Toward the Realization of Large-Scale and Highly-Efficient Information-Centric Networking

Toward the Realization of Large-Scale and Highly-Efficient Information-Centric Networking

Ryo Nakamura

March 2020

Department of Informatics

Graduate School of Science and Technology

Kwansei Gakuin University

Abstract

In the recent years, Information-Centric Networking (ICN) that mainly focuses on contents that are transferred and received instead on end hosts that transmit and receive contents has been under the spotlight. In particular, CCN (Content-Centric Networking) and NDN (Named Data Networking) among ICNs have been attracting attention as promising network architectures for realizing ICN.

Notable features of ICN architectures compared with the conventional TCP/IP network are adoption of unique content identifiers, location independence, and in-network content caching. In an ICN, contents are stored in one or more content providers. The primary objective of ICNs are efficient content delivery from content providers to content consumers called entities. A requesting entity injects a content request into the ICN, which tries to deliver the content request to nearby content provider(s) through routers. The content is sent back from the content provider to the requesting entity. Because of in-network caching, the content might be directly sent back from one of caching routers.

The performance of ICNs has been actively studied in the literature, however, to realize global-scale ICNs, it is crucial to clarify the scalability of ICNs regarding the number of nodes (i.e., the network size). Furthermore, to realize large-scale ICNs in real network as a communication infrastructure, it is also important to improve the efficiency of ICNs as well as to reveal the scalability of ICNs.

In this thesis, we tackle to research issues on realizing large-scale and highly-efficient ICN. Specifically, we investigate the scalability of ICNs in terms of the network size using experiments and mathematical analyses. Also, to improve the efficiency of ICNs, we investigate the optimality of the shortest-path routing in ICN, and

propose a lossy link detection mechanism for CCN.

First, we focus on CCN, which is one of major network architectures realizing ICN, and investigate the scalability of CCNx, open-source CCN implementation, in terms of the number of nodes. As performance metrics, we measure the total throughput of content deliveries, the packet loss ratio in the network, and the average content delivery time. We also examine the performance bottleneck of CCNx through system-wide profiling, which quantitatively shows that per-packet digest-based authentication is the performance bottleneck in CCNx. Our findings include that the communication performance was degraded when the number of CCN routers exceeds 30–40, and that the Data-chunck digest computation consumes approximately 20% of the total CPU time. As a result of estimating the impact of hardware-offloading of Data-chunk digest computation, we found that the average content delivery time can be significantly reduced.

Secondly, we analytically obtain performance metrics for CCN using the MCA (Multi-Cache Approximation) algorithm. Our analytical model contains multiple routers, multiple repositories, and multiple entities. We obtain three performance metrics: content delivery delay (i.e., the average time required for an entity to retrieve a content through a neighboring router), throughput (i.e., number of contents delivered from an entity per unit of time), and availability (i.e., probability that an entity can successfully retrieve a content from a network). Through several numerical examples, we investigate how network topology affects the performance of CCN. A notable finding is that content caching becomes more beneficial in terms of content delivery time and availability (resp., throughput) as distance between the entity and the requesting repository narrows (resp., widens).

Thirdly, we focus on a large-scale ICN and reveal the scaling property of ICN. For answering research questions regarding the scaling property of ICN, we derive the cache hit probability at each router, the average content delivery delay of each entity, and the average content delivery delay of all entities over a content distribution tree comprised of a single repository, multiple routers, and multiple entities. Through several numerical examples, we investigate the effect of the topology and the size

of the content distribution tree and the cache size at routers on the average content delivery delay of all entities. Our findings include that the average content delivery delay of ICNs converges to a constant value if the cache size of routers are not small, which implies high scalability of ICNs, and that even when the network size would grow indefinitely, the average content delivery delay is upper-bounded by a constant value if routers in the network are provided with a fair amount of content caches.

Fourthly, we investigate the optimality of the shortest-path routing that is a straightforward approach for content routing in ICNs. Namely, we try to answer research questions regarding the optimality of the shortest-path routing. We compare the application-level performances with the shortest-path routing and with the optimal routing obtained by searching all detour paths existing in the vicinity of the shortest-path routing (optimal k-hop detour routing). Our findings include that the shortest-path routing is suitable when the network is balanced and cache sizes at routers are homogeneous, and that the optimal k-hop detour routing is suitable when the network is unbalanced and variation in cache sizes is large.

Finally, by extending a packet loss detection mechanism called Interest ACKnowledgement (ACK), we propose a lossy link detection mechanism called LLD-IA (Lossy Link Detection with Interest ACKs), which is a mechanism for an entity to estimate the link where the packet was discarded in a network. Also, we show that LLD-IA can effectively detect links where packets were discarded under moderate packet loss ratios through simulations.

Acknowledgements

I would like to express my sincere appreciation to my supervisor, Professor Hiroyuki Ohsaki of Kwansei Gakuin University, for his great help and guidance. Without his enthusiasm and continuous support, this thesis would not have been completed.

I would like to extend my appreciation to Professor Hiroyoshi Miwa and Associate Professor Yusuke Sakumoto of Kwansei Gakuin University for their insightful comments and advice to this thesis. Moreover, I would like to express my appreciation to Professor Takeshi Kawabata and Professor Nagisa Ishiura of Kwansei Gakuin University. They have given me warm comments and advice.

I am heartily thankful to Assistant Professor Sho Tsugawa of University of Tsukuba. He gave me valuable comments and warm encouragement.

I am heartily thankful to all the members in the Network Architecture laboratory with Kwansei Gakuin University for their support. Especially, I would like to thank to Dr. Yasuhiro Yamasaki for his insightful comments and advice on our collaborative work. I also would like to thank to Mr. Ryotaro Matsuo, Mr. Yuichi Yasuda, Mr. Kazuyuki Yamasahita, Mr. Ryota Sakaguchi, Mr. Chuta Minamiguchi, and Mr. Ryuichiro Maegawa for valuable discussions.

Finally, I would like to greatly thank to my family for their warm support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the recent years, Information-Centric Networking (ICN) that mainly focuses on contents that are transmitted and received instead on end hosts that transmit and receive contents has been under the spotlight [1-3].

Notable features of ICN architectures compared with the conventional TCP/IP network are adoption of unique content identifiers, location independence, and in-network content caching [3]. In an ICN, *contents* are stored in one or more *content providers*. The primary objective of ICNs are efficient content delivery from content providers to content consumers called *entities*. A requesting entity injects a *content request* into the ICN, which tries to deliver the content request to nearby content provider(s) through *routers*. The content is sent back from the content provider to the requesting entity. Because of in-network caching, the content might be directly sent back from one of *caching routers*.

In the literature, several ICN architectures have been proposed, each of which has commonalities and differences with others. Two of the most popular ICN architectures are CCN (Content-Centric Networking) [1] and NDN (Named Data Networking) [2]. In CCN and NDN, routers in a network can cache the forwarded contents to its buffer memory, and can return requested contents from the buffer memory. As a consequence, CCN and NDN are expected to reduce content delivery delays from a repository (i.e., content provider) to entities and traffic volume transferred through the network.

ICN can be regarded as a cache network, which is significantly different from conventional TCP/IP networks, so many performance studies of ICNs have been performed in the literature. Performance studies of ICN can be classified into two categories: performance evaluation through experiments and simulations [4-8] and mathematical analyses [9-15]. Through those studies, the effectiveness of ICNs have been clarified in terms of system-level performance metrics (e.g., cache hit ratio at a router) as well as user-level performance metrics (e.g., content delivery delay and throughput).

Since ICNs adopt different communication paradigm from the conventional TCP/IP networks (e.g., name-based communication and in-network caching), a variety of research studies rather than performance studies have been performed. In particular, one of fundamental research topics is to design content caching algorithm, which effectively utilizes network resources [16]. In addition to designing content caching algorithms, several content routing mechanisms which utilize caches at intermediate routers called cache-aware routings have been proposed [17, 18]. Also, multiple transport protocols for ICNs which consider differences between ICNs and the conventional TCP/IP network have been proposed [19-25].

The performance of ICN has been actively studied as described above, however, to realize global-scale ICNs, it is crucial to clarify *the scalability of ICNs* in terms of the number of nodes (i.e., the network size) and the number of contents in the network. In this thesis, we focus on the scalability of ICNs regarding the network size (i.e., the number of consumers, routers, and content providers).

Furthermore, to realize large-scale ICNs in real network as a communication infrastructure, it is also important to improve *the efficiency of ICNs* as well as to reveal the scalability of ICNs. A lot of studies have been devoted to improve ICNs, however, there exist several research issues on the following ICN-specific problems; what content routing should be used among the conventional shortest-path routing and cache-aware routings proposed in past studies, and how transport protocol for ICNs should be designed.

In this thesis, we tackle to research issues on realizing large-scale and highly-

efficient ICN networks. Specifically, we investigate the scalability of ICNs in terms of the network size using experiments and mathematical analyses. Furthermore, to improve the efficiency of ICNs, we investigate the optimality of the shortest-path routing in ICN, and propose a lossy link detection mechanism for CCN which helps to design efficient congestion controls for ICNs.

In Chapter 2, we focus on CCN, which is one of major network architectures realizing ICN, and investigate the scalability of CCNx, open-source CCN implementation, in terms of the number of nodes. As performance metrics, we measure the total throughput of content deliveries, the packet loss ratio in the network, and the average content delivery time. We also examine the performance bottleneck of CCNx through system-wide profiling to improve the scalability of CCNx,

In Chapter 3, we analytically derive content delivery delay (i.e., the average time required for an entity to retrieve a content through a neighboring router), throughput (i.e., number of contents delivered from an entity per unit of time), and availability (i.e., probability that an entity can successfully retrieve a content from a network) on an arbitrary CCN network. Through several numerical examples, we investigate how network topology affects the performance of CCN.

In Chapter 4, we focus on a large-scale ICN and reveal the scaling property of ICN. For answering research questions regarding the scaling property of ICN, we derive the cache hit probability at each router, the average content delivery delay of each entity, and the average content delivery delay of all entities over *a content distribution tree* comprised of a single repository, multiple routers, and multiple entities. Through several numerical examples, we investigate the effect of the topology and the size of the content distribution tree and the cache size at routers on the average content delivery delay of all entities.

In Chapter 5, we investigate the optimality of the shortest-path routing in ICN through several experiments. Specifically, using our mathematical analysis of ICN in Chapter 3, we compare the application-level performances with the shortest-path routing and with the optimal routing obtained by searching all detour paths existing in the vicinity of the shortest-path routing (optimal k-hop detour routing).

In Chapter 6, we propose a lossy link detection mechanism called LLD-IA (Lossy Link Detection with Interest ACKs), which is a mechanism for an entity to estimate the link where the packet was discarded in a network. LLD-IA is an extension of a fast packet loss detection mechanism called Interest ACK. Through simulations, we investigate how accurately LLD-IA can detect links where packets were discarded under moderate packet loss ratios.

Finally, in Chapter 7, we summarize this thesis and address future directions.

Chapter 2

Performance Evaluation and Improvement of Large-Scale Content-Centric Networking

2.1 Introduction

In the recent years, Content-Centric Networking (CCN) [1] has been under the spotlight as one of the information-centric networks, which primarily focus on contents transmitted within the network rather than on hosts sending/receiving those contents. CCN adopts a request-and-response communication model. In CCN, a unique identifier is assigned to every content. The content request packet (*Interest packet*) from a user is routed among CCN routers using longest-prefix matching of the content identifier to search for the location of the content. The content discovered is

This chapter is a minor version of [26].

delivered to the user as a response packet (*Data packet*) by retracing the path of the request packet.

A CCN router has a buffer memory called CS (Content Store), and it caches the forwarded Data packet in the buffer memory. CCN routers on a network cache the forwarded contents, and reuse data. When a CCN router receives another Interest packet for the same content, it returns the Data packet from the cache so that the amount of traffic in network can be reduced and the content delivery time can be shortened.

In the literature, the effectiveness of CCN has been investigated mainly through simulation experiments. On the contrary, a software implementation called CCNx [27] has been developed as an open-source software, and several performance studies of CCN through experiments have been performed.

However, to the best of our knowledge, scalability of CCN in terms of the number of nodes has not been fully understood. For large-scale deployment of CCN in real networks as a communication infrastructure, it is crucial to clarify how the CCN architecture itself and its components such as CCN routers and repositories are scalable in terms of the number of nodes.

In this chapter, we investigate the scalability of CCNx, an open source CCN implementation, in terms on the number of nodes (i.e., CCN routers and repositories). Specifically, $2N$ `ccnd` daemons, which correspond to N CCN routers and N repositories, and CCN request generators are not executed on a single physical computer, but separately executed on two physical computers. This enables us to investigate the scalability of CCNx while excluding the measurement noise (i.e., CPU consumption) caused by CCN request generators. Using virtualization technology, a large-scale CCN network with many nodes are constructed in a single physical computer. In our experiments, contents stored in the repositories are requested from different entities for performance benchmarking. As performance metrics, we measure the total throughput of content deliveries, the packet loss ratio in the network, and the average content delivery time. We also examine the performance bottleneck of CCNx through system-wide profiling, which will quantitatively show that per-packet digest-based

authentication at CCN routers is the performance bottleneck in CCNx. We also reveal how the scalability of CCNx in terms of the number of nodes can be improved by hardware offloading of Data-chunk digest computation.

This chapter is organized as follows. Section 2.2 introduces previous works related to scalability of CCN and performance studies using CCNx. Section 2.3 explains the methodology of our experiments such as hardware and software used in experiments, workload generation, performance metrics, and factors to be studied. Section 2.4 presents experiment results for addressing the scalability of CCNx in terms of the number of nodes, including detailed examination of the CCNx performance bottleneck through system-wide profiling. Section 2.5 investigates how the scalability of CCN can be improved with hardware offloading of Data-chunk digest computation at CCN routers. Finally, Section 2.6 concludes this chapter.

2.2 Related Work

In [28], Perino *et al.* quantitatively discuss the scalability of CCN routers to address whether an Internet-scale CCN network can be realized with the latest technologies. Consequently, the authors conclude that using the state-of-the-art technologies for CCN routers, the CCN architecture could scale to the size of the current ISP networks and CDNs (Content Distribution Networks), but not to the size of the current Internet.

In the literature, several simulation studies for investigating the CCN performance have been performed. For instance, in [8], Chiocchetti *et al.* address the scalability of CCN in terms of the number of nodes by comparing simulation times and memory usages when running simulations of different network sizes. The authors show that the scalability of CCN is mostly determined not by the number of nodes in the network but by the number of contents in the network.

Several experimental performance evaluations of CCNx, an open-source software implementation of CCN, have been performed for measuring the end-to-end performance and examining the performance bottleneck in CCN routers [4-6]. In [4], Yuan *et al.* measure the throughput and the resource usage from experiments using CCNx

in a small-scale network. The authors show, to realize a CCN network with 1 [Gbit/s] effective throughput, performance issues of CCNx such as exact string matching in PIT (Pending Interest Table) and CS (Content Store) lookups, and longest-prefix string matching in FIB (Forwarding Information Base) lookup should be solved. On the contrary, in [5], Tang measures the end-to-end performance of CCNx running in a virtualization infrastructure called SAVI (Smart Applications on Virtual Infrastructure). The author shows that the throughput of CCNx is increased by more than 12% by optimizing the function for Data packet decoding called `ccn_skeleton_decode`. In [6], Guimarães *et al.* present a testbed for experiments called FITS (Future Internet Testbed with Security) and measure the performance of CCNx running on the testbed. The authors show that the processing time for packet parsing in CCNx is 19 % larger than that of TCP/IP, and the average content delivery delay (referred to as *download time* in [6]) with CCNx is approximately 25 % smaller than that with TCP/IP.

However, in those studies, the scalability of CCN in terms of the number of nodes has not been addressed. Hence, it has not been understood how the end-to-end performance of CCN (e.g., throughput and content delivery time) is degraded as the number of nodes in the network increases. In this chapter, we therefore experimentally investigate the scalability of CCNx in terms of the network of nodes.

2.3 Experiment Methodology

In this chapter, we investigate the scalability of CCNx in terms of the number of nodes through experiments. In what follows, we explain the methodology used in our experiments.

We used Debian GNU/Linux with 64-bit Linux kernel version 4.4.1 and two Intel-based computers (Core i7 4790 3.60 [GHz] with 8 [Gbyte] memory) connected with a Gigabit Ethernet switch. The process scheduler was CFS (Completely Fair Scheduler), the default scheduler in the Linux kernel. In what follows, one computer used for constructing a virtual CCN network is called S_N server for network, and the other computer used for generating workload is called S_G server for request generators.

