarding. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ICN '12, pages 73–78, New York, NY, USA, 2012. ACM.

[158] L. Vicisano, J. Crowcroft, and L. Rizzo. Tcp-like congestion control for layered multicast data transfer. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 996–1003 vol.3, 1998.

[159] Ymir Vigfusson, Hussam Abu-Libdeh, Mahesh Balakrishnan, Ken Birman, Robert Burgess, Gregory Chockler, Haoyuan Li, and Yoav Tock. Dr. multicast: Rx for data center communication scalability. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 349–362, New York, NY, USA, 2010. ACM.

[160] Curtis Villamizar and Cheng Song. High performance tcp in ansnet. *SIGCOMM Comput. Commun. Rev.*, 24(5):45–60, October 1994.

[161] Mark Weiser. The Computer for the Twenty-First Century. *Scientific American*, 265(3):94–104, 1991.

[162] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In *ACM Symposium on Operating Systems Principles*, pages 16–31. ACM New York, 1999.

[163] Wm. A. Wulf and Sally A. McKee. Hitting the memory wall: implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, March 1995.

[164] Xipeng Xiao, Alan Hannan, Brook Bailey, and Lionel M Ni. Traffic engineering with mpls in the internet. *Network, IEEE*, 14(2):28–33, 2000.

[165] Haiyong Xie, Yang Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. P4p: Provider portal for applications. In *Proceedings of ACM SIGCOMM*, Seattle, WA, August 2008.

[166] S. Hares Y. Rekhter, T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[167] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive forwarding in named data networking. *SIGCOMM Comput. Commun. Rev.*, 42(3):62–67, June 2012.

[168] Shuchang Zhou. An efficient simulation algorithm for cache of random replacement policy. In *Proceedings of the 2010 IFIP international conference on Network and parallel computing*, NPC'10, pages 144–154, Berlin, Heidelberg, 2010. Springer-Verlag.

Aalto-DD 209/2014

ISBN 978-952-60-6017-0 (printed)
ISBN 978-952-60-6018-7 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Communications and Networking
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS