Computer Science and Engineering

Efficient Methods on Reducing Data Redundancy in the Internet

Sumanta Saha

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 153/2015

Efficient Methods on Reducing Data Redundancy in the Internet

Sumanta Saha

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on the 30th of October, 2015 at 12:00 o'clock noon.

Aalto University
School of Science
Computer Science and Engineering
Data Communication Software

Supervising professor
Professor Antti Ylä-Jääski, Aalto University, Finland

Thesis advisor
Dr. Andrey Lukyanenko, Aalto University, Finland

Preliminary examiners
Professor Thomas C. Schmidt, Hamburg University of Applied Sciences, Germany
Professor George Xylomenos, Athens University of Economics and Business, Greece

Opponent
Professor Mika Ylianttila, University of Oulu, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 153/2015

ISBN 978-952-60-6421-5 (printed)
ISBN 978-952-60-6422-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-6422-2

Unigrafia Oy
Helsinki 2015

Finland

Publication orders (printed book):
sumanta.saha@aalto.fi

Author
Sumanta Saha

Name of the doctoral dissertation
Efficient Methods on Reducing Data Redundancy in the Internet

Publisher School of Science

Unit Computer Science and Engineering

Series Aalto University publication series DOCTORAL DISSERTATIONS 153/2015

Field of research Data Communication Software

Manuscript submitted 15 June 2015 **Date of the defence** 30 October 2015

Permission to publish granted (date) 20 August 2015 **Language** English

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

Abstract

The transformation of the Internet from a client-server based paradigm to a content-based one has led to many of the fundamental network designs becoming outdated. The increase in user-generated contents, instant sharing, flash popularity, etc., brings forward the needs for designing an Internet which is ready for these and can handle the needs of the small-scale content providers. The Internet, as of today, carries and stores a large amount of duplicate, redundant data, primarily due to a lack of duplication detection mechanisms and caching principles. This redundancy costs the network in different ways: it consumes energy from the network elements that need to process the extra data; it makes the network caches store duplicate data, thus causing the tail of the data distribution to be swapped out of the caches; and it causes the content-servers to be loaded more as they have to always serve the less popular contents.

In this dissertation, we have analyzed the aforementioned phenomena and proposed several methods to reduce the redundancy of the network at a low cost. The proposals involve different approaches to do so--including data chunk level redundancy detection and elimination, rerouting-based caching mechanisms in information-centric networks, and energy-aware content distribution techniques. Using these approaches, we have demonstrated how we can perform redundancy elimination using a low overhead and low processing power. We have also demonstrated that by using local or global cooperation methods, we can increase the storage efficiency of the existing caches many-fold. In addition to that, this work shows that it is possible to reduce a sizable amount of traffic from the core network using collaborative content download mechanisms, while reducing client devices' energy consumption simultaneously.

Keywords cache; redundancy; energy; ICN; Internet

ISBN (printed) 978-952-60-6421-5 **ISBN (pdf)** 978-952-60-6422-2

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2015

Pages 148 **urn** http://urn.fi/URN:ISBN:978-952-60-6422-2

Preface

This dissertation represents a culmination of work and learning, which has taken place during last five years from 2010 to 2015 in the Data Communication Software Lab, Aalto University School of Science. The work was carried out part-time alongside a full-time day job at Nokia and then Ericsson. In this way, I was able to follow my research interest while still keeping a close contact with the industry. I am grateful to both the companies for allowing me to pursue my research career.

The thesis is article based and every published article was co-authored. My primary task has been to define problems and algorithm to solve the problems. To that extent, I did literature study, analyzed problems, defined algorithm to solve the problem, developed prototypes, performed measurements, and analyzed the measurement results. My co-authors has assisted my in various phases of the work ranging from defining the problem to performing the measurements.

I would like to pay gratitude to my supervisor Prof. Antti Ylä-Jääski for believing in me and allowing me to explore my own research interests. He was always there to support me in every possible ways towards the completion of this thesis.

Thanks to my instructor, Dr. Andrey Lukyanenko, for his immense support towards the degree. He has been instrumental in defining the research goals and formulating the solution proposals. Without his guidance and coaching, it would have been very difficult to achieve the goal. I am grateful to him for sharing his experience and expertise in the work.

The comments of Prof. Thomas C. Schmidt of Hamburg University of Applied Sciences and Prof. George Xylomenos of Athens University of Economics and Business as the pre-examiners of this dissertation were valuable and encouraging. The comments were helpful to improve the quality of the thesis. I would also like to extend my gratitude to Prof.

Sasu Tarkoma for showing interest in my work many times and for believing in me.

I also want to extend my gratitude and thanks to my co-authors, including Dr. Ashraful Hoque and Maneesh Chauhan, for helping me out in completing the scientific publications which contributed towards the degree. Without their help it would not have been possible for me to publish quality papers.

I want to thank the department secretaries and university administrators for creating excellent working environment. I would like to mention Soili Adolfsson, Maarit Vuorio and Emma Holmlund for their support during my work. It was because of their sincere help that I was able to concentrate on the research.

Thanks to my friends in Otaniemi, Espoo. They have been instrumental in providing me with all the excitement and social interactions needed to make the difficult journey easier. Without them, life abroad would have been a lot more lonely and boring.

Finally, and most importantly, I am grateful to my family. My parents and my beloved wife have been ever encouraging and supportive during my doctoral studies. They always wanted me to achieve this goal and supported me in every way possible. They have lifted me up when I was down, and rejoiced with me during my success. I am dedicating this dissertation to them.

Espoo, Finland, October 1, 2015,

Sumanta Saha

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 Sumanta Saha, Andrey Lukyanenko, Antti Ylä-Jääski. CombiHeader: Minimizing the number of shim headers in redundancy elimination systems. In *IEEE INFOCOM Workshops*, Shanghai, China, 798-803, doi: 10.1109/INFCOMW.2011.5928920, April 2011.

II Sumanta Saha. On Reducing the Processing Load of Redundancy Elimination Algorithms. In *IEEE GLOBECOM Workshops*, Texas, USA, 1106-1110, doi: 10.1109/GLOCOMW.2011.6162349, December 2011.

III Sumanta Saha, Andrey Lukyanenko, Antti Ylä-Jääski. Cooperative Caching through Routing Control in Information-Centric Networks. In *IEEE INFOCOM*, Turin, Italy, 100-104, doi: 10.1109/INFCOM.2013.6566743, April 2013.

IV Sumanta Saha, Andrey Lukyanenko, Antti Ylä-Jääski. Efficient Cache Management in Information-Centric Networks. *Computer Networks*, vol:84, issn:1389-1286, 32-45, doi: http:// dx.doi.org/10.1016/j.comnet.2015.04.005, July 2015.

V Sumanta Saha, Maneesh Chauhan, Andrey Lukyanenko. Beyond the Limits: Maximization of ICN Caching Capabilities with Global Detour Algorithm. In *IEEE Symposium on Computers and Communications*, Larnaca, Cyprus, July 2015.

VI Sumanta Saha, Mohammad Hoque, Andrey Lukyanenko. Analyzing Energy Efficiency of a Cooperative Content Distribution Technique. In *IEEE GLOBECOM*, Texas, USA, 1-6, doi: 10.1109/GLOCOM.2011.6133841, December 2011.

Author's Contribution

Publication I: "CombiHeader: Minimizing the number of shim headers in redundancy elimination systems"

The author was the primary contributor to the conceptualization of the paper along with the co-authors. He also performed the prototyping and simulation effort needed for the research.

Publication II: "On Reducing the Processing Load of Redundancy Elimination Algorithms"

The author was the primary contributor to the conceptualization of the paper, and was the primary contributor to the prototyping and simulation effort.

Publication III: "Cooperative Caching through Routing Control in Information-Centric Networks"

The author conceptualized the paper with the co-authors, and was the primary contributor to the validation and simulation effort.

Publication IV: "Efficient Cache Management in Information-Centric Networks"

The key idea was developed primarily by the author with the co-authors. In addition, the author also implemented and measured the necessary simulation results to validate the work.

Publication V: "Beyond the Limits: Maximization of ICN Caching Capabilities with Global Detour Algorithm"

The author was a key contributor to the conceptualization of the paper and formulating the paper content. He also developed the core modules of the prorotype.

Publication VI: "Analyzing Energy Efficiency of a Cooperative Content Distribution Technique"

The author developed the concept of the paper along with the co-authors, and was the primary contributor to the prototyping effort.

List of Figures

List of Abbreviations

AS	Autonomous System
BGP	Border Gateway Protocol
BRITE	Boston university Representative Internet Topology Generator
CAIDA	Center for Applied Internet Data Analysis
CDN	Content Delivery Network
CEE	Cache Everything Everywhere
DARPA	Defense Advanced Research Projects Agency
DC	Data Center
DV	Distance Vector
ICN	Information Centric Networking
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IS-IS	Intermediate System-to-Intermediate System
ITU	International Telecommunication Union
LRU	Least Recently Used
LS	Link State
OSPF	Open Shortest Path First
PSM	Power Saving Mode
RAND	Research ANd Development
RE	Redundancy Elimination
RIP	Routing Information Protocol
RRC	Radio Resource Control
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Unified Resource Identifier
VoD	Video on Demand

1. Introduction

The Internet has witnessed an unprecedented growth in both scale and capacity over the last couple of decades. This network of machines, which originally served to connect a few government offices and laboratories, now serves millions of users around the world. It is now being used as a world-wide broadcasting service, an information dissemination mechanism, and as a collaboration service between individuals without geographic restriction [1]. To perform such broad functions on a global scale, the Internet has evolved from a client-server based computing machine to a data-based planet-scale sharing tool. Two things accelerated this evolution: the social media and mobile technology. These two developments have caused the Internet to be at the heart of our personal, social, and business communications, resulting in data growth on a huge scale. According to the International Telecommunication Union (ITU), the Internet was used by 2.5 billion people in 2011 [133], while in the same year Cisco estimated global Internet traffic to be 31 Exabytes per month [30], with a global IP traffic projection of 110 Exabytes per month by 2016.

With this scale of growth, the Internet is also changing its nature. When it was first conceptualized in a series of memos [78] written by J.C.R. Licklider explaining his "Galactic Network" concept, the Internet was thought of as a globally connected set of computers through which everyone can access necessary data. The idea later flourished through the work of DARPA [37], and with the help of revolutionary work by Kleinrock [69, 70] on packet switching theory. In addition to that, parallel work at the RAND group [18] and the NPL group [39, 38] also furthered the idea of interconnected machines for distributed communication independently. Getting access to important information stored in secure servers in an intercon-

[1] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet

nected network was the primary use of the Internet for decades. The mechanics of the network were designed and engineered with this application in mind. However, with time and with the emergence of new applications over the Internet, the nature of usage has changed. From being a tool to communicate with remote servers for performing processor intensive jobs and retrieving the results, the Internet has become a tool to share and access content irrespective of its location. Thus, from being a client-server model, the Internet has evolved into what is called a content-eyeball-transit [73] model.

To enable the refreshed usage of the Internet, the mechanics used as building blocks also evolved. Whereas, powerful servers were once well-defined with respect to their location, the location is now becoming less and less important. Application layer search engines (e.g. Google, Bing, Yahoo, etc.) do the heavy lifting in terms of finding the location of a particular content by conveniently crawling through the whole Internet so that the end users can be aware of the content name without necessarily knowing its location in the network. These search engines perform their task by indexing all the content in the Internet along with its location, and performing a search on this database upon request. The underlying network is not aware of content, and the routing and switching mechanism just works on the address of the server which hosts the content. In addition to this, the content providers have also resorted to different solutions to make content more readily available for the users. Solutions such as web cache, content delivery network (CDN) are some examples.

A web cache is a temporary storage usually set up by network administrators and is used to reduce bandwidth usage, server load, and lag time. A cache is usually set up as a bump-in-the-wire device where it intercepts web requests and stores the response content locally so that the cache can serve any future request for the same content without forwarding the request upstream. Web caches are usually local, and content providers do not have any control over the local caches. On the other hand, the content providers also strive to increase the availability of their content and decrease the perceived lag from the user's perspective. CDN–an interconnected set of caches placed in geographically strategic positions to deliver content fast and efficiently–usually serves this purpose. It effectively duplicates the content and places it in scattered geographical locations in order to use proximity to the best advantage. Content owners usually purchase the services from CDN providers and let their content be dupli-

cated in different servers of the CDN. Subsequently, any request for the content is redirected to the CDN servers and served from there, relieving the original content server.

While web caches and CDNs are widely deployed and used in commercial content delivery, Internet researchers are busy trying to reinvent the Internet to better suit its renewed use. Information Centric Networks (ICN) [7] is one such clean-slate design of the Internet where the content is at the center of any operation. ICN takes into account the fact that the users are not interested in server addresses any longer; rather the content is what they want. To cater to this need, ICN performs routing based on the content name instead of the address. This idea removes the need for application layer search engines as the network itself will perform the search for the content and fetch it for the user. There have been many contemporary ICN proposals [71, 36, 63, 109], each having their own architecture and primitives. However, the common part of each proposal is how the information in the Internet is identified. Instead of the typical IP addresses and server based URLs, ICN uses unique identifiers for each piece of content irrespective of where it originated from. Most of the ICN proposals also have the functionality to generate the identifiers in such way that the same contents will always get the same identifier, regardless of whether they originate from the same server. Many proposed ICN architectures also allow built-in caches [84, 68, 66] where content fragments are cached in the on-path routers and are served from the cache when necessary. ICN caches are intrinsically different from those of the current Internet. Since the contents in the ICN are identified by a content name and not by the server address, a particular piece of content can still be served to the end user from the cache, even if the original content server is offline. Additionally, as the caching schemes of the ICN usually retain content based on popularity, the availability of a piece of content can be almost guaranteed as long as its popularity exists above a certain threshold. A similar feature is also achieved in the current Internet by means of CDNs, where the original content server does not have to be online to make the content available. However, CDN is not a feature of the network itself; rather it is an overlay.

Most of the above caching schemes in the contemporary Internet and in ICN depend on caching on the path of the communication[2]. While this

[2]On-path caching usually means caching on the cheapest path from the end-user to the content server, while off-path means outside the default path from

mode of operation performs better in terms of responsiveness and delay, it demands heavy duplication of the cached content so that most of the on-path communications are through the cache.

Although caching and other techniques such as CDN work quite well in distributing content across the world, these techniques are mostly designed for unicast traffic where the content is delivered to one person at a time. However, in many cases multicast, where the content is simultaneously delivered to multiple parties, is more efficient than unicast. In the cases of content broadcast (e.g. TV and radio channels), popular content delivery, etc., multicasting allows better use of the network bandwidth by transmitting only one copy of the content as far as possible. The standardized solution for content multicasting is IP multicasting [42, 56, 33]. However, the standards require that the on-path network elements support multicasting, which renders its use quite limited as the support is not widespread, and without support throughout the packet path, it does not work. Consequently, many multicasting solutions depend on application layer multicasting to ensure that there is no dependency on the network elements in the middle. There has been a lot of research done in the past decade to explore and analyze the efficacy of different application layer multicasting content delivery methods, including server based ones and peer to peer (P2P) solutions [57, 117, 124]. In addition to the bandwidth saving properties of such multicasting technologies, these methods also affect the overall power consumption level of the processing network elements (e.g. routers, due to less packet processing during content delivery) and the data centers storing them (as it is not necessary to store copies of the same data in multiple data centers). The power usage for data centers has grown exponentially, with data centers being responsible for up to 1.5% of total US electricity consumption in 2007 according to the US EPA [135]. Moreover, by 2012, the U.S. Department of Energy expects the cost of power for data centers to exceed the cost of the original capital investment [134]. Effective use of a multicasting system allows the content distributors to cut the power usage of the data centers, and the end users to use an energy efficient network interface to download content.

In recent years, with the exponential increase in content consumption, the load on the data centers for CDNs and web caches has increased. Although, a major portion of the content follows the “long tail” effect [14], a time limited temporal effect causing content to be dramatically popular

the end-user to the server.

for a short period of time before sliding back to the tail end of the long tail curve. Content providers can predict and utilize this temporal effect and benefit in terms of downlink bandwidth savings and energy preservation.

1.1 Motivation

Discussion in the previous section has revealed that both the core and the edge networks carry excessive amount of redundant data to provide end users a consistent and lag-free experience. The cost behind this redundancy is manifested in a higher bandwidth requirement in network dimensioning, higher storage requirements for the caches and data centers, a cost increase in leasing bandwidth from upper tier providers, higher processing power consumption for the network elements, and higher energy consumption for processing redundant traffic.

Redundant traffic comes in different flavors. Application layer redundancy is the easiest to detect as the whole application layer object (e.g. music file, text file, video file, etc.) is duplicated, and it can mostly be detected by inspecting the object names. There has been numerous efforts to optimize such detection [104, 85, 9, 102, 101], both in web caching and in the CDN area. However, there has been a less explored variant of data redundancy where the detection is not so straightforward. Such redundancy is called the packet level redundancy or data level redundancy as this can only be detected by going under the hood and inspecting the raw data chunks for partial matches. There have been a few research activities done around this type of redundancy detection [11, 10, 126, 120]. However, the research was limited and did not explore the full possibilities of such redundancy detection and elimination processes. Nevertheless, the work has been very successful in igniting the interest of the research community in independent redundancy elimination. Anand et. al. have shown that the application-independent redundancy detection can detect intrinsic similarities between two completely unrelated contents [11], and it is possible to significantly reduce the load of the core network links by detecting such redundancy and then removing it from the traffic. For a transparent end user experience, the reduced traffic is usually reconstructed downstream by using cached pieces of content from earlier traffic. Although the efficacy of such redundancy elimination has been shown in existing research works, the optimal use of it and the side effects of such elimination processes were not analyzed completely. To be more spe-

cific, redundancy elimination in such a way requires a certain amount of metadata to be attached to each IP packet [10], which then consumes the expensive bandwidth of the core network. Moreover, to process such reduced data flow, each of the core routers needs to spend valuable processing power to eliminate and then reconstruct the traffic. The effect of such processing and the overhead metadata is not clearly understood at this time. This thesis presents an analysis of the overhead incurred due to the redundancy elimination process as discussed above, and proposes a novel algorithm to reduce the overhead. In addition, this work also analyzes the processing need for such an operation, and demonstrates how the proposed algorithm reduces the processing need.

In addition to the redundancy elimination processes for existing IP traffic in the Internet, the caching processes of the clean slate Internet paradigms, such as the Information Centric Networking (ICN), are also reviewed. ICN proposals usually have caching schemes built in within the network [63]; however, to deliver a better response time to the end users, the cache space is utilized non-optimally, by filling the caches with duplicate pieces of popular data. While this approach increases the end-user perception of a lag-free network, it does not allow the use of the total cache space available for the network. Instead of storing duplicate data in many places, controlling the storage mechanism will mean more content can be delivered from the cache, and the load of the origin server can be reduced. This thesis work proposes such a cache network architecture where the cache content duplication is minimized.

Moreover, as discussed in previous sections, the energy consumption of the equipment connected to the Internet, including the routers, switches, data centers, and end user terminals, consume a sizable amount of energy today. By reducing the redundancy in the Internet, it is also possible to reduce this power consumption, both in the routing and switching equipment as well as in the end user equipment. While we can reduce the power consumption in the core devices by reducing processed traffic, we can also reduce the consumption at the end devices by making sure that less power hungry interfaces are used to download data. Both of these goals can be achieved by first introducing something such as IP multicast, and then forcing end users to use less power hungry network interfaces.

1.2 Problem Scope

The scope of this thesis work is to address the problem of redundant data in the Internet in the form of duplicated data in the traffic flowing between two parties, and in the form of duplicated data in the caches around the world. The work looks at these problems from both the traditional TCP/IP based Internet perspective and, in addition, the newer clean-slate ICN point of view.

In the case of the traditional Internet, the caching schemes are not embedded in the network (they are usually overlays), and thus are optimized separately from the network. This study concentrates on improving the traffic redundancy elimination processes in the case of the traditional Internet. Existing solutions use substantial amount of metadata to enable the downstream nodes to reconstruct the data. This work takes a closer and more detailed look than that of typical caching overlays to discover data level redundancy among traffic flows. That redundancy is then eliminated using a process which minimizes the processing and resource need of the routing boxes.

On the other hand, information centric networks have built-in caching schemes, which do not depend on any application layer overlays. However, due to extensive in-network caching, the duplicacy among the caches is high, causing a small proportion of data to occupy most of the cache space. This duplication is found both within the autonomous system and across them. This work addresses the problem of excess redundancy in the cache space of ICN and proposes ideas to maximize the cache availability parameter of the network by the means of detours and clustering. The solution proposal consists of both an intra-domain and an inter-domain part, where either the autonomous systems implement the caching improvements locally, or the systems cooperate with each other to improve caching parameters globally.

In addition, this dissertation also considers the current trend of large scale content distribution and the abundance of handheld mobile devices, and explores the problem space of energy efficiency in relation to the content consumption by end users. Recent trends in Internet content consumption indicate a move to bigger content with higher quality while the data speed of the air interfaces such as cellular networks is rising to be on par. However, the capacity of the batteries in the cellular devices is not increasing at the same rate to support these power hungry high-bandwidth

interfaces. Thus, a common problem with today's cellular devices is battery depletion. In addition, large scale content delivery to end- users without proper optimization can also fill the pipe of the core network with duplicate deliveries of the same content to multiple end-users. We also, therefore, look at the possibilities of reducing this duplicacy while inspecting the effect of such optimization on the battery life of the end devices.

1.3 Methodology

Analyzing a problem space and addressing a problem often can be done in two different ways: by theoretically modeling the problem and mathematically solving it or by practically simulating the problem and then testing different solution methods to measure how the solutions work in comparison with the original. In this thesis, we employ the latter.

Extensive background research was performed to understand the current solution space for the Internet data flow and the amount of undetected redundancy in it. In addition, the solution proposals for the elimination of such redundancy were also reviewed. Using the input from the study, this work then formulated an algorithm to optimize the redundancy elimination mechanisms. A similar approach is also taken to understand and analyze the current caching mechanisms in the ICN space. With that background in mind, this work proposes a caching network architecture and content retrieval algorithm to increase the cache capacity of the network.

The proposed algorithms and architectures are then verified by a custom simulator which simulates a very large network similar to the Internet, and then routes content requests and responses between node pairs. Different measurement points can be defined in the simulator, which collects several parameters of the routing operation such as the hop count, server hit, cache hit, etc. In addition, several other contemporary caching schemes were also implemented in the simulator for comparison purposes.

A network topology generator such as BRITE [82] was used for micro experiments where different network topologies with specific properties were used to demonstrate the capability of the proposed algorithms. Real Internet topology from the CAIDA [58] Skitter dataset was used for macro level measurements to demonstrate the applicability of the proposals in a large scale network.

The size of the cache per node was varied according to the experimental

need. We have also explored how different cache sizes affect the proposed algorithm. However, in most cases, when drawing conclusion from the experiments, we have tried to keep our cache sizes realistic (taking into account research done on real world traffic, e.g., [50]).

In addition, for energy efficiency related experiments, the experiment methods are comprised of practically measuring the energy consumption of real world devices while experiments are ongoing. These practical results are then compared with results derived from ideal consumption models to reach a conclusion about the validity of the proposed method.

1.4 Contributions

The primary contributions of the thesis work are in its novel proposals for reducing data redundancy from different parts of the Internet. We have presented lightweight, energy efficient algorithms to address redundancies in different parts of the network. We have explored novel caching mechanisms to reduce redundancy in more recent clean-slate Internet architectures such as the ICN. In addition to that, we have performed simulations at a micro and macro level to demonstrate the applicability of the proposed methods in different kinds of networks. The effect of data redundancy on the energy consumption of hand-held devices was also analyzed. The results of the research were presented in six different peer reviewed publications.

Publication I presented a method of redundancy elimination for the current Internet which is application independent, and reduces the need for metadata in the redundancy elimination technique. The metadata processing algorithm is designed so that, irrespective of the application layer properties, duplicate data can be removed in the upstream routers, and the data flow can be reconstructed at the downstream routers. In addition, the amount of information needed in the metadata was reduced to optimize the overhead required for such elimination techniques. An extensive set of measurements were done to demonstrate the applicability of such an algorithm in the network.

Publication II extends the aforementioned redundancy elimination technique and analyzes the algorithm from a resource utilization point of view. The paper shows how the proposed algorithm reduces the processing need of the intermediate routers while also lowering the memory access delay.

A novel caching technique is proposed in Publication III to reduce the

amount of duplicated data stored in the cache. The technique is applicable to the clean-slate Internet paradigm called the information centric network. To enable gradual deployment of such cache optimization, the work proposes an autonomous system (AS) level deployment plan. Through this method, the content of the cache is distributed across the network in such a way that instead of duplicating a particular piece of content many times in the cache network to increase availability, the content is stored once in a deterministic way to allow optimized use of the storage capacity. Using this deterministic property, the routing detour is then used to retrieve the data and maximize the cache hit. Publication IV extends this concept and explores the possibilities of routing detours where the content requests and responses are routed through non-optimal paths to increase cache availability and hit rate. The paper presents extensive experiment results analyzing the effect of different versions of the detour mechanism. Publication V further elaborates on the idea of the detour and tries to solve the problem of increased path length and possible congestion. To this extent, this paper explores the possibilities of clustering in the caching network so that the cache contents are deterministically duplicated within the Internet. This should allow the content requestors to find the nearest copy of the content in the cache and fetch it from there.

While redundancy elimination techniques and clever caching techniques offer us a lowered volume of data in the core network, content distribution techniques can also have an impact on the energy efficiency of handheld end devices. To understand this, a content distribution technique is proposed in Publication VI, which uses a collaborative content distribution technique to reduce the use of energy-hungry interfaces in favor of more efficient interfaces. The delivery of the content is optimized in favor of minimizing duplicate downstream traffic. Thus, the proposed method reduces duplicacy in downstream traffic and, in addition, reduces the energy efficiency of the participating clients. We performed extensive measurements with real devices to understand the effect of such changes on the power consumption of the devices and presented them in our publication. In addition, we also proposed an economic model for the content distribution mechanism where the end users are rewarded for choosing the proposed method.

1.5 Structure of the Thesis

The thesis is structured as a summary narrative of the work done during the duration of the degree. We present a short introduction of the overall work in Chapter 1. Then in Chapter 2 we present a thorough account of the prior work done on this topic. Chapter 3 summarizes the overall contribution this thesis presents.

2. Background

This doctoral dissertation presents some efficient methods for reducing data redundancy in the Internet. To provide a context for the reader concerning the overall field of the dissertation, this chapter discusses the Internet in general and then delves deep into the subject of data flow, and data redundancy in the Internet. We also present the general data dissemination models in the Internet along with necessary background information on Information Centric Networking, a new paradigm for data distribution. This chapter should provide the reader with the necessary information to understand different types of redundant data in the Internet and their implications.

2.1 The Internet

The Internet is a network of interconnected machines, routers, servers, mobile devices, and more recently many consumer devices. It has revolutionized the communications world from its very roots. It serves as a worldwide multicasting technology, an information dissemination system, a collaboration medium, a social interaction tool, and a universal means of mobile communication. The idea that started as a series of memos by J.C.R. Licklider of MIT [78] resulted eventually in ARPANET [116], ending up as a series of interconnected machines on a global scale. During the early period of the Internet, several research groups such as the ones at MIT [78], at RAND [18], and at NPL [39] have independently started to develop the idea of packet switching. The concepts were developed further in ARPANET, and finally researchers demonstrated a successful working packet-switched network in 1972. From that point on, the Internet has become one of the most significant mediums of human communication in history, and one of the most widely-used data storage mechanisms.

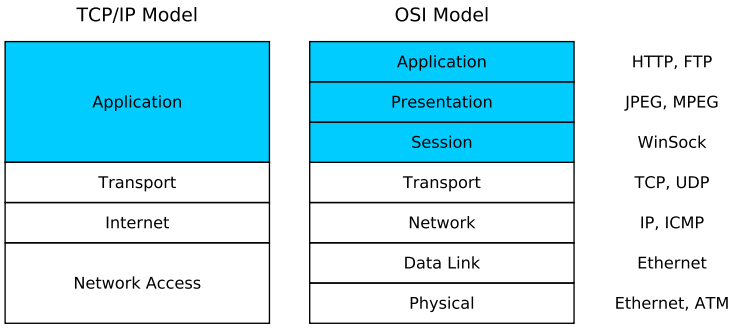

Figure 2.1. OSI and TCP/IP Models and the protocols used in the different layers

In the modern world, the Internet is used for personal communication (email, social networks), for entertainment (VoD, youtube, online radio), for data sharing (photo, video), for data storage (cloud storage), for computation (cloud computing), and for many other things. And all these are based on the simple but elegant design of packet switching. In addition to the packet switching technology (which is defined in a protocol called Internet Protocol or IP [61, 41]), there are several layers of protocols running to ensure that this planet-scale distribution system works like it does today. The stack of these protocols are defined according to either the OSI model [151] or the TCP/IP model [123]. Both of these models specify layers of protocols that the end clients need to implement to successfully communicate with each other. The TCP/IP model is mostly a simplified derivative of the OSI model and is more relevant to the current Internet. The relationship between the two models along with the protocols used in each of the layers is depicted in Figure 2.1. Thus, when talking about the Internet, we don't talk only about the physical network that encompasses the whole world, but instead about the entire stack of protocols that enable us to share data with each other.

2.2 The Internet Architecture

Everything in the Internet, whether it is a photo or a music file, is represented by data packets. Several protocols such as ALOHANet [24], Ethernet [83], IP [61, 41], TCP [106], UDP [105], etc., has been defined to facilitate interoperability between the machines participating in data communication. In the following subsections, we will briefly discuss the usual norms of the data communication in the Internet, and then describe the

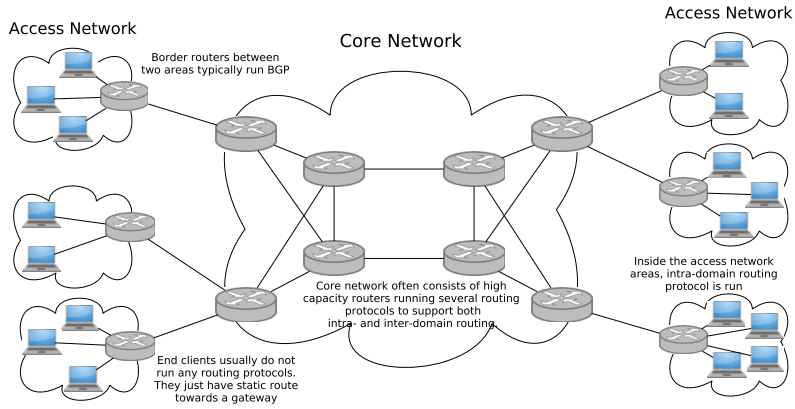

Figure 2.2. Core and Access Network

divisions of data network according to their role in data communication. In addition to that, we will also discuss the communication model and its evolution over time.

2.2.1 Core Network

The Internet was built by connecting small local area networks together. These connecting links are collectively called the core network of the Internet. Core network links are usually high capacity communication links under the administration of network operators. This is the central part of the whole network and facilitates data communication between the service providers. This network carries the aggregation of the data served by the access networks to the consumers. Thus, the total connectivity and bandwidth requirement are much higher compared to a single access network. A simple visualization is shown in Figure 2.2.

To transport the incoming traffic towards the correct destination, the core network runs several routing protocols. Some protocols are optimized for working well in smaller domains, while others are more suitable for aggregating large scale routing information. We will briefly discuss some routing algorithms in subsequent sections.

The aggregation of the traffic from many access networks causes a lot of redundancy in the core network. As many of the content requests made from the access network are requesting the same or similar content, a core network without any redundancy detection serves the same content many times over, causing bandwidth consumption of duplicate traffic. This situation is less problematic in access networks because usually access net-

works are carrying traffic for a small set of subscribers and, thus, the duplicacy is small. Moreover, as the links in core network carry substantial amount of traffic from many end users, optimizing them also carries a higher benefit.

2.2.2 Routing Protocols

Routers on the path of a data packet have the job of running routing algorithms or protocols to route the packet to the correct destination via the best path available. There are usually multiple paths from an origin to the destination, and the best route is chosen according to different parameters such as the hop count, link speed, or cost of the path. Using routing protocols, the routers gather information about the state of the network. Such gathering can be done in two different ways: either the routers know about the routers directly neighboring them, and they just share information with them, or the routers gather a summary of information about the whole Internet, and use that to plan a route across the network. The first family of routing algorithms is called a distance vector (DV), while the latter is called a link state (LS) algorithm. The DV algorithms are based on Bellman-Ford [20] and Ford-Fulkerson algorithms, while the LS algorithms are based on the Dijkstra algorithm [44]. However, the information tends to grow very quickly and the routing protocols based on DV and LS get overwhelmed. To avoid this, the concept of hierarchical routing was introduced. In hierarchical routing, routers are classified in groups known as regions. The routers inside a region have information about only the routers inside the region. The routers on the border of the regions gather knowledge about other regions in a compact form. Thus, the routing is divided into two different functionalities: intra-domain routing, and inter-domain routing.

A typical example of intra-domain routing is OSPF [92], which is a LS algorithm, and uses Hello messages to discover all the routers within a domain through link state advertisements. Using these advertisements each router of the domain creates the shortest path tree to every other router. The tree connectivities are validated after each fixed time interval. To support big domains, OSPF also has the concept of hierarchy built into it. OSPF can divide a particular domain into areas, and then operate inside the areas. The routing information of an area is summarized and exchanged with other areas. There are other well-known and widely used intra-domain routing protocols such as RIP, IS-IS [99], etc. Each has its

advantages and disadvantages. However, OSPF is the most used one.

Regarding inter-domain routing, Border Gateway Protocol or BGP [115] is the protocol of choice. BGP is a path vector protocol, meaning it provides the whole path a particular packet should take to reach its destination. However, it can be intuitively understood that, the path length can be very long if the packet is sent across the world, and for a source router to know the ID of all the routers across the whole path is not a realistic option either. However, this is actually not a problem, as BGP does not deal with router level path vectors, but works with autonomous systems or AS (or the domain we talked about in the previous paragraph) level paths. Thus, BGP defines which ASes a packet should pass through to reach the destination. Routing inside each of those ASes will be determined by the intra-domain routing running inside. To be able to handle the huge amount of routes across the world, BGP works only with summary routes where a very condensed piece of information specifies what a particular AS can serve to the world. This information gets summarized even more as the information propagates upwards in the domain ladder.

Using this intra- and inter-domain routing, the core network is able to route the data from every corner of the world to any destination without delay.

2.2.3 Access and Edge Network

An access network is the part of the network which connects the subscribers with their service providers. In many cases, the access network consists only of the part which is on the consumer side, whereas the part on the access provider side that connects with the consumer is called the edge network. An access network usually has a comparatively low bandwidth requirement and carries data for individual customers. Thus, the requirements for access network routers and other equipment are also different from the rest of the network. Access networks often also are comprised of authentication and authorization of the end user to validate the end-users' access rights to the network and the class of service to be provided.

2.2.4 Data Flow

Now that we have discussed the parts of the network that carries the data, let us take a look at how the data flows within the network. With

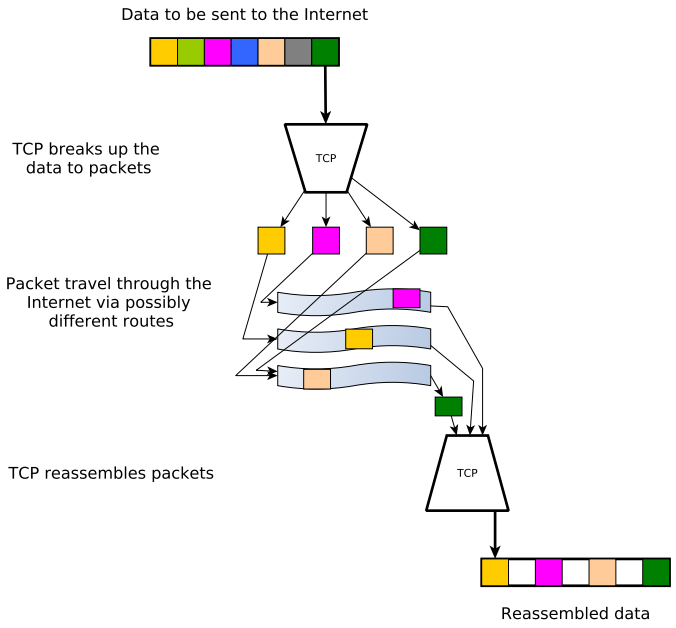

Figure 2.3. Data flow mechanism in the Internet

the help of the TCP/IP protocol suit [61, 106], the Internet was designed to be highly robust against failure of any section of the network. The data to be sent over the network is first divided into smaller chunks called packets and then transmitted over the available routes to the destination. The division procedure is based on the TCP/IP protocol suits and depends on the capacity of the underlying physical network. Usually the physical network has a capacity to carry a maximal sized packet, and the division process respects that. The routing of packets to the destination operates in many ways like a road network where an intelligent traffic system controls which path will be taken by the incoming traffic, taking into account congestion and cost. Data packets are routed through the least expensive path to the destination, and the routing decision is made per hop, that is, instead of making a decision about the whole path from source to destination at the start, the path choice is done at each step of the routing process. A good analogy can be made with the postal system. When delivering a post, usually the postal system just decides on the best route to deliver to the next responsible post office instead of deciding on the whole route. This distributed decision process makes the system lightweight, more responsive to local congestion scenarios, and requires less internal communication. The traffic police in the road analogy are equivalent to

the router in the Internet. The router takes the decision about routing the packets via the correct path to the next hop. Routers are also knowledgeable concerning the routing policies that the operator wants to follow. This allows the routers to operate in a very intelligent and economical way. Several application layer protocols, such as OSPF [92], IS-IS [99] and BGP [115], dictate the information flow between the routers to ensure that the routers are apprised of the routing situation elsewhere in the Internet topology. However, these protocols do not directly affect the transportation process of the data packets. In our previous postal system analogy, we can think of the role of OSPF or BGP as an information sharing system whereby the postal workers become familiar with the routes within a particular city and the routes between it and other cities. The transportation of the data packets is governed by protocols such as Ethernet, IP and TCP. They make sure that the packets are delivered to the correct address, and in many cases also ensure reliability of delivery. For example, in case of a link failure the data is just sent over another section of the Internet. Upon receipt of the data packets, the receiver reconstructs the content based on the information encoded in the protocol headers. Any failed or corrupt packet can be resent over the alternate paths. A visual demonstration of the data flow in the Internet is shown in Figure 2.3.

The network which carries the data consists of different types of routers, switches, and other network nodes linked by Ethernet links, fibers, microwave links, etc. For the sake of easy management and better understandability, the network is divided into various logical parts such as, the Access network, the Core network, and the Edge network. In addition to these network types, the data communication model between the consumer and the producer can also be categorized according to this procedure. We will discuss this more in the coming sections.

2.2.5 Client Server Model

The Internet, as an original concept, was started as a medium of client-server communication, where the server provided centralized data and computing power and was located at a fixed address, while the client consumed that service by contacting the server's address. In this model, the server and its address were the most important information for the client rather than the data that it wanted to consume [149].

This model of communication was very relevant during the period when client machines were not very powerful and thus had to rely mostly on

powerful centralized computers to provide services. The thin client model was based on a very restricted availability of equipment with high computing power. The cost and maintenance of those high end devices also excluded general people from owning them. Thus, thin client machines relied on buying services from a few powerful servers. However, the emergence of more powerful and lower-priced machines caused this model to quickly disappear. The reliance on servers, nonetheless, still exists today although with the (key) difference that the client devices now rely on the servers for content instead of computing power.

2.2.6 Content Eyeball Model

In recent years, the communication paradigm has shifted from the traditional client-server model towards a more content-eyeball-transit model [73, 80], where the ISPs are divided into three different types: content, eyeball, and transit [49]. Eyeball ISPs specialize in delivering content to the last-mile consumers, that is, to hundreds of thousands of users. Examples of such ISPs are Comcast, BBC, etc. Content ISPs provide hosting and access for end users and commercial content providers such as Google, Youtube, or Netflix. Typical examples of content ISPs are CDNs like Akamai. A third class of ISP, called the transit ISPs are the ones that provide transit services for other ISPs and naturally form a full mesh topology to allow universal Internet access. Examples of such ISPs are usually Tier-1 ISPs such as Level-3, Qwest, etc. In the content-eyeball-transit model, the content or data takes precedence over the location of the server. This model of communication demonstrates a content-oriented paradigm, and got its popularity due to the increased computing power in the client machines and the interest in content rather than its source.

In the contemporary world, where the consumers search and retrieve content using search engines, VoD sites, etc., the location of the content provider is of hardly any importance from the consumer's point of view. Thus, the content centric paradigm has become the communication paradigm of choice in the current internet. In this paradigm, content searching facilities (e.g. Google, Bing, Yahoo) take the central role in locating the content in the Internet. The end users interact with the search facilities and consume the content returned by them, mostly irrespective of where the data is originating from.

2.3 Data Redundancy in the Network

As discussed in Section 2.2.1, the core network carries the aggregated traffic of all the access networks connected to it. This creates a large amount of redundancy in the network due to duplicated request for similar contents from multiple consumers. In addition to this, data objects (such as music, video, etc.) can also have internal redundancy [11, 45], which is only discoverable by looking inside the data packets.

2.3.1 Traffic Aggregation Points

A complete end-to-end data flow from a server to a consumer encounters several traffic aggregation points in the network. At those points, the traffic from multiple downstream networks are aggregated in order to be carried over a higher bandwidth network. For example, traffic coming from the consumers is aggregated at the access network entry point, after which the traffic from the access networks is aggregated at the core network entry point, and so on (see Figure 2.2). Each such aggregation generates data duplicacy in the network. This duplicacy creates unnecessary data load on the network which, consumes expensive bandwidth and generates latency. There have been many solutions to avoid such duplicacy, for example, edge caches and content delivery networks (CDN). However, firstly, these solutions themselves generate a different type of duplicacy where identical pieces of data are stored in multiple places of the network, and secondly, they cannot detect the type of data duplicacy where a part of a content object is duplicated in another content object. Both of these problems are addressed in this dissertation and in the publications presented. The caching duplicacy can be alleviated by carefully determining how to cache the objects and where, and then modify the routing accordingly, while the redundancy detection can be improved by carrying out an inspection at a level even more fine-grained than the object level.

2.3.2 Object-level Redundancy

Popular duplicacy removal tools used in the Internet (e.g. web caches [122], content delivery networks [97, 55], P2P caches [100]) deal with object level and application level redundancy. These techniques can detect application layer object requests from clients and serve them from local storage. In

this context, the application layer object refers to objects such as music files, video files, compressed files, text files, html files, etc. The detection of duplicates is not done based on the content matching of the objects, rather it is done based on the name of the objects. So, object naming is very important in detecting duplicates in object-level redundancy detection. If two identical objects have different names, they are not detected as duplicates.

In a typical access network, object level redundancy elimination can reduce a substantial amount of duplicate data because people in the same organization or household tend to consume a similar type of data. For example, the popular webpages (e.g. news portals, web based email, search engines) often have images and other html content on the page, which is retrieved whenever the page is accessed. With object level redundancy elimination, these duplicate objects are not fetched multiple times from the server. Rather, they are delivered from the local cache. The efficacy of such solutions, as well as their improvements, have been studied in numerous projects [144, 143, 48, 62, 145]. These types of techniques store popular objects in storage devices located at the edge of the network, so that requests for popular objects can be satisfied before reaching the core network.

2.3.3 Chunk-level Redundancy

The next generation of object level redundancy detection is chunk level redundancy, where instead of data objects, a mechanism to detect data chunk duplicacy is used. By data chunk, we refer to fixed-size data blocks inside a content. The role of the chunking mechanism is to determine a chunk size and then divide the content into fixed-size pieces. A popular example of chunk redundancy is Bit torrent [31], where the content to be shared is divided into fixed-size chunks and distributed among sharing peers. The advantage of chunking becomes apparent when the peers are able to download or share parts of the complete content. By utilizing this property, Bit torrent users are able to share content in a very distributed way, where a consumer can download a piece of content from multiple parties who are sharing different pieces of it.

However, fixed-size chunk redundancy detection has the shortcoming that an insertion or deletion in the middle of the content causes all the chunk boundaries after the modification to be shifted. Thus, for example, although practically 99% of the content might still be the same, due

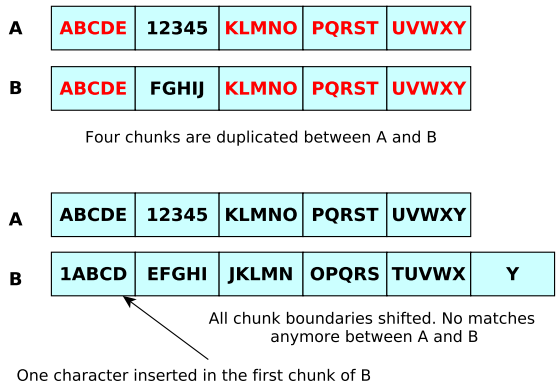

Figure 2.4. A simple insertion causes all the chunk boundaries to shift, resulting in a total mismatch

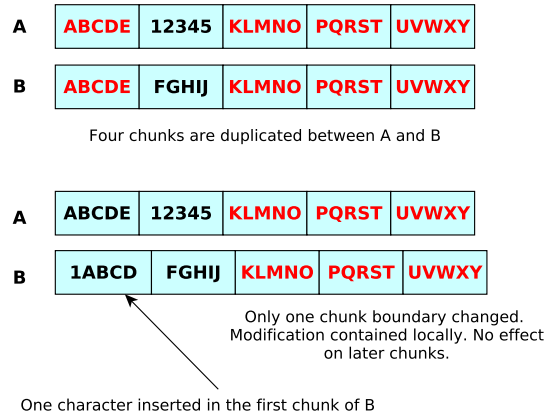

Figure 2.5. A simple insertion causes only one boundary to be changed, resulting in similarity preservation

to some insertions or deletions at the beginning of the content, all the chunk boundaries are shifted with the result that a large proportion of the chunks might prove to be (ultimately) unusable. This scenario is depicted in Figure 2.4, where due to only one character insertion at the beginning of content B, 4 out of 5 matches with A are completely destroyed. The primary takeaway from this small example is that the fixed-size chunking mechanism cannot tolerate any insertions/deletions in the content.

On the other hand, unlike object level redundancy, chunking does not depend on human readable names; rather each chunk is identified by a fingerprint which depends on the content of the chunk. Thus, identical chunks always have the same fingerprint, allowing identification of duplicate chunks reliably.

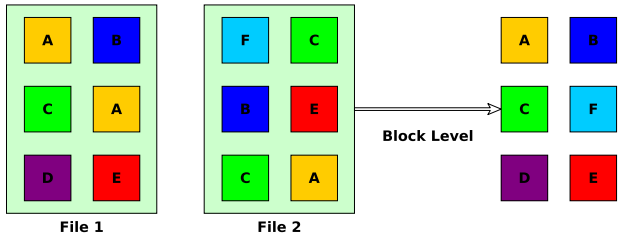

Figure 2.6. Similarities between two files only detectable at data level (Best viewed in the electronic version or in a color print)

2.3.4 Data-level Redundancy

Data level redundancy refers to the packet level, protocol independent duplicacy in the network streams that does not depend on any fixed size. For example, if two different traffic streams between two different client-server pairs have arrays of duplicate byte streams, then those duplicates are called data level redundancy. Detecting such a level of redundancy is more involved than object level redundancy or chunk level redundancy because it cannot make any assumptions regarding the level of duplicacy based on the name of the object or the format of the object. Data level redundancy detection among network streams borrows from different file compression [8, 113] and network file system optimization mechanisms [94]. Primary difference between chunk level fixed-size redundancy and data level redundancy is in the determination of the chunk size and boundary. While in chunk level redundancy the chunk sizes are usually fixed, and the chunk boundary is determined solely based on this fixed size, in data level redundancy the chunk size and the boundary are variable. The boundary determination depends on the content itself rather than some fixed size. As a result, any insertion or deletion within the content has only a local effect, and does not change all the subsequent chunks. If we take the previous example in Section 2.3.3, the unfortunate situation of one insertion causing a total mismatch should not happen in variable size chunking. This is shown in Figure 2.5.

Data level redundancy has been discovered and utilized in many of the offline data synchronization techniques such as the well-known Linux utility rsync [125], and some levels of RAID. When a system is able to detect and utilize data level redundancy, it can inspect the data inside the content objects (e.g. a video file) and discover similarities at a block of byte level. As shown in Figure 2.6, this allows the system to efficiently

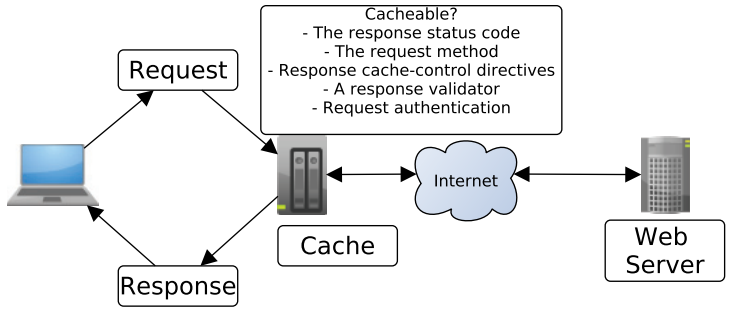

Figure 2.7. Web Caching Process

reduce the amount of data needed to be stored or backed up, while still retaining the capability to reproduce the original contents losslessly using metadata.

2.4 Redundancy Elimination Techniques

A natural afterthought related to the detection of object level and data level redundancy concerns its elimination. Over the past few decades, many research works have concentrated on eliminating object level redundancy as that was the most prominent and obvious duplicacy to eliminate. In recent years, however, as object level redundancy elimination techniques have become more mature, the research world have gone one step further and started to look into data level redundancy elimination techniques. In addition to these techniques, the distribution of data also helps eliminate duplicacy. When distributing the same object to multiple subscribers, the servers can use clever techniques [15, 65, 77, 23] so that the content is duplicated only at the very last moment to avoid unnecessary traffic in the core network.

2.4.1 Web Caches

One of the most popular object level redundancy elimination techniques is a web cache [142, 62, 140]. The widespread use of the World Wide Web (WWW) has caused the web to become the major medium of content-spreading to the users. A web page typically contains an assortment of presentation objects including text, music, video, pictures, etc., to introduce a topic to the consumer. Popular examples of web pages are news portals, web based email services, search engines, Wikipedia, etc. Typ-

ically, the content of the web pages are static (The number of dynamic web pages are increasing though), and thus can be stored in a cache to be served to the consumers locally instead of reaching out to the servers.

A web cache is a mechanism for temporarily storing these web contents near to the end user to reduce bandwidth usage, server load, and delay. As shown in Figure 2.7, the web cache intelligently stores the content going downstream through it and remembers the content's name. The name of an object in the HTTP protocol is called the Unified Resource Identifier or URI of the object. This URI is usually the name of the file prepended to the location of the file in the World Wide Web. The location is indicated by the domain name (e.g. www.example.com), followed by the directory structure in the server. Thus, the full URI of an object called test.txt can be www.example.com/some/directory/test.txt. The storing of the element depends on various properties of the object, such as, whether the object is a static one or a dynamic one, whether the object is cacheable according to the content originator, etc. Usually, these properties of the objects are specified in the HTML code of the web page. The object is stored in the cache by parsing and extracting from the response of the content server. If another client at a later time makes a request for the same content, the cache offers that content directly from the storage, rather than forwarding the request upstream. This mechanism makes it possible to cut short a large amount of static content traffic from the core network.

2.4.2 Data-level Redundancy Elimination

Data level redundancy detection techniques have responsibility for detecting protocol independent redundancy in the data stream going through the network. The techniques are employed to try to inspect raw data packets going through the network and detect duplicate data byte arrays, irrespective of the protocol of the stream or the application layer of the data. These duplicate byte arrays are then removed from the core network to reduce the traffic. Starting with Spring et. al's [121] pioneering work, several other research works [10, 126, 11, 120, 60] have validated the existence of such redundancy and the efficacy of the elimination techniques.

Eliminating redundancy by inspecting byte streams is not a recent idea. This idea has been in existence in the fields of compression [8] and networked file systems [94, 22, 25] for more than a decade. Earlier, this technique was used offline to understand the differences between the two

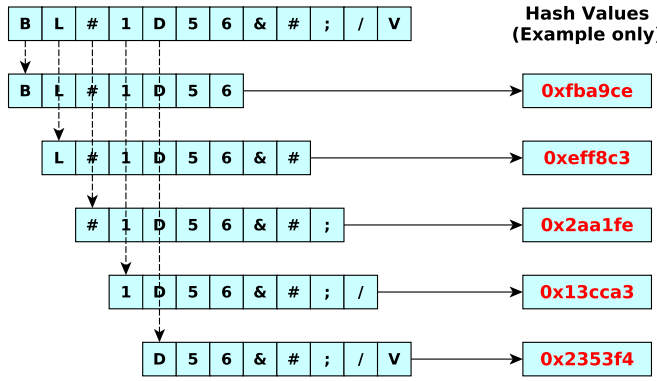

Figure 2.8. Sliding window based rolling hash computation

storage mediums and then to perform synchronization without transferring the whole content. In addition to that, the same techniques were used to perform compression of files by detecting similar regions of a file and by replacing them with a marker so that the file can be reconstructed later.

The primary technique used in these applications is called differential compression (DC). Originally differential compression techniques were used in remote file synchronization to be able to transfer just the difference between two files rather than transferring the whole. The procedure used in differential compression is different from the procedures used in other patching oriented differencing mechanisms such as Binary Delta Compression [19], which require the knowledge of file versioning and find the difference based on that knowledge. On the other hand, DC mechanisms are file agnostic and calculate the difference on the fly based on byte streams.

DC works by dividing a file's data into variable length chunks. The chunking process is a function of the data rather than the size. Thus, any intermediate additions or deletions of data do not affect the surrounding matching chunks. The division is done by computing the local maxima of a fingerprinting function, which is computed at every byte of the stream. The fingerprinting function used in this case is a function that can be computed incrementally. In many cases, the Rabin Fingerprinting Algorithm [113] is used to compute the fingerprint. This fingerprinting operation is a method of implementing fingerprints using polynomials over a finite field, where a finite field is a set containing a finite number of elements. Given an n-bit message, we can view it as a polynomial of degree

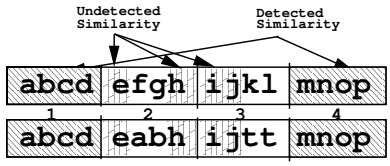

Figure 2.9. Matching regions that cannot be detected by chunk-level RE, but can be detected by maximal-match RE with many memory accesses. © [2011] IEEE

$(n-1)$ over a given finite field denoted by $f(x)$. Then a random irreducible polynomial $p(x)$ of degree k over the same finite field is picked. We define the fingerprint of the message to be the remainder after division of $f(x)$ by $p(x)$ over the same finite field. The application of this fingerprinting for this work can be visualized by thinking of a sliding window passing over the byte stream, while the function computing the fingerprint is applied over the bytes inside the window, as shown in Figure 2.8. If we compute the function f over a byte range $d_i...d_j$, it should be possible to compute $d_{i+1}...d_{j+1}$ incrementally by adding the byte d_{j+1} and subtracting the byte d_i. Although any other hash function could be used to generate the fingerprint over a block of data, Rabin fingerprinting is usually chosen as it is possible to reuse the previous results while computing an incremental rolling hash. The DC algorithm slides the hashing window continuously by adding bytes to the end and subtracting from the front, and generating the hash value for each combination. Whenever, it finds a local maxima of the hash value, the byte position is chosen as the chunk boundary.

After the chunk generation is complete, a strong hash value is calculated for each chunk, which is then used to compare the chunks of two arbitrarily different byte streams. The length of the hash value is set so that the chance of collision is minimized.

DC algorithms are in many ways similar to the older rsync [129] protocol, but include many improvements to further eliminate duplicate byte ranges from two different streams without having to be aware of application layer details, such as file names etc.

The matching of incoming traffic with the chunk store is typically done at chunk level. However, matching regions can extend beyond the chunk boundary. To detect this similarity, two different approaches have been taken thus far: 1. Chunk level RE and 2. Maximal Match RE. In Chunk level RE, the comparison is done between representative fingerprints of whole chunks, which means that partial chunk matches are not detected (similar regions in chunk pairs 3 and 4 are not detected in Figure 2.9).

This problem renders the case for bigger chunks less useful, which, however, can be a better choice with respect to the processing load of the network nodes performing the chunking process. On the other hand, Maximal Match RE performs rigorous byte-by-byte matching to extend a matching chunk to its maximum. This allows maximum matching at the expense of more memory accesses.

When the chunking process is decided upon, all the incoming traffic via a RE-capable router is divided into chunks and the representative fingerprints are stored in a database. At a later time, if any of the incoming chunks match any of the fingerprints in the database, that chunk is removed and a fingerprint is inserted as a header in the data stream. This deletion is done because the presence of a chunk fingerprint in the database proves that the same data passed through this router previously and should also be found downstream in other routers as well. Thus, if the chunk is now removed and the fingerprint inserted, the whole data stream can be reconstructed downstream by other routers having the exact same chunk in their data store. Consequently, the network between the two routers performing the elimination and the reconstruction would have to carry much less data than they normally do.

This redundancy elimination technique can address the redundancy in the network that the object based techniques cannot. With the current trend in the Internet, media files of similar content are being shared with different names and addresses. To discover the internal similarities between these contents, the data level redundancy elimination techniques are the only option.

2.4.3 Content Distribution Techniques

In the current model of Internet, content consumption is the major driver of data growth. Different video sharing portals such as Youtube, or music sharing portals such as iTunes deliver millions of copies of the same content to different users across the world. To efficiently handle this delivery process, clever content delivery mechanisms needs to be in place so that the duplicacy of data on the wire is reduced as much as possible. The most common techniques used for this purpose are multicast [42], and content delivery networks (CDN) e.g. Akamai [97], etc. However, techniques such as IP multicast require support from the infrastructure (i.e. the intermediate routers need to support IP multicast and have to also support the creation and deletion of multicast trees, etc.). Due to this, the use of mul-

ticast is still limited. On the other hand, CDNs are more of an application layer protocol and thus do not depend much on the infrastructure level dependencies.

IP Multicast

IP, as a protocol, supports three different manners of communication. IP unicast sends separate datagrams to each of the recipients, IP broadcast sends each datagram to all the recipients on a single subnet, and IP multicast sends each datagram to only the interested parties. To allow multicasting to work, the hosts must be configured to send and receive multicast data, and the routers must support Internet Group Membership Protocol (IGMP), multicast forwarding and multicast routing protocols. A set of IP addresses are reserved as multicast IP addresses.

When sending a multicast message, the source sends the multicast datagrams to one of the multicast IP addresses. This is known as the group address. Any host interested in the multicast datagrams needs to contact the local router and become a part of the multicast group. It then starts to receive the subsequent multicast datagrams sent to that particular group. Routers use a multicast routing protocol to determine which subnets contain at least one group member, and to forward multicast datagrams only to those subnets that have group members or a router that has downstream group members.

CDN

CDNs work by maintaining big server farms in different geographical locations and by duplicating content in them to serve content locally. CDN services are built by exploiting how the world wide web works. When a web browser makes a request for a resource, the first step is to perform a DNS lookup. A DNS lookup is like a translation service which translates from the domain name to an IP address. With this IP address, the browser then can contact the content server directly. It is understandable that the farther content server is, the slower it will be to fetch content. Therefore, content providers set up servers in strategically decided locations with copies of the served contents. These servers are called edge servers, and combined these are called a CDN.

When a browser sends out a request to be handled by a CDN, the DNS resolution process becomes a little more involved than a simple IP translation. The server handling the DNS request looks at the incoming request to determine the best set of edge servers to serve the request. At its sim-

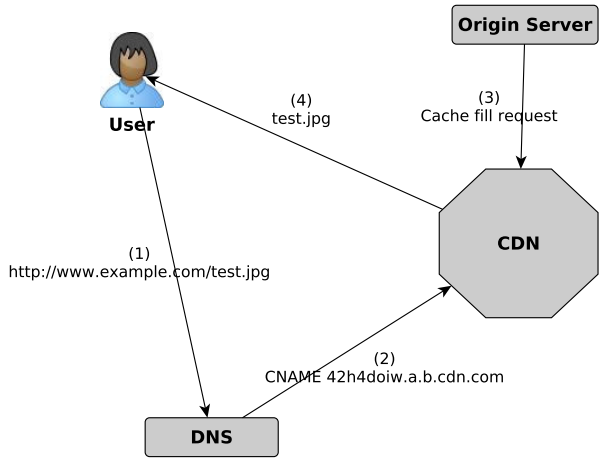

Figure 2.10. How a CDN works

plest, the DNS resolver performs a lookup and returns the address of the edge server closest to the requestor. Thus, based on the location of the requester, the IP address of the server can be different. Please keep in mind that the content providers can have other parameters for selecting the best edge servers, such as the cost of the server, speed of the link, etc.

After the selection of the edge server is done, the next step works in the same way as web caches. Upon receiving the content request, the edge server checks its cache to see if the content is present. If the content is present and has not expired, then it is served from the edge server. Otherwise, the edge server forwards the request to the origin server of the content, and retrieves the content from there. The flow of the operation is shown in Figure 2.10. Note that this flow is typical of pull CDNs. There are also push CDNs where the content providers push their whole content beforehand so that all the content is already available for the consumers without requiring the CDN to pull the content from the origin server.

In most cases, the CDN service is provided by 3rd party administrators while the content providers purchase the service to host their content in the farms. In addition to the CDN services, it is possible to develop customized application-specific content distribution techniques to reduce server load and to minimize client costs.

2.4.4 Energy Efficiency

Clever content distribution techniques can also minimize the consumed energy at the client side. As more and more clients are now mobile, saving

energy during content download is more important than ever. This can be achieved by designing the distribution techniques in such a way that the more demanding network interfaces (such as the 3G or 3G interfaces) are used less, while the power-frugal interfaces are used to download most of the content.

Different network access technologies have different mechanisms for saving power while transmitting or receiving data. IEEE 802.11 (commonly known as WiFi) interfaces come with a default power saving mechanism called power saving mode (PSM) [59]. With this mechanism, the interface periodically wakes up to check whether it has any data to send or receive, otherwise it stays in sleeping mode. On the other hand, cellular network interfaces such as 3G or LTE use power management mechanisms through the corresponding Radio Resource Control protocol (RRC). RRC has different states that the interfaces can be in. Each state corresponds to a different power saving stage of the cellular interface. For example, 3G RRC has four states [1] while LTE RRC has only two states [2].

Content distribution mechanisms that can efficiently manipulate the power saving states of the interfaces consume less power than others. Most of these techniques are used for multimedia streaming techniques as those data transfers are typically active for longer periods of time compared to simple data downloads. For WiFi, such methods include naive methods that just turn off the interface when the buffer is full [21], self tuning buffer management [12], fuzzy adaptive approaches [17], traffic prediction approaches [28], and many other efforts that propose modifying the default behavior of PSM [13, 95, 72, 112]. On the other hand, for cellular interfaces (e.g. 3G, 4G interfaces), the control of RRC is in the hands of the operators. Thus, at the device level, it is difficult to manipulate the interface idle times for better energy efficiency. Studies has suggested aggressive timer settings or changing the inactivity timers dynamically based on traffic patterns [76, 111]. Based on the inter-arrival time of the packets, modified timer settings were also proposed [47, 132].

Some recent research works have proposed methods of combining the cellular and ad-hoc connectivity of a hand-held device [136, 57, 79, 124, 75, 150]. By understanding the characteristics of the power saving mechanisms of the network interfaces and using them correctly, the battery power of the device can be significantly saved.

2.5 Information Centric Networking

In an effort to redesign the Internet to better reflect its usage, researchers have thought of a clean-slate architecture where instead of using a client-server paradigm, a more data centric approach is taken. In this approach, data is at the center of user interest, and instead of mentioning the server where the data is available, users just have to mention the data they are interested in. The network is then responsible for routing the request to the source of data and then routing the response back to the requestor. The machinery of the network need to be different from today's network so that the routing of data depends of the data itself rather than the server IP address.

2.5.1 Motivation

While the Internet was designed primarily as a medium of communication between a known server and a client, the current usage of the Internet has changed towards a more content oriented paradigm, where the content is consumed irrespective of where it originates from [73]. Information Centric Networking (ICN) [146] is one of the future Internet paradigms which takes this change into consideration [71, 63]. ICN removes the need for application layer translation services that supply the server address upon being given the content name, and it allows content providers to serve content from any location on the network.

Storage prices have decreased faster than bandwidth cost [40, 16], making it easier to cache all the content within a network than to fetch it from the source. In addition, recent surveys [46, 5, 11] have shown that a large portion of the network traffic is redundant and can be cached. ICN makes it easier to cache content irrespective of its application layer characteristics.

ICN also makes it possible to ensure security of the data chunks by means of a digital signature, and optionally encryption. Each of the ICN content objects can be digitally signed and can contain information about the publisher of the content. The content can optionally be encrypted as well. Thus, the consumers can detect whether the content originates from a publisher they trust, and also can ensure that the content has not been modified en route. Unlike today's network where the data path needs to be secured so that unsecured content can pass through it, ICN can ensure security independent of the data path.

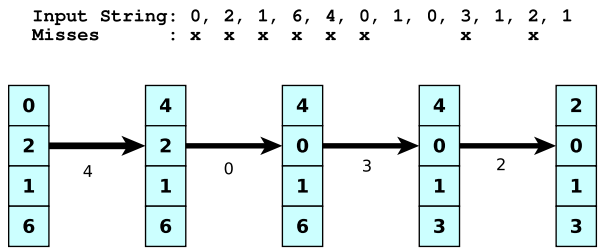

Figure 2.11. Cache replacement with LRU

2.5.2 ICN Projects

Multiple architectural proposals for Information Centric Networks have been put forward, of which CCNx [63], NetInf [35, 36], Dona [71], PSIRP [128], PURSUIT [51] are among the more notable. Many of these are results of various broader projects such as 4WARD [3, 64, 6], COMET [32, 27, 52], NDN [96], PSIRP [109], PURSUIT [110], and SAIL [119, 36].

Each of the proposals has its unique network architecture, caching, and data dissemination techniques. For example, CCNx [63] uses interest messages and self-identifying names to route data requests. It uses a cache everything everywhere (CEE) approach, and relies on cache eviction methods such as Least Recently Used (LRU). LRU, as the name indicates, performs cache eviction based on the timestamp a particular cache slot was accessed on. The staler the cache slot is, the more chance it has of getting evicted from the slot in the event of new content arriving. A simple illustration of LRU is shown in Figure 2.11, where we have a cache space of 4 slots, and the replacement policy is LRU. According to the incoming string, the contents of the cache slots are being evicted while the oldest cache occupants are being evicted out and new ones inserted in.

On the subject of other ICN architectures, NetInf [107], an architecture developed as a result of multiple projects such as 4WARD [3, 64, 6] and SAIL [119], allows the use of several caching mechanisms such as cooperative caching with the help of a centralized system, or a more proactive caching where caches subscribe to the contents themselves. In PSIRP [109, 74, 127] caching is limited to the transmission path, and registered within the name resolution system. PURSUIT [51], which is a continuation of PSIRP, does not cache along the path to allow asymmetric routing. Nevertheless, the caching is still concentrated in predefined areas. None of these architectures propose any method for the efficient

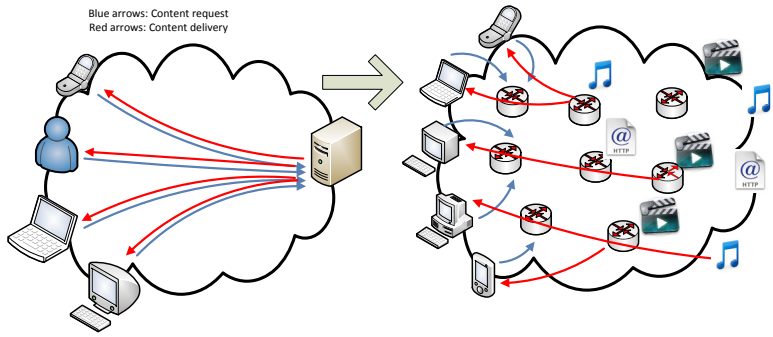

Figure 2.12. From an IP network to ICN network (Best viewed in the electronic version or on a color print)

use of the available cache space, resulting in high degree of dependency on the servers for data delivery. Interested readers are referred to the project websites for more detailed information.

2.5.3 Data Distribution Principle

The common principle among all the different ICN proposals is the emphasis on changing the data distribution technique that is used in the current Internet. In the current Internet architecture, data is usually located in a server that needs to be known to the client in order to retrieve the data. However, with potentially unlimited amount of data available, it is not possible for a client to know where to find the required content. To alleviate this problem, application layer search engines are heavily relied on to support human clients by providing the address of the server holding the content upon being given the content description. In ICN architecture, the network takes this role and the need for knowing the server serving the data no longer exists. When a client searches for the content name, the network routes the request according to the content name instead of the server address. Moreover, the routing information in the network elements is based on the content's name. When the request is routed through to the nearest server, the response is routed back and cached in different caching points along the return path. These caches then can work as a

data source for further requests of the same data. A very simple depiction of an ICN network and its relation with an IP network is shown in Figure 2.12.

Each of the application layer contents (video, music, text, etc.) are usually divided into smaller blocks of data which are individually addressed. The division is done in such a way that changes in one block do not affect the others. Thus, partial changes in the application layer content do not affect the other similar parts, making it possible to detect block level similarity. Request for application layer contents are typically divided into multiple requests for the constituent data blocks, and those are then independently fetched from the network. Enhancements to ICN architectures have also been proposed where the content blocks are not only identified by their content but also by their type (reliable/real-time/etc.) [131]. The content blocks can take different routes, can arrive in a different order than originally intended, and can be served by different sources. As long as the fingerprints of the content blocks match, they are stitched together at the requestor, and delivered to the consumer. The blocks are usually cached all along the path, or can be cached only in specific locations based on the caching architecture of the proposal. A survey of different caching schemes used in ICN can be found in [148].

The content based data blocks and fingerprint based naming of the blocks make it difficult to efficiently name the ICN data blocks [53], especially on a larger scale, such as in inter-domain routing. Many proposals such as Dona [71] uses hierarchical resolution, while others use DHT based name resolution [67, 34], or rendezvous based hierarchy [114]. Moreover, such naming schemes also mean that the routers need to keep a large amount of state data to ensure proper routing capability without flooding the network. This means expensive memory consumption and expensive processing power involved in searching large databases [130, 137]. In addition, large data dependent states also increase security threats on the data path [139, 138].

2.5.4 Redundancy in ICN

The data centric approach and aggressive caching schemes of most of the ICN architectures can lead to a considerable amount of data redundancy in the network. This redundancy is a result of caching the same content in many places of the network, essentially in nearby positions that could easily be replaced with only a single copy of the content without compro-

mising content delivery efficiency. However, most of the ICN architecture proposals today do not take this optimization into consideration, and consider storage space to be cheap and affordable enough to waste on redundant copies of data. Contrary to common belief, storing redundant copies might affect the overall cache efficiency of the network. Keeping the duplicate copies means that some other unique copies of content might need to be thrown out causing future requests for the disposed data to be routed to the server.

The redundancy in ICN is easier to detect than in the traditional Internet because the content in ICN is identified by definitive fingerprints and thus duplicates can easily be detected. The elimination techniques of such redundancy are discussed in [103, 141]. Because it is possible to deliver a complete object that is utilizing chunks from different sources, an ICN network can chose to keep only a few copies of a particular chunk in well-known places and deliver from there instead of blindly caching it on every path towards the clients.

While most of the ICN proposals took the mainstream approach of caching redundant copies of data throughout the network, only a few [26, 108] has explored the area of probabilistic caching where data is not cached everywhere while traveling from one point to another. Rather, the caching decision is selective and based on some probabilistic approach depending on different network parameters. However, these approaches only deal with on-path caching and does not take the advantage of off-path data availability.

Cooperative caching is another front which was researched quite extensively during the early phases of web caching, and was found very costly in terms of signaling and state [144]. Proper cooperation among the routers require heavy exchange of cache state which can cripple the network with signaling load.

2.5.5 Open Issues in ICN

In spite of many useful features, the primary issue that ICN has to face is that of deployability. Is ICN going to be deployed in real world? In responding to that question the first thing to consider is economic. Who should pay the money for the deployment? It can be the end user, for whom the advantage of ICN are nothing that cannot be done reasonably well by a combination of search engines and browsers. It can also be the content service provider, who would then need to know whether ICN can

outperform and provide better value for money than CDN. If the network provider is to bear the costs then the key issue is whether it will provide better caching opportunities and reduced transit costs. But when all is said and done, there still remains every possibility that the higher tier ISPs will resist the deployment. For all the above reasons, the economic benefits and model of ICN needs to be researched and thought through to make it a success in the practical domain.

Another factor to take into account is that the resource management of ICN still lacks maturity. Which parties will perform the heavy loading of caching the content and serving others at their own expense? Without a clear benefit model, this question will remain unsolved. This dissertation looks at some of the resource and cache management cases where the incentive of investing in resource follows a similar economic model as CDNs.

2.6 Summary

In this chapter, we have discussed the Internet and its data distribution techniques. In addition, the different types of network within the Internet were also discussed along with the changing nature of the data communication paradigm over time. We have also discussed the redundancy of data in the Internet and the proposals to eliminate it. A brief review of the effect of such elimination techniques on the energy efficiency of the handheld devices was also discussed. The background review was finished with a quick peek at the very latest clean-slate Internet paradigm called the Information Centric Networking and its own version of data redundancy.

3. On Reducing Redundancy in the Internet

Data redundancy at various levels of the Internet causes valuable resources in the network to be wasted and delays being introduced for less popular data objects. As evidenced by the previous chapters, data redundancy exists in different forms in the network, and various measures can be taken to reduce this redundancy. In addition, data redundancy also affects the energy efficiency of the elements in the network. Elements in the network need to process, forward, and store redundant data, and these operations require energy. Thus, by reducing redundancy, it is possible to reduce the required energy in the network. This chapter discusses the concepts of data redundancy and the contributions of this thesis to the field.

3.1 Data Redundancy in the Internet

The data distribution paradigm of the Internet was originally a client-server based one, that is, the clients request data from a server and the server serves the data to them. The paradigm did not, however, take into account the scenario that most of the data paths might be duplicated if people from nearby regions starts to request the same data (which is often the case for popular contents). Consequently, the core of the network tends to carry large amounts of duplicated data. Later, researchers and networking experts started to understand the problem and addressed it by various means such as web caches, content data networks, and so on.

The web caches or caches in general in the Internet work as a temporary repository of popular content on the path between the clients and the server. When the data is served to the client, it is stored in the intermediary caches and reused later when requests for the same object go through the same path to the server. The data objects residing in the intermediate

caches are validated after a predetermined time interval to prevent data staleness. This concept of web caching was, and still is, hugely successful. However, one constraint assumed by the web caches is that the cache matching is done on the application layer object level. By application layer object, we mean the objects that are usually transferred over the HTTP protocol (the predominant application layer protocol in the Internet) such as web pages, music, video, compressed files, and other similar objects. This constraint results in a very high level of object matching schemes, where a simple change of object name causes a mismatch to occur between two intrinsically same objects. For example, if the same video is named sample1.avi in one server and sample2.avi in another, they will not be considered as the same video by the web caches, and both will be stored separately.

3.2 Content Independent Redundancy Elimination

Redundancy elimination (RE) techniques that utilize DC technology employ the above mentioned chunking mechanism to efficiently eliminate the maximum amount of redundancy. One of the most important parameters in RE algorithms is the average chunk size. The average chunk size depends on the fingerprinting function that is used over the sliding window. Often, the fingerprinting function used in RE algorithms is a Rabin fingerprinting algorithm [113]. By tuning parameters in the Rabin algorithm, it is possible to change the average chunk size produced by the chunking algorithm. In the following subsections we will discuss how this ability to fingerprint variable lengths of content chunks can help us determine similarity between two completely unrelated traffic flows.

3.2.1 Content Awareness Vs Unawareness

Redundancy elimination techniques in the Internet consists of well established and widely used processes such as web caching. In this technique, application layer objects (such as music, video, files, etc.) are detected and stored in middle boxes for future delivery to the clients without reaching out to the server when the object name matches. Thus, this method of caching is content-aware, and does not work if the type of the content changes even though the internal data is completely the same. For example, if there are two servers X and Y serving the same content a.zip and

using the URIs X/a.zip and Y/a.zip, then the caching nodes are able to detect this and can serve the clients requesting Y/a.zip from the cache if some other client has requested X/a.zip in the past. However, if for some reason, server Y renames the file a.zip to a1.zip, the caching nodes cannot detect the similarity anymore and will forward all the requests for a1.zip to Y.

On the other hand, content unaware solutions do not depend on the apparent name or type of the content, but rather they treat each content as a raw traffic stream, and analyze the data inside it to understand the similarity. Thus, the process is unaware of the type of the contents, and all that matters is the internal data-level similarity between them. Consequently, even if there is partial similarity between two contents, it can be detected. After detecting this partial similarity, it is possible to remove this similar region or to transmit it only once.

3.2.2 Inter-Content Redundancy Detection and Elimination

As discussed in sections 2.4.2 and 3.2.1, it is possible to discover partial similarities between two streams of data traffic. The operation is explained in brief below:

In a content router, each incoming data stream is inspected and chunked (Section 2.4.2) into variable sizes of data pieces. After the chunking process, a fingerprint for each chunk is created using a strong hash function, which is then stored in an internal database for future use. As the boundaries of chunks are dependent on the property of the content and not on any fixed size, so any insertion or deletion in the content just changes the chunk boundary locally and does not have any global effect on chunk boundaries. Subsequently, for each data chunk, the fingerprint is matched against the existing fingerprints in the database of that router. In case of a match, that particular data chunk is removed from the data stream and a small header containing the fingerprint corresponding to that chunk is inserted. Later, when the data stream with the header reaches a downstream router that also has the same data chunk stored in the database, the header is replaced again with the data so that the original content is reconstructed.

There are two different aspects of this process that requires further thought. The first one is that the upstream routers need to have the knowledge of the content store of the downstream routers so that they know when to reconstruct the content back to its original form instead of

forwarding the minimized traffic. The second one is that the smaller the chunk size is, the better the probability of finding a similar region between two traffic streams. However, this also means that the overhead for headers is maximized because for every small chunk we have to insert a fixed size header, and the smaller the chunk is, the bigger the overhead is. Most of the previous work [120, 10, 25] just used a small chunk size to report high similarity detection. However, they failed to mention the increasing number of headers to be transmitted, and the added complexity in the operation. While the importance of choosing the right chunk size is well understood and analyzed in [126] and [54], no work so far has proposed any solution to combine the advantages of using small chunk sizes (better similarity detection) with that of bigger ones (better compression rate). Most of the current solutions either use a fixed size chunk—inserting many shim headers in the outgoing packet, or use a byte-by-byte comparison to maximize the matching region—generating numerous memory accesses.

In Publication I, we have proposed a novel algorithm named CombiHeader to minimize the number of headers used in the RE algorithms while maximizing the similarity between two data streams. For traffic flows with around 70% similarity, CombiHeader uses fewer than 20% shim headers compared to general RE solutions. CombiHeader also uses very few memory accesses to enlarge the chunks to their maximum matching size. While a byte-by-byte comparison to expand a 1024-byte chunk to a 1374-byte similarity region takes around 351 memory accesses and one shim header, CombiHeader running with 256-byte chunk size requires only nine memory accesses and one shim header in the best case. This is around 2% of the actual number of accesses.

The idea of CombiHeader is based on the presumption of spatial locality. We assume that if two consecutive chunks are highly popular, then the probability of them always appearing together is higher. Therefore, if we can keep track of adjacent popular chunks and their probability of appearing together, it is possible to merge them together and transmit only one header for more than one chunk. However, merging two headers together will cause the constituent smaller chunks not to be matched anymore.

CombiHeader employs a specialized data structure to tackle this problem. With this data structure, both the elementary and the merged chunk fingerprints are stored in such a way that it is possible to gradually pile up chunk hits to reach a maximal region and then transmit in a burst.

Algorithm 1 Pseudocode of CombiHeader algorithm

```
if current chunk is cache miss then
    if miss streak = 1 then
        Update last seen Combi and Elementary node
        Create new complex CombiNode if possible
    else
        Clear last CombiNode and transmit it
        Update last seen elementary node
    end if
    Transmit full text of current chunk
else
    if miss streak = 1 then
        Start fresh, clear last seen CombiNode
        Do not transmit anything
        Buffer current chunk as last seen elementary
        miss streak← 0
    else
        Update last seen elementary node
        Try to create new CombiNode with (Last CombiNode, current Node)
        if Try failed then
            Transmit and clear last CombiNode
        else
            Do not transmit anything
            Complex CombiNode is accumulating chunks to transmit
        end if
    end if

end if
```

For each adjacent chunk pair, the algorithm maintains a link, and updates the link hit count whenever those two chunks appear one after another in a stream of data. Further, based on a threshold (θ) of link hits, a new merged chunk (CombiNode) is generated from the two elementary chunks, and is attached to the link. After the creating a new CombiNode based on the hitcount of the elementary nodes, no header is inserted when a constituent elementary node is hit, but rather the algorithm waits for the next chunk and matches it up with the linked elementary fingerprint. In case of a match, it inserts only the CombiHeader (Header created from a CombiNode) that has been created earlier instead of two elementary headers. The algorithm is illustrated in Algorithm 1.

This algorithm increases the performance of redundancy elimination by reducing the number of headers it inserts in the traffic, while concurrently detecting similarities at the same level as small chunks. For details about the performance measurement setup and results, please refer to Publication I.

3.3 Eliminating Redundancy by Caching in Information Centric Networks

Redundancy elimination can also be achieved by using a well-researched and well-known technique called caching, where contents are stored in intermediate locations to allow quicker delivery of the content to the consumer as well as to reduce the content server load. By delivering in this fashion the content from an intermediate location, the load on the rest of the data path is also reduced. Moreover, it is not necessary to deliver the same content over the same path multiple times if the location of caching is well placed. Caching is an integral part of the current Internet although it is not a property of the network layer. Application layer overlays, such as web caches or CDNs create the necessary caching infrastructure in the Internet. In clean-slate internet architecture proposals (such as ICN), caching has been given even more importance and is proposed as an integral part of the network layer.

Information Centric Networks or ICN is a new paradigm of networking where instead of a client-server communication, the primary focus is to retrieve data irrespective of the origin. Accommodating the current eyeball–transit–content model [73] of the Internet requires a radical change in its architecture, and ICN [71, 63] comes into picture in

that respect. One of the integral parts of most of the ICN proposals is caching [3, 109, 119], which allows quick delivery of content without overloading the content server. Caching, as a technique, can have multiple efficiency aspects. While some proposals for caching choose to concentrate on the most popular contents of the Internet by duplicating popular contents everywhere to reduce latency, others concentrate on utilizing the available caching space efficiently to cache as much data as possible and to reduce duplicacy. Quite a few studies have looked into the first category of caching [43, 118], but the second category (storage efficiency) has been ignored by the majority of studies. Considering that ICN poses no limitation on where the data can be served from, better cache storage efficiency can result in a much better data retention capability for the network. The network can retain the data objects[1] in the in-network-cache[2] based on the interest level, and thus keep it available for the clients even in the absence of the server. For this to happen, the total cache space in the network has to be utilized efficiently to retain as much data as possible.

3.3.1 Caches in Information Centric Networks

The idea of ICN stemmed from a very early work named TRIAD [29], which introduced publish-subscribe as a core Internet paradigm (inspired by [98]). A long time after TRIAD was proposed, networking researchers started to realize the validity of the work, which resulted in a widespread interest in similar topics, producing different ICN architectures such as CCN [63], 4WARD [3, 64, 6], PSIRP [109, 74, 51, 127], SAIL [119, 36], and COMET [32, 27, 52]. One common aspect of almost all of these proposals is universal caching with minor variants. That is, instead of having a few specialized caching points across the network as an overlay, caching is implemented by all the ICN nodes.

3.3.2 Distributed Caching Responsibility

Caching reduces redundancy in the Internet in two ways. One is the traditional functionality of caching: serving content from a local cache and thus reducing duplicate content transfer in the core network. The second is by means of reducing the duplicacy of the cache contents of the Internet

[1] Please note we do not mean application layer objects here, but rather any units of data such as packet-chunks used in, for example, CCNx [63]

[2] In-network-cache refers to a caching scheme where caching is integrated in core network elements, such as routers, instead of being overlaid as HTTP caches

as a whole. While the first method was and still is the primary focus of research, the second can be thought of an extension of the first, which, without sacrificing the benefits presented by caching, attempts to optimize the storage space usage of the cache. By doing this, it makes sure that the aggregated cache space of the network is used to its full potential and the maximum amount of content is stored in the cache.

To achieve such a utilization of the cache, a distributed control mechanism is needed. This mechanism will control how the content is stored in the cache and where. Moreover, the placement must be deterministic so that searching for content is as lightweight as possible. To achieve this, Publication IV proposes a distributed algorithm where the signaling is piggybacked over the existing signaling traffic for routing, and the individual autonomous system (AS) can decide how to manage its own caches and the content it wants to cache. The goal is to achieve a controlled distribution of data objects in order to keep the number of redundantly duplicated objects controllable, while retaining the benefits of replacement policies.

The algorithm to distribute the caching responsibility in an Information Centric Network works in two separate phases. In the first phase, limited to within the AS, each AS decides its own caching policy so that the content is spread around within the AS. The proposed method dictates that the border router of the AS will implement a hashing function that will deterministically determine which caching router within the AS will cache a particular content. After this decision, the content packets are rerouted through the selected cache. This procedure distributes the cache's content within the AS and utilizes the available caching space efficiently.

While the intra-AS caching distribution allows each AS to maximize the utilization of its own cache, more cooperation is required to extend this benefit internet-wide. To achieve this, Publication IV proposes different scenarios, ranging from a scenario where the content always follows the shortest path back to the requestor to a scenario where the request and the response take a detour to accommodate the caching interest of an intermediate AS. In this inter-AS caching distribution mechanism, each AS disseminates its own caching interest in some compact format (for example as a content ID range) piggybacked in routing messages. Every AS, upon receiving these interest messages, creates a map of the data interest of the Internet. Subsequently, when a data request is generated within that AS, the routing mechanism sends the request to the AS interested in

Algorithm 2 Proposed caching algorithm

```
{//Each router knows the topology and interest range using algorithms such as pathlet}
if (req_pkt not contains AS_path) then
    Compute all possible shortest AS_path from Requestor to Server
    if (algorithm == SCENE1) then
        AS_path = Select the first shortest path
    else if (algorithm == SCENE2) then
        AS_path = Select the first shortest path containing an interested AS
    else if (algorithm == SCENE3) then
        AS_path = Select a path going through an interested AS (might not be the shortest)
    end if
end if
for (each AS in AS_path) do
    Calculate designated_router for req_pkt (e.g. using consistent hash)
    Route to designated_router from Border_router
    Check cache in designated_router for cache hit
    if (Cache Hit) then
        break
    else
        Route to Border_router of next AS
    end if
end for
for (each AS in reverse AS_path starting from cache hit) do
    Calculate designated_router for data_pkt (e.g. using consistent hash)
    Route to designated_router from Border_router
    if (Data_pkt not in cache) then
        Cache the data
    end if
    Route to Border_router of next AS
end for
```

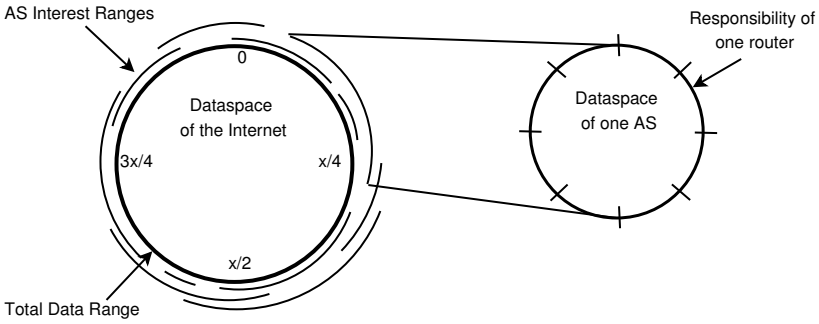

Figure 3.1. Example data interest of the ASes, and data responsibility of the routers. © [2015] Elsevier

that particular data instead of sending it to the server serving it. Different versions of this algorithm differ in how they select the best from the multiple paths towards the server. With a more conservative approach, only the paths with the shortest path lengths are considered, and the path which contains an interested AS is chosen. On the other hand, the most experimental approach does not take into account the path length at all, and sends the data request through an interested AS, even if the route is longer than the alternatives. By rerouting content and requests through interested ASes, the caching can be distributed throughout the network, and thus avoid duplicating of the content. Additionally, it also avoids the creation of hotspots. By distributing the caching responsibility, this proposal achieves a very high retention rate as the content is not duplicated and, thus, the need to evacuate a particular piece of content is reduced. The algorithm and three different scenarios (1. Select the first shortest path, 2. Select the first shortest path having an interested AS in it, and 3. Select the path having an interested AS in it) are described in Algorithm 2.

3.3.3 Cache Efficiency

The algorithm described in Section 3.3.2 should result in a well-distributed caching architecture where individual ASes are able to cache according to their interest. The distribution should result in a situation where the network is able to store a maximal amount of content in its cache without creating many duplicates, while the interest based caching will allow the ASes to tailor their own storage according to their clients' needs. On the other hand, the caching responsibility can also be based on a central

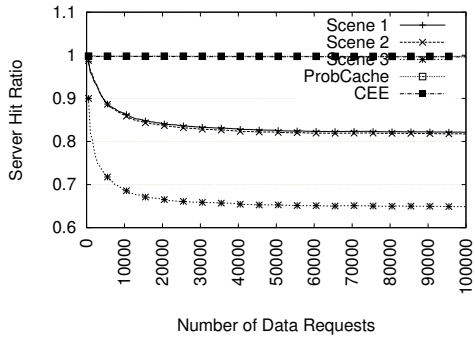

Figure 3.2. Server hit ratio. © [2015] Elsevier

authority rather than interest. The central authority based responsibility division would take into account different parameters to position the contents in optimal locations. In Publication IV we have simulated and measured the performance of the interest based caching distribution with the different scenarios as described in Algorithm 2. Additionally, in Publication V, we have explored the benefits that can be achieved using a central authority based responsibility division.

In the interest based cache responsibility allocation, as shown in Figure 3.1, each AS expresses its interest in a range of content from the total content ID range of the Internet. Thus, there can be overlaps, and duplications in the expressed interest ranges, which is quite common in the current architecture of caches, where CDNs are often used as a localized repository of popular data. In our measurements (details can be found in Publication IV), we have found that the total amount of content requests reaching the server has decreased in each of the scenarios compared to the traditional caching everything everywhere (CEE) strategy. As shown in Figure 3.2, Scenario 3, where the content requests are routed through a non-shortest path to go through an interested AS, performs the best in reducing the server load.

While reducing the load on the server is one of most important effects of the proposed algorithm, the storing of unique content pieces in the caches reducing redundancy is in line with the focus of this dissertation. Using the AS internal hash function to distribute the caching responsibility internally, and then using the cache interest messages to distribute the caching responsibility across different ASes will ensure that the same content is not duplicated heavily. Consequently, the total cache space usage of the network is maximized. The maximized use of the cache space

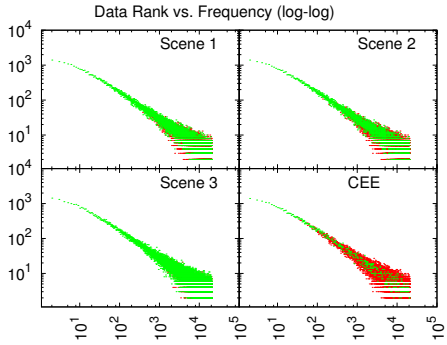

Figure 3.3. Retention of unique data objects in cache (Better viewed in color print or online). © [2015] Elsevier

results in more data objects being stored in the cache. If we consider a network where the content ID and their request popularity is plotted, the resulting graph usually follows a zipf [4] distribution, where only a few content objects are highly popular while many content objects are quite low in popularity. Due to high duplication in traditional methods of caching, only the highly popular objects are stored in the cache and the rest are served from the servers. On the other hand, the proposed solution, due to reduction of duplicity, allows many more objects to be stored in the cache. Figure 3.3 shows a typical content request zipf graph in a log-log scale for different caching schemes. For each of the subfigures, the red dots represent the content objects that were not found in the cache after a fixed period of cache warming, and the green dots represent the objects that were found in the cache. As expected, the proposed solutions were able to store more unique content objects than the traditional methods. The advantage of doing so is less server load, more cache hits, and serverless content serving for a period of time. More about this can be found in Publication IV.

3.3.4 Centralized Caching Authority

The efficiency result presented in Section 3.3.3 and in Publication IV were retrieved by simulating a network based on the assumption that the independent ASes have their own interest in a range of contents and express that interest to others. These interests can be overlapping, but it can also be the case where there is no interest in a certain range of contents. The distribution of the interest is also independent of the shortest paths between potential clients and the servers. Consequently, rerouting through

the interested AS might have the side effect of increasing the hop count of the content retrieval process, resulting in a more congested network. To understand the effect of interest distribution over the path length of content retrieval, we have simulated a network where the interest distribution is done by a central authority (hereafter called the Oracle). Moreover, we have also experimented with multiple sets of interested ASes to increase the probability that an interested AS is found nearby without increasing the path length for rerouting. Overall, there were two goals behind this exploration:

- To explore the possibilities that a centralized caching responsibility management system offers us. While we understand such a caching authority is not a trivial task, it was not our goal to solve that particular practical problem. Rather, we are exploring, if such a universal oracle is possible in the ultimate cooperation case, whether it will provide us with any benefit.

- To explore the benefits of clustering. We performed various experiments to understand whether dividing the world of caching routers into multiple geographically divided clusters and then duplicating the caching responsibilities among them would provide us with any better benefit than a single cluster worldwide.

Algorithm 3 Responsibility Distribution in a Cluster

1: { // The ASes have been partitioned into n equivalent clusters based on the mutual proximity and popularity ranking of content IDs which are based on a Zipf-Mandelbrot distribution.}
2: **for all** $content\_id$ **in** $content\_pool$ **do**
3: Get $Rank$ of $content\_id$
4: **for all** $cluster$ **in** $Clusters$ **do**
5: $index := Rank$ mod $sizeof(cluster)$
6: Assign Caching Responsibility to AS at $index$ in $cluster$
7: **end for**
8: **end for**

In this proposal, the network is first divided into k clusters, where within each cluster the caching responsibilities are distributed in such a way that the popular and unpopular contents are evenly distributed and no hotspot

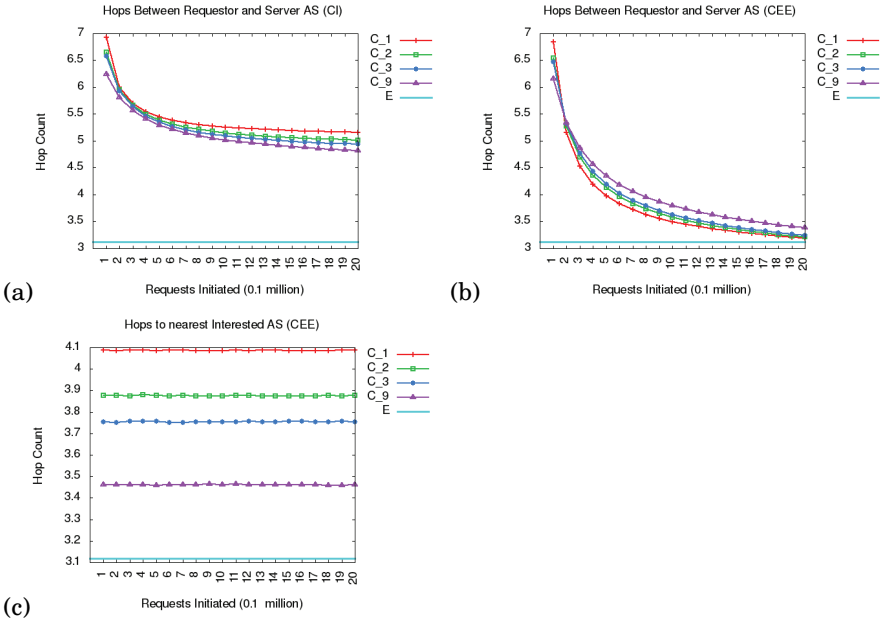

Figure 3.4. (a) Average hop count for CI, (b) Average hop count for CEE, (c) Average hops from Requester AS to the closest responsible AS. C_i represents i clusters and E represents the average expected value of hops between any two random ASes based on offline analysis. © [2015] IEEE

is created. The distribution technique can be as simple as establishing a ranking of the content popularity and then using a round robin to distribute the responsibility. Algorithm 3 describes the process.

Due to the k-clustering policy described above, we can expect a maximum of k copies of a cached element in the network. As the value of k is known beforehand, we have experimented with a procedure where the requester always finds out the nearest responsible AS from k responsible ASes, and reroutes the request through that (please see Algorithm 4). The probability of finding a responsible AS that does not increase the hop count drastically compared to the shortest path is better with multiple interested ASes than just having one interested AS in the network. With this increased probability of finding a shorter rerouting path, the increase of the hop-count introduced in Publication IV should be theoretically alleviated. However, experimental results show that the benefits might not be as good as anticipated.

We have performed our experiment on a real Internet topology, and divided the network into a different number of clusters. We have also measured the effect of caching everywhere on the path of the content (CEE), and caching only in the interested AS (CI). We have also computed for ref-

Algorithm 4 Clustered Detour Algorithm

```
1: { /* Responsibility Assignment for the whole population of data is done in every AS
   cluster as per the interest allocation algorithm. Server AS is fixed while Requester
   AS is chosen at random. Caching policy can be Caching at Responsible AS only or
   Caching Everything Everywhere */ }
2: Initialization: Create n AS clusters and assign caching responsibility
3: while iteration < MAX_NO_OF_ITERATIONS do
4:     Get random Requester_AS
5:     Get Data_ID randomly from zipf − mandelbrot distribution
6:     Find all Responsible AS for Data_ID
7:     if No Responsible AS found then
8:         Get shortest path to Server_AS from Requester_AS
9:         Route request to Server_AS
10:        if Data found Cached on path then
11:            Serve data from intermediate AS where found
12:        else
13:            Serve data from Server
14:        end if
15:    else
16:        Get shortest paths to all Responsible AS from Requester_AS
17:        Select Responsible AS at smallest hop count from Requester_AS
18:        Get shortest path to Responsible AS
19:        Route request to Responsible AS
20:        if Data found cached on path to Responsible_AS then
21:            Serve data from intermediate AS where found
22:        else
23:            if Data found at Responsible_AS then
24:                Serve data from Responsible_AS
25:            else
26:                Get shortest path to Server from Responsible_AS
27:                Route request to Server_AS
28:                if Data found Cached on path to Server_AS then
29:                    Serve data from intermediate AS where found
30:                else
31:                    Serve data from Server
32:                end if
33:                Cache data at Responsible AS
34:                Serve data from Responsible_AS to Requester
35:                Cache data at Requester
36:            end if
37:        end if
38:    end if
39:    Increment iteration
40: end while
```

erence the average distance between two random ASes within the whole network. Figure 3.4 shows how different numbers of clusters has influenced the average path length of the requests. Subfigures (a) and (b) show the required average hop count for satisfying the content request while the network warms up. The individual curves show the statistics for an increasing number of clusters in the network. In addition, there is also a curve representing the average expected value of the distance between two random ASes in the network. A steady drop in the required hopcount is observed in both the figures. An interesting pattern to observe is that the increase in the number of clusters did not have any significant effect on the hopcount. Moreover, when caching everywhere, a single cluster seems to perform better than multiple clusters. Subfigure (c) shows an analysis of the distance of the nearest interested AS for the content requests generated during the interval. As expected, the nearest interested AS becomes closer as the number of clusters increase. However, this does not affect the content fetching hopcount as expected. This counter-intuitive result shows us that clustering the network into multiple parts and duplicating the caching responsibility might not be as effective as it seems theoretically. Detailed measurement results and visualizations can be found in Publication V.

3.4 Redundancy Elimination by Cooperation

In the previous chapters, we have discussed reducing the redundancy of Internet traffic by means of inspecting the content of the packet and removing duplicate parts from the core network as well as by optimizing the caches to store redundancy free data. In this chapter, we will explore how duplicate traffic in the network can be reduced by means of cooperation among the clients.

The Internet has been transformed from a means to perform simple client-server communication into a large content sharing network. A common phenomenon in content sharing is the flash popularity of a certain content over a short period of time. During this period, a large number of consumers across the world download and enjoy the content using different means. Usually, the method of delivery from the content providers to the consumers involves some monetary transaction (or not for free content) and then a unicast delivery of the content. Another flash popularity trend, especially among the young generation, is that of the peer-trend.

Once a particular piece of content becomes “hip” or stylish among a group of peers, every member of that peer group wants to download and consume that content to be up-to-date with the latest trend. However, a major portion of the content distribution (music, videos, games) in any case follows the “Long Tail” effect [14]. Consequently, only a few content distributors utilize this temporal common interest in recent “popular” content. One of the primary reasons for such conservative behavior on the distributors' part is to prevent piracy and unauthorized distribution [117].

We explored the possibility of utilizing this group based interest in content to reduce the duplicacy in the content delivery. In addition, we also analyzed the effect of client cooperation on the energy efficiency of the client devices. The implemented solution, named GroupDL, works with a group of friends who are interested in the same content. It allows several of these friends to gather and order the content together. The content server, upon receiving a group request, delivers just one copy of the content to the whole group in a way that the group can collaborate and reconstruct copies of the content for each member. Clients download only a portion of the whole content using the cellular interface while retrieving the rest of it from their peers using another interface. The solution also offers a security mechanism to prevent any eavesdropping, and allows a tight grip over the content by the provider.

3.4.1 Cooperation Techniques

Some recent research works have proposed methods of combining the cellular and ad-hoc connectivity of a hand-held device [136, 57, 79, 124, 75, 150] to reduce the amount of power consumption. In addition, the authors of [136] explored the possibility of improving the data download speed by using opportunistic ad-hoc connections among nearby peers. In this way, the nearby peers collaborate and share the downloaded data that might be of interest to other peers. Proposals in [79, 57, 124] are also similar in that they also seek for cooperation among peers; however, they are based on a high-maintenance tree-based node structure, which prevents them from being very energy efficient.

The cooperation technique proposed in Publication VI, GroupDL, utilizes the interest of a group of nearby people (e.g., a group of friends) in the same content. Instead of each member of the group downloading the content individually and generating duplicate traffic over the network, the work proposes a model where the content provider sells a bulk-cooperative

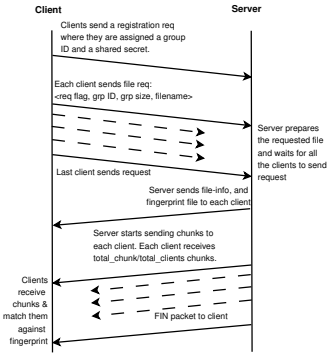

Figure 3.5. GroupDL protocol: The server push. © [2011] IEEE

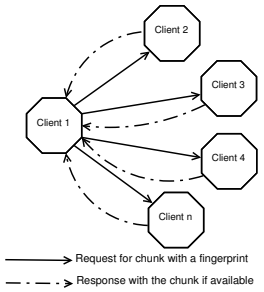

Figure 3.6. GroupDL Protocol: The client exchange. © [2011] IEEE

download. In this model, the interested group buys the bulk content from the provider at a reduced price, and gets entitled to a single copy of the content (as shown in Figure 3.5), which then can be shared among them via a second peer to peer network connection (as shown in Figure 3.6). The whole process of downloading the copy from the provider, and then distributing it among the group members is done using a safe and secure method where the interest of the content provider is preserved. Due to the single unicast download from the content provider, the load on the servers is lower than a traditional content download. Additionally, the consumers receive a lower rate from the provider, which should incentivize the consumers to download in a group rather than individually.

The advantage of GroupDL is twofold. The network needs to carry less duplicate traffic, thus reducing redundancy, and the client terminals' (assuming the consumers are using mobile terminals such as cell phones) energy consumption can be reduced by using a less power hungry interface for peer to peer transfer of data.

3.4.2 Measurement Techniques

To measure the effect of GroupDL over the total network bandwidth, and the energy consumption of client devices, we have performed two different levels of measurement. The first part of the measurement was done using the NS-3 simulator. The simulation provided us with the data of how the amount of duplicate traffic over the Internet is reduced according to the increase of the group size of cooperating users. In addition, measurements were also performed using real cell phones to get a definite idea of the power consumption.

For the prototype implementation, a version of the algorithm was developed in C for the Maemo platform [81], which is a Linux flavored mobile operating system used in Nokia N900 devices. The server was accessed through a 3G (HSPA) connection using a subscription with a maximum 384 kbps download speed, and the maximum WiFi throughput was 10 Mbps. The clients were kept within a room, and the WLAN interfaces of the clients were used in ad-hoc mode for the collaboration process. We have measured 0.9 Watt as an ideal value of 3G interface power usage to compute theoretical predictions, and 1.25 Watt and 1.18 Watt for WiFi transmit and receive operations respectively.

The prototype consists of a GroupDL server accessed over the 3G network and multiple GroupDL clients running on N900 devices. These client devices can turn only one interface on at a time, consequently, after the server-push phase, the interface was changed from 3G to WLAN or Bluetooth to continue with the collaboration part.

3.4.3 Effect of Cooperation on Energy Efficiency

Let us take a look at the measurement performed on real devices to understand the effect of cooperation on energy consumption.

GroupDL allows the content provider to distribute the energy consumption of the service to the edge. As the provider is transmitting only one copy of the content per group, the power usage at the data centers should come down in orders of magnitude. On the other hand, the subscribers have the option of using any of the available local area interfaces (e.g. WiFi, infrared, Bluetooth) for the collaboration part, allowing them to drive down their own energy consumption level to a minimum. Adding to the list of advantages, the usage of the operator base station as well as the congestion factor of the shared cellular spectrum are minimized while

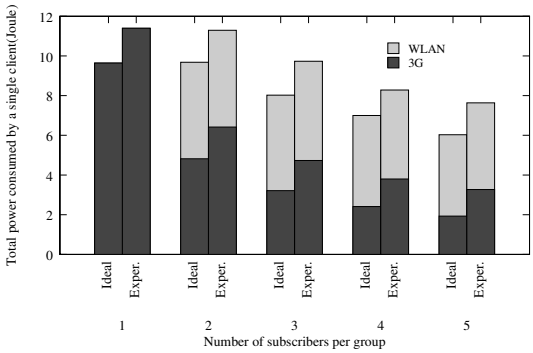

Figure 3.7. Total energy consumption of a client device. © [2011] IEEE

using GroupDL.

We have created a simple theoretical model of the energy consumption profile to understand how much effect the switching from cellular interface to the WiFi interface would have. Details of the theoretical model can be found in Publication VI. However, when we measured the total energy consumption of the N900 devices using GroupDL, the power usage level of the devices revealed many other factors influencing the energy consumption value. Figure 3.7 shows a comparison between the theoretical value and the measured value of a real device. To understand the different influences, let us consider a detailed energy consumption scenario from a mobile device during a download session. Figure 3.8 presents the power usage of a traditional client and a GroupDL client with two and five clients in a group, from top to bottom respectively. The numbered labels in the figure show the different events reflected in the power usage of the device. For the traditional download, three separate events occur during the lifetime of the download process: the device turns the 3G interface on and starts downloading (label 1). Upon completion of the download, the device starts the process of turning off the 3G interface (label 2) and it enters an idle mode (label 3). On the other hand, a client running a collaborative process such as GroupDL passes through more operations: after downloading via the 3G interface (label 4), the device turns the 3G interface off and WLAN interface on (label 5), after which the collaboration process via the WLAN interface starts (label 6) and at the end the WLAN interface is turned off (label 7).

The power usage of the 3G interface is seen to be higher than expected, and the underlying cause of this is the inactivity timers of the radio re-

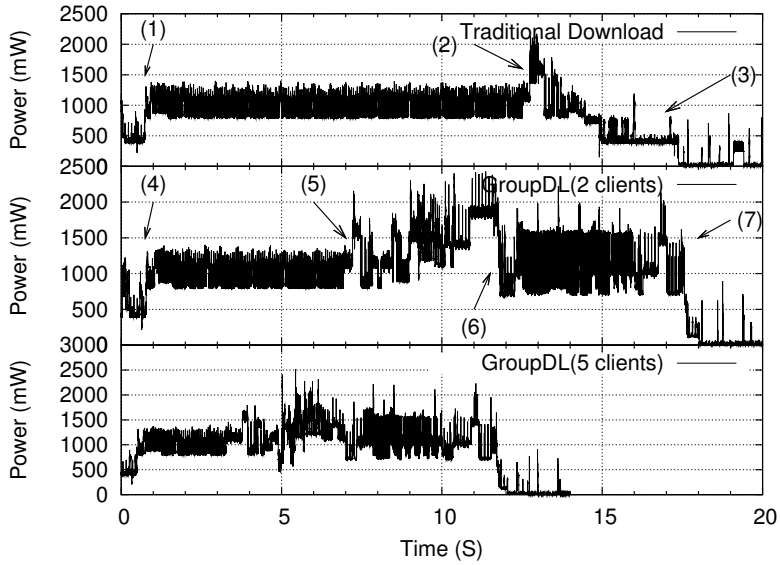

Figure 3.8. Power usage of a GroupDL client. © [2011] IEEE

source controller. These timers control the transition of the 3G interface from one state to another [147]; thus if the incoming packets are spaced at intervals as not to allow these timers to go off, the 3G interface of the mobile device will stay active. A detailed analysis of the power consumption can be found in Publication VI.

This work demonstrates that cooperative content distribution algorithms have a positive effect on both the bandwidth usage and the energy consumption values of mobile hand-held devices. The work also analyzes different network conditions and the 3G network technicalities that can affect the energy consumption of the mobile devices during such operation. Overall, the measurement analysis shows that the energy consumption of mobile devices can be driven down considerably using cooperative download algorithms in addition to less bandwidth usage and better economic features.

3.5 Open Questions and Future Work

In the first phase of our research, we have discussed the content-independent redundancy detection and its effect on processing power and memory consumption. While Publication I and Publication II have extensively discussed the methods of reducing the needed states in memory and the pro-

cessing, there is still scope to improve the elimination process for different parts of the network. The traffic profile for the core network is different from that of the access network, and a profile analysis could greatly improve how the redundancy elimination process is applied to the traffic. On the other hand, the clean-slate Internet architecture proposals, such as ICN's, already name a piece of content according to what it contains instead of the server address. This feature helps the redundancy detection and elimination problem. Nevertheless, how feasible it will be to demand content processing from intermediate routers from an economic point of view remains to be analyzed.

Turning our focus to more futuristic Internet architectures, Publication IV, Publication III and Publication V focus on the caching mechanism of ICN, a new family of internet architecture proposals. Reducing duplicacy is somewhat taken care of in the ICN architecture as the contents are already addressed using what they contain rather than the server address, and thus make it easy to detect duplicacy irrespective of its origin. However, duplicacy still remains in caching. We have proposed several methods of reducing such duplicacy within the caches; however, there is still scope to improve on our proposals. A common source of feedback concerning our detour-based proposals was a concern about latency and congestion. Whether it is feasible to solve the problem of detour without increasing the latency (at all) remains an open question, however, as demonstrated in the publications, latency can definitely be lowered by increasing the level of cache hits. A future proposal can take into account latency as a proximity parameter instead of hop count to make sure that the path with the least latency among the suitable paths are chosen. In addition, the clustering proposal in Publication V can be further improved to take into account the interest of a region, and thus allow intelligent clustering. Further, a hybrid system can be developed including both CDNs and proposed clustering methods to gain the benefits of both worlds. While the CDNs can offer caching services for big enterprises, caching the most popular part of the content popularity distribution, the small router-based content caches implementing the detour proposal can find their own niche by serving the small to medium content providers.

4. Conclusion

The Internet, as of today, is primarily used as a means of content sharing. With the ever increasing user-base of the Internet, it is becoming increasingly difficult for the content providers to cope with the demand. While large corporations resort to content sharing services such as CDNs, small to medium scale enterprises struggle financially to purchase and maintain such a service. Considering that overlay caching services are already provided by the contemporary Internet, and there is the promise of even better in-network caching services by clean-slate designs such as ICN, the opportunities for using these caches to alleviate the aforementioned problem are immense. As the current caching practices encourage content duplication to ensure that popular contents are available by the shortest possible route, the majority of the content is swapped out of the caches to accommodate this redundancy. On the other hand, due to large scale duplication of content by the consumers, redundancy at a content level also rises very rapidly in the core transport network, which cannot be detected by traditional methods due to various reasons such as different file names etc.

This work explores the aforementioned phenomena, and proposes several solutions to reduce redundancy in the network by either efficiently detecting and removing the inter-content similarities, or by utilizing the available cache space of the network more intelligently to cache more objects than are possible today. This dissertation demonstrates that by efficiently detecting inter-content duplication, it is possible to reduce a high proportion of duplicacy without incurring expensive data or processing overheads.

In addition to reducing the content redundancy of the data packets passing through the network, it is also important to utilize the existing caches in the network efficiently. While most of the existing works look at the

caches from a latency reduction point of view, this dissertation looks at the problem from a different viewpoint and demonstrates the effect of the caches if they are optimized to utilize space and reduce duplicity. The experimental results show that it is possible to increase the cached contents drastically by making sure that duplication is minimized in the caches.

A supplementary work of this dissertation looks at the impact of redundancy elimination on the energy consumption of the mobile devices. During the distribution of popular contents, a redundancy aware distribution mechanism can make sure that the traffic is duplicated until the last mile as little as possible, and that the clients are using less energy-hungry methods of downloading. As a result, the congestion at the core network is reduced, and the energy consumption of the end-devices is minimized.

Bibliography

[1] 3GPP. Radio Resource Control (RRC) Protocol specification. *3GPP TS 25.331*, May 1999.

[2] 3GPP. E-UTRA; Radio Resource Control (RRC) Protocol Specification. *3GPP TS 36.331*, May 2008.

[3] The FP7 4WARD Project. `http://www.4ward-project.eu/`, August 2014.

[4] L. A. Adamic. Zipf, power-laws, and pareto - a ranking tutorial. `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`, 2002.

[5] B. Ager, F. Schneider, Juhoon Kim, and A Feldmann. Revisiting cacheability in times of user generated content. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–6, March 2010.

[6] Bengt Ahlgren, Matteo D'Ambrosio, Marco Marchisio, Ian Marsh, Christian Dannewitz, Börje Ohlman, Kostas Pentikousis, Ove Strandberg, René Rembarz, and Vinicio Vercellone. Design considerations for a network of information. In *Proceedings of the 2008 ACM CoNEXT Conference*, page 66. ACM, 2008.

[7] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[8] M. Ajtai, , R. Burns, R. Fagin, D.D.E. Long, and L. Stockmeyer. Compactly encoding unstructured inputs with differential compression. *Journal of the ACM (JACM)*, 49(3):318–367, 2002.

[9] Sebastian Altmeyer and Claire Maiza Burguière. Cache-related preemption delay via useful cache blocks: Survey and redefinition. *Journal of Systems Architecture*, 57(7):707–719, 2011.

[10] A. Anand, V. Sekar, and A. Akella. SmartRE: an architecture for coordinated network-wide redundancy elimination. In *Proc. of SIGCOMM*, pages 87–98, 2009.

[11] Ashok Anand, Chitra Muthukrishnan, Aditya Akella, and Ramachandran Ramjee. Redundancy in network traffic: findings and implications. *ACM SIGMETRICS Performance Evaluation Review*, 37(1):37–48, 2009.

[12] Manish Anand, Edmund B. Nightingale, and Jason Flinn. Self-tuning wireless network power management. In *Proceedings of the 9th Annual*

International Conference on Mobile Computing and Networking, MobiCom '03, pages 176–189, New York, NY, USA, 2003. ACM.

[13] G. Anastasi, M. Conti, E. Gregori, and A. Passarella. 802.11 power-saving mode for mobile computing in wi-fi hotspots: Limitations, enhancements and open issues. *Wirel. Netw.*, 14(6):745–768, December 2008.

[14] C. Anderson. *The long tail: How endless choice is creating unlimited demand*. Random House Business Books, 2006.

[15] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004.

[16] Anirudh Badam, KyoungSoo Park, Vivek S. Pai, and Larry L. Peterson. Hashcache: Cache storage for the next billion. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 123–136, Berkeley, CA, USA, 2009. USENIX Association.

[17] Susmit Bagchi. A fuzzy algorithm for dynamically adaptive multimedia streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, 7(2):11:1–11:26, March 2011.

[18] P. Baran. On distributed communications networks. *Communications Systems, IEEE Transactions on*, 12(1):1–9, March 1964.

[19] Using BDC technology to update Windows operating system. `http://www.microsoft.com/en-us/download/details.aspx?id=1562`, August 2014.

[20] Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[21] D. Bertozzi, L. Benini, and B. Ricco. Power aware network interface management for streaming multimedia. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 2, pages 926–930 vol.2, Mar 2002.

[22] N. Bjørner, A. Blass, and Y. Gurevich. Content-dependent chunking for differential compression, the local maximum approach. *Journal of Computer and System Sciences*, 76(3-4):154–203, 2010.

[23] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[24] G.W. Brock. *The second information revolution*. Harvard University Press, 2003.

[25] A.Z. Broder. On the resemblance and containment of documents. In *Proc. Compression and Complexity of Sequences*, pages 21–29. IEEE, 2002.

[26] Wei Chai, Diliang He, Ioannis Psaras, and George Pavlou. Cache "less for more" in information-centric networks. *NETWORKING 2012*, pages 27–40, 2012.

[27] Wei Koong Chai, Ning Wang, I. Psaras, G. Pavlou, Chaojiong Wang, G.G. de Blas, F.J. Ramon-Salguero, Lei Liang, S. Spirou, A. Beben, and E. Hadjioannou. Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services. *Communications Magazine, IEEE*, 49(3):112–120, 2011.

[28] Surendar Chandra. Wireless network interface energy consumption: Implications for popular streaming formats. *Multimedia Syst.*, 9(2):185–201, August 2003.

[29] David R. Cheriton and Mark Gritter. Triad: A scalable deployable nat-based internet architecture. Technical report, Stanford University, 2000.

[30] Cisco. The Zettabyte Era–Trends and Analysis. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html`, June 2014.

[31] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[32] COntent Mediator architecture for content-aware nETworks. `http://www.comet-project.org/`, August 2014.

[33] M. Cotton, L. Vegoda, and D. Meyer. IANA Guidelines for IPv4 Multicast Address Assignments. RFC 5771 (Best Current Practice), March 2010.

[34] Matteo D'Ambrosio, Christian Dannewitz, Holger Karl, and Vinicio Vercellone. MDHT: A hierarchical name resolution service for information-centric networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 7–12, New York, NY, USA, 2011. ACM.

[35] Christian Dannewitz. Netinf: An information-centric design for the future internet. In *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.

[36] Christian Dannewitz, Dirk Kutscher, Börje Ohlman, Stephen Farrell, Bengt Ahlgren, and Holger Karl. Network of information (netinf)–an information-centric networking architecture. *Computer Communications*, 36(7):721–735, 2013.

[37] Defence advanced research projects agency. `http://www.darpa.mil/default.aspx`, August 2014.

[38] D. Davies. The control of congestion in packet-switching networks. *Communications, IEEE Transactions on*, 20(3):546–550, Jun 1972.

[39] D. W. Davies, K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. A digital communication network for computers giving rapid response at remote terminals. In *Proceedings of the First ACM Symposium on Operating System Principles*, SOSP '67, pages 2.1–2.17, New York, NY, USA, 1967. ACM.

[40] IETF Decoupled Application Data Enroute (DECADE) Workgroup. `http://datatracker.ietf.org/wg/decade/documents/`, August 2014.

[41] S. Deering and R. Hinden. *RFC 2460 Internet Protocol, Version 6 (IPv6) Specification*. Internet Engineering Task Force, December 1998.

[42] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (INTERNET STANDARD), August 1989. Updated by RFC 2236.

[43] Mohamed Diallo, Vasilis Sourlas, Paris Flegkas, Serge Fdida, and Leandros Tassiulas. A content-based publish/subscribe framework for large-scale content delivery. *Computer Networks*, 57(4):924–943, 2013.

[44] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[45] Frederick Douglis, Purushottam Kulkarni, Jason D LaVoie, and John Michael Tracey. Method and apparatus for data redundancy elimination at the block level, March 13 2012. US Patent 8,135,683.

[46] Jeffrey Erman, Alexandre Gerber, Mohammad T. Hajiaghayi, Dan Pei, and Oliver Spatscheck. Network-aware forward caching. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 291–300, New York, NY, USA, 2009. ACM.

[47] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 281–287, New York, NY, USA, 2010. ACM.

[48] 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.

[49] Peyman Faratin, David D Clark, Steven Bauer, and William Lehr. Complexity of internet interconnections: Technology, incentives and implications for policy. In *35th Research Conference on Communication, Information and Internet Policy (TPRC)*. TRC, 2007.

[50] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, KC 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*, pages 147–158. ACM, 2013.

[51] Nikos Fotiou, Pekka Nikander, Dirk Trossen, and George C Polyzos. Developing information networking further: From PSIRP to PURSUIT. In *Broadband Communications, Networks, and Systems*, pages 1–13. Springer, 2012.

[52] Gerardo García, Andrzej Beben, Francisco J Ramón, Adrián Maeso, Ioannis Psaras, George Pavlou, Ning Wang, Jaroslaw Sliwinski, Spiros Spirou, Sergios Soursos, et al. Comet: Content mediator architecture for content-aware networks. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, 2011.

[53] Ali Ghodsi, Teemu Koponen, Jarno Rajahalme, Pasi Sarolahti, and Scott Shenker. Naming in content-oriented architectures. In *SIGCOMM workshop on ICN*, pages 1–6. ACM, 2011.

[54] E. Halepovic, C. Williamson, and M. Ghaderi. Exploiting Non-Uniformities in Redundant Traffic Elimination. Technical report, University of Calgary, 2010. http://hdl.handle.net/1880/48155.

[55] Markus Hofmann and Leland R Beaumont. *Content networking: architecture, protocols, and practice*. Elsevier, 2005.

[56] H. Holbrook, B. Cain, and B. Haberman. Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard), August 2006.

[57] H.Y. Hsieh and R. Sivakumar. On using peer-to-peer communication in cellular wireless data networks. *Mobile Computing, IEEE Transactions on*, 3(1):57–72, 2005.

[58] Bradley Huffaker, Young Hyun, Dan Andersen, and KC Claffy. The Skitter AS Links Dataset. `http://www.caida.org/data/active/skitter_aslinks_dataset.xml`, 2012.

[59] IEEE. IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pages i–513, 2003.

[60] S. Ihm, K.S. Park, and V.S. Pai. Wide-area network acceleration for the developing world. In *Proc. of the 2010 USENIX ATC*, page 18, 2010.

[61] Internet Engineering Task Force. *RFC 791 Internet Protocol - DARPA Internet Program, Protocol Specification*, September 1981.

[62] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 213–222, 2002.

[63] 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 CoNEXT*, pages 1–12. ACM, 2009.

[64] M Johnsson, J Huusko, T Frantti, F Andersen, T Nguyen, and M Leon. Towards a new architecture framework—the nth stratum concept methods, 2008.

[65] Jussi Kangasharju, James Roberts, and Keith W Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376–383, 2002.

[66] K. Katsaros, G. Xylomenos, and G.C. Polyzos. A hybrid overlay multicast and caching scheme for information-centric networking. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–6, March 2010.

[67] Konstantinos Katsaros, Nikos Fotiou, Xenofon Vasilakos, ChristopherN. Ververidis, Christos Tsilopoulos, George Xylomenos, and GeorgeC. Polyzos. On inter-domain name resolution for information-centric networks. In *NETWORKING 2012*, volume 7289 of *Lecture Notes in Computer Science*, pages 13–26. Springer Berlin Heidelberg, 2012.

[68] Konstantinos Katsaros, George Xylomenos, and George C Polyzos. Multicache: An overlay architecture for information-centric networking. *Computer Networks*, 55(4):936–947, 2011.

[69] Leonard Kleinrock. *Information Flow in Large Communication Nets*. Ph.d. thesis proposal, Massachusetts Institute of Technology, May 1961.

[70] Leonard Kleinrock. *Communication Nets; Stochastic Message Flow and Delay*. Mcgraw-Hill (New York), 1964.

[71] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, 2007.

[72] Ronny Krashinsky and Hari Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. *Wirel. Netw.*, 11(1-2):135–148, January 2005.

[73] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. *SIGCOMM Comput. Commun. Rev.*, 41(4), 2010.

[74] Dmitrij Lagutin, Kari Visala, and Sasu Tarkoma. Publish/Subscribe for Internet: PSIRP Perspective. *Future Internet Assembly*, 84, 2010.

[75] L.K. Law, S.V. Krishnamurthy, and M. Faloutsos. Capacity of hybrid cellular-ad hoc data networks. In *INFOCOM 2008.*, pages 1606–1614. IEEE, 2008.

[76] Chi-Chen Lee, Jui-Hung Yeh, and Jyh-Cheng Chen. Impact of inactivity timer on energy consumption in wcdma and cdma2000. In *Wireless Telecommunications Symposium, 2004*, pages 15–24, May 2004.

[77] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in vanet. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, 2006.

[78] 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.

[79] H. Luo, X. Meng, R. Ramjee, P. Sinha, and L.E. Li. The design and evaluation of unified cellular and ad hoc networks. *IEEE Transactions on Mobile Computing*, pages 1060–1074, 2007.

[80] R.T.B. Ma, Dah Ming Chiu, J.C.S. Lui, V. Misra, and D. Rubenstein. On cooperative settlement between content, transit, and eyeball internet service providers. *Networking, IEEE/ACM Transactions on*, 19(3):802–815, June 2011.

[81] Home of the Maemo community, 2011. `http://maemo.org/`.

[82] Alberto Medina, Ibrahim Matta, and John Byers. BRITE: A Flexible Generator of Internet Topologies. Technical report, Boston University, Boston, MA, USA, 2000.

[83] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, July 1976.

[84] Zhongxing Ming, Mingwei Xu, and Dan Wang. Age-based cooperative caching in information-centric networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 268–273. IEEE, 2012.

[85] Sparsh Mittal. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems*, 4(1):33–43, 2014.

[86] J. Moy. RFC 1131: OSPF specification, October 1989. Obsoleted by RFC1247 [89]. Status: PROPOSED STANDARD.

[87] J. Moy. RFC 1245: OSPF protocol analysis, July 1991. See also RFC1247, RFC1246 [89, 88]. Status: INFORMATIONAL.

[88] J. Moy. RFC 1246: Experience with the OSPF protocol, July 1991. See also RFC1247, RFC1245 [89, 87]. Status: INFORMATIONAL.

[89] J. Moy. RFC 1247: OSPF version 2, July 1991. See also RFC1246, RFC1245 [88, 87]. Obsoleted by RFC1583 [90]. Obsoletes RFC1131 [86]. Status: DRAFT STANDARD.

[90] J. Moy. RFC 1583: OSPF version 2, March 1994. Obsoleted by RFC2178 [91]. Obsoletes RFC1247 [89]. Status: DRAFT STANDARD.

[91] J. Moy. RFC 2178: OSPF version 2, July 1997. Obsoleted by RFC2328 [92]. Obsoletes RFC1583 [90]. Status: DRAFT STANDARD.

[92] J. Moy. RFC 2328: OSPF version 2, April 1998. See also STD0054 [93]. Obsoletes RFC2178 [91]. Status: STANDARD.

[93] J. Moy. STD 54: OSPF version 2, April 1998. See also RFC2328 [92].

[94] A. Muthitacharoen, B. Chen, and D. Mazieres. A low-bandwidth network file system. In *Proc. of SOSP*, pages 174–187, 2001.

[95] Suman Nath, Zachary Anderson, and Srinivasan Seshan. Choosing beacon periods to improve response times for wireless http clients. In *Proceedings of the Second International Workshop on Mobility Management &Amp; Wireless Access Protocols*, MobiWac '04, pages 43–50, New York, NY, USA, 2004. ACM.

[96] Named Data Networking. `http://named-data.net/`, August 2014.

[97] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.

[98] Vitria's publish-subscribe architecture: Publish-subscribe overview. `http://www.vitria.com/`, 1998.

[99] D. Oran. RFC 1142: OSI IS-IS intra-domain routing protocol, February 1990. Status: INFORMATIONAL.

[100] PeerApp: P2P and Media Caching. http://www.peerapp.com, July 2014.

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

[102] Andrea Passarella. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Computer Communications*, 35(1):1–32, 2012.

[103] Diego Perino, Matteo Varvello, and Krishna PN Puttaswamy. ICN-RE: redundancy elimination for information-centric networking. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 91–96. ACM, 2012.

[104] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

[105] J. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force, August 1980.

[106] John Postel. Transmission control protocol. RFC 793, Internet Engineering Task Force, September 1981.

[107] Petteri Pöyhönen. The network of information: Architecture and applications. http://www.sail-project.eu/wp-content/uploads/2011/08/SAIL_DB1_v1_0_final-Public.pdf, 2012.

[108] I. Psaras, W.K. Chai, and G. Pavlou. To cache or not to cache: Probabilistic in-network caching for information-centric networks. In *ACM SIGCOMM Workshop on ICN*, Helsinki, Finland, 2012.

[109] Publish-Subscribe Internet Routing Paradigm. http://www.psirp.org/, August 2014.

[110] EU FP7 PURSUIT. http://www.fp7-pursuit.eu/, August 2014.

[111] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling resource usage for mobile applications: A cross-layer approach. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 321–334, New York, NY, USA, 2011. ACM.

[112] D. Qiao and K.G. Shin. Smart power-saving mode for ieee 802.11 wireless lans. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1573–1583 vol. 3, March 2005.

[113] M.O. Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ. Harvard, 1981.

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

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

[116] Lawrence G. Roberts. Multiple computer networks and intercomputer communication. In *Proceedings of the First ACM Symposium on Operating System Principles*, SOSP '67, pages 3.1–3.6, New York, NY, USA, 1967. ACM.

[117] Pablo Rodriguez, SeeMong Tan, and Christos Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *SIGCOMM Comput. Commun. Rev.*, 36:75–78, January 2006.

[118] E. J. Rosensweig and Jim Kurose. Breadcrumbs: Efficient, best-effort content location in cache networks. In *INFOCOM*, 2009.

[119] Scalable and Adaptive Internet Solutions. `http://www.sail-project.eu/`, August 2014.

[120] Y. Song, K. Guo, and L. Gao. Redundancy-Aware Routing with Limited Resources. In *Proc. of 19th IEEE ICCCN*, pages 1–6, 2010.

[121] N.T. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proc. of SIGCOMM*, pages 87–95, 2000.

[122] Squid web proxy cache. `http://www.squid-cache.org/`, July 2014.

[123] W. Richard Stevens. *TCP/IP Illustrated (Vol. 1): The Protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[124] M. Stiemerling and S. Kiesel. Cooperative P2P Video Streaming for Mobile Peers. In *Proc. of 19th Int. Conf. on Computer Comm. and Networks (ICCCN)*, pages 1–7. IEEE, 2010.

[125] Torsten Suel and Nasir Memon. Algorithms for delta compression and remote file synchronization. In *In Khalid Sayood, editor, Lossless Compression Handbook*. Academic Press, 2002.

[126] K. Tangwongsan, , H. Pucha, D.G. Andersen, and M. Kaminsky. Efficient similarity estimation for systems exploiting data redundancy. In *Proc. of IEEE INFOCOM*, pages 1–9, 2010.

[127] Sasu Tarkoma. *Publish / Subscribe Systems: Design and Principles*. Wiley Publishing, 1st edition, 2012.

[128] Sasu Tarkoma, Mark Ain, and Kari Visala. The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture. In *Future Internet Assembly*, pages 102–111, 2009.

[129] Andrew Tridgell, Paul Mackerras, et al. The rsync algorithm, 1996.

[130] C. Tsilopoulos, G. Xylomenos, and Y. Thomas. Reducing forwarding state in content-centric networks with semi-stateless forwarding. In *INFOCOM, 2014 Proceedings IEEE*, pages 2067–2075, April 2014.

[131] Christos Tsilopoulos and George Xylomenos. Supporting diverse traffic types in information centric networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 13–18, New York, NY, USA, 2011. ACM.

[132] Anna Ukhanova, Evgeny Belyaev, Le Wang, and Søren Forchhammer. Power consumption analysis of constant bit rate video transmission over 3g networks. *Computer Communications*, 35(14):1695 – 1706, 2012. Special issue: Wireless Green Communications and Networking.

[133] International Telecommunication Union. The World in 2011: ICT Facts and Figures. `http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf`, 2011.

[134] Quick start guide to increase data center energy efficiency, 2011. `http://www1.eere.energy.gov/femp/pdfs/data_center_qsguide.pdf`.

[135] Report to congress on server and data center energy efficiency, 2007. `http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf`.

[136] Long Vu, Ivica Rimac, Volker Hilt, Markus Hofmann, and Klara Nahrstedt. ishare: Exploiting opportunistic ad hoc connections for improving data download of cellular users. *IDEALS @ University of Illinois*, 2009.

[137] Matthias Wählisch, Thomas C. Schmidt, and Markus Vahlenkamp. Bulk of Interest: Performance Measurement of Content-Centric Routing. In *Proc. of ACM SIGCOMM, Poster Session*, pages 99–100, New York, August 2012. ACM.

[138] Matthias Wählisch, Thomas C. Schmidt, and Markus Vahlenkamp. Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure. *Computer Networks*, 57(16):3192–3206, Nov. 2013.

[139] Matthias Wählisch, Thomas C. Schmidt, and Markus Vahlenkamp. Lessons from the past: Why data-driven states harm future information-centric networking. In *Proc. of IFIP Networking*, Piscataway, NJ, USA, 2013. IEEE Press.

[140] Jia Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[141] Liang Wang, Walter Wong, and Jussi Kangasharju. In-network caching vs. redundancy elimination. *arXiv preprint arXiv:1311.7421*, 2013.

[142] Duane Wessels and Kim Claffy. ICP and the Squid web cache. *Selected Areas in Communications, IEEE Journal on*, 16(3):345–357, 1998.

[143] Alec Wolman, Geoff Voelker, Nitin Sharma, Neal Cardwell, Molly Brown, Tashana Landray, Denise Pinnel, Anna Karlin, and Henry Levy. Organization-based analysis of web-object sharing and caching. In *Proceedings of the 2Nd Conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*, USITS'99, pages 3–3, Berkeley, CA, USA, 1999. USENIX Association.

[144] Alec Wolman, M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. *SIGOPS Oper. Syst. Rev.*, 33(5):16–31, December 1999.

[145] K-Y Wong. Web cache replacement policies: a pragmatic approach. *Network, IEEE*, 20(1):28–34, 2006.

[146] G. Xylomenos, C.N. Ververidis, V.A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K.V. Katsaros, and G.C. Polyzos. A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, 16(2):1024–1049, Second 2014.

[147] S.R. Yang, S.Y. Yan, and H.N. Hung. Modeling UMTS power saving with bursty packet data traffic. *IEEE Transactions on Mobile Computing*, pages 1398–1409, 2007.

[148] Guoqiang Zhang, Yang Li, and Tao Lin. Caching in information centric networking: a survey. *Computer Networks*, 57(16):3128–3141, 2013.

[149] Hai Zhang. Architecture of network and client-server model. *CoRR*, abs/1307.6665, 2013.

[150] Q. Zhang, F.H.P. Fitzek, and Marcos Katz. Evolution of heterogeneous wireless networks: Towards cooperative networks. In *3nd International Conference of the Center for Information and Communication Technologies (CICT)*, Nov 2006.

[151] H. Zimmermann. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. In C. Partridge, editor, *Innovations in Internetworking*, pages 2–9. Artech House, Inc., Norwood, MA, USA, 1988.

The transformation of the Internet from a client-server based paradigm to a content-based one has led to many of the fundamental network designs becoming outdated. The increase in user-generated contents brings forward the needs for designing an Internet which is ready for these and can handle the needs of the small-scale content providers. The Internet today carries and stores a large amount of redundant data, primarily due to a lack of duplication detection mechanisms. This redundancy costs the network in different ways: it consumes energy from the network elements; it makes the network caches store duplicate data, thus causing the tail of the data distribution to be swapped out of the caches; and it causes the content-servers to be loaded more as they have to always serve the less popular contents. In this dissertation, we have analyzed the aforementioned phenomena and proposed several methods to reduce the redundancy of the network at a low cost.

ISBN 978-952-60-6421-5 (printed)
ISBN 978-952-60-6422-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Computer Science and Engineering
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS

Aalto-DD 153/2015