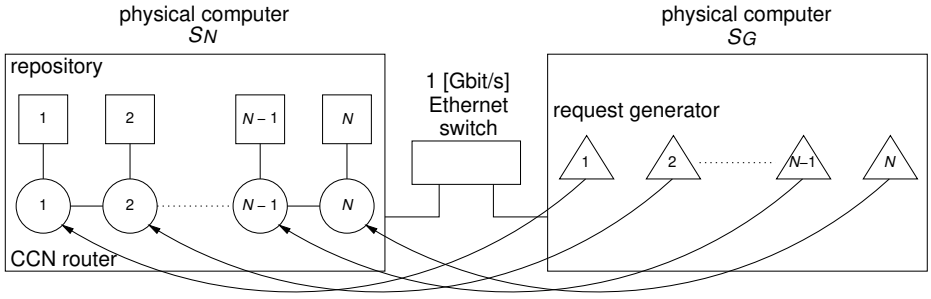


Figure 2.1: Experiment setup: two physical computers, each of which runs either 2 N CCNx daemons connected forming a linear network topology or N request generators.

We used CCNx version 0.8.2 and CCNx version 1.0.

A virtual CCN network comprised of multiple CCN routers and repositories was constructed in a physical computer S_N. $2\,N$ daemons, which correspond to N CCN routers and N repositories were executed. Every daemon is assigned an unique port number. CS (Content Store) sizes of all CCN routers were equally set to 100 [content]. Every repository was provided with 100 unique contents of 1,500 [byte].

We used two types of network topologies: linear network topology shown in Fig. 2.1 and random network topology. In this thesis, we intentionally used linear network topology to directly investigate the effect of the increase in the network size on the performance of CCN.

In a linear network topology, N CCN routers are connected in serial, and N repositories are connected to respective CCN routers. Entries of FIBs in all CCN routers are configured to construct the linear network topology.

In a random network topology, the network topology for CCN routers is generated by a randomized BA (Barabási–Albert) model, which generates *scale-free* networks, for a given network size N and the average degree k. The randomized BA model is an extension of the original BA model [29]; in the randomized BA model, the average number $\overline{m}$ of nodes are newly added at each preferential attachment stage, rather than the constant number m of nodes. We used the average degree $k = 4$ as a typical value of the average degrees of ISP topologies [30]. Similar to a linear network topology, N repositories are connected to respective CCN routers. Entries of FIBs in all CCN routers are configured according to the shortest-path to the nearest

repository.

For workload (i.e., series of Interest packets) generation, we developed a CCN request generator, which randomly sent Interest packets encoded in the CCNB (CCN Binary) format (CCNx version 0.8.2) or TLV (Type-Length-Value) format (CCNx version 1.0) to specified CCN router's face. Request generator $i\,(1 \leq i \leq N)$ repeatedly injects Interest packets to CCN router i requesting a content stored in repository $j\,(\neq i)$. The interval between successive Interest packet generation was given by the exponential distribution with the mean of 1.0 [s]. Request generator i requests 100 contents stored in repository j in a random order.

The clock speed of CPU in physical computer S_N was varied from 0.90 [GHz] to 3.60 [GHz] using cpufreq, the CPU frequency scaling module in the standard Linux kernel. On the other hand, the clock speed of the CPU in physical computer S_G was set to 3.60 [GHz].

As performance metrics, we used followings.

- Throughput

  Throughput is the number of successful content deliveries in the network per a unit time. Note that our throughput definition is for the entire network (i.e., network-level throughput), rather than for every entity (i.e., end-to-end throughput).

- Packet loss ratio

  Packet loss ratio is the rate of packet losses in the network, which is defined as the ratio of the number of unsuccessful content deliveries to the number of total Interest packets injected in the network.

- Average content delivery time

  Average content delivery time is the average time elapsed since an entity sends an Interest packet to the network by the time when the entity receives the corresponding Data packet from the network. Note that the average content delivery time is the average of all *successful* content deliveries, which do not include unsuccessful (e.g., lost or pending) content deliveries.

We obtained those performance metrics from CCN request generators. A single experiment was lasted for 60 [s] and the throughput, the packet loss ratio, and the average content delivery time in the last 40 [s] were calculated to ignore variability in transient state. Also, utilizing /proc file system on the Linux kernel, CPU utilization on the physical computer S_N was calculated while CCN request generators send requests. Experiments were repeated 10 times for a given condition, and the average and its 95% confidence interval for every measurement were calculated.

2.4 Experiment Results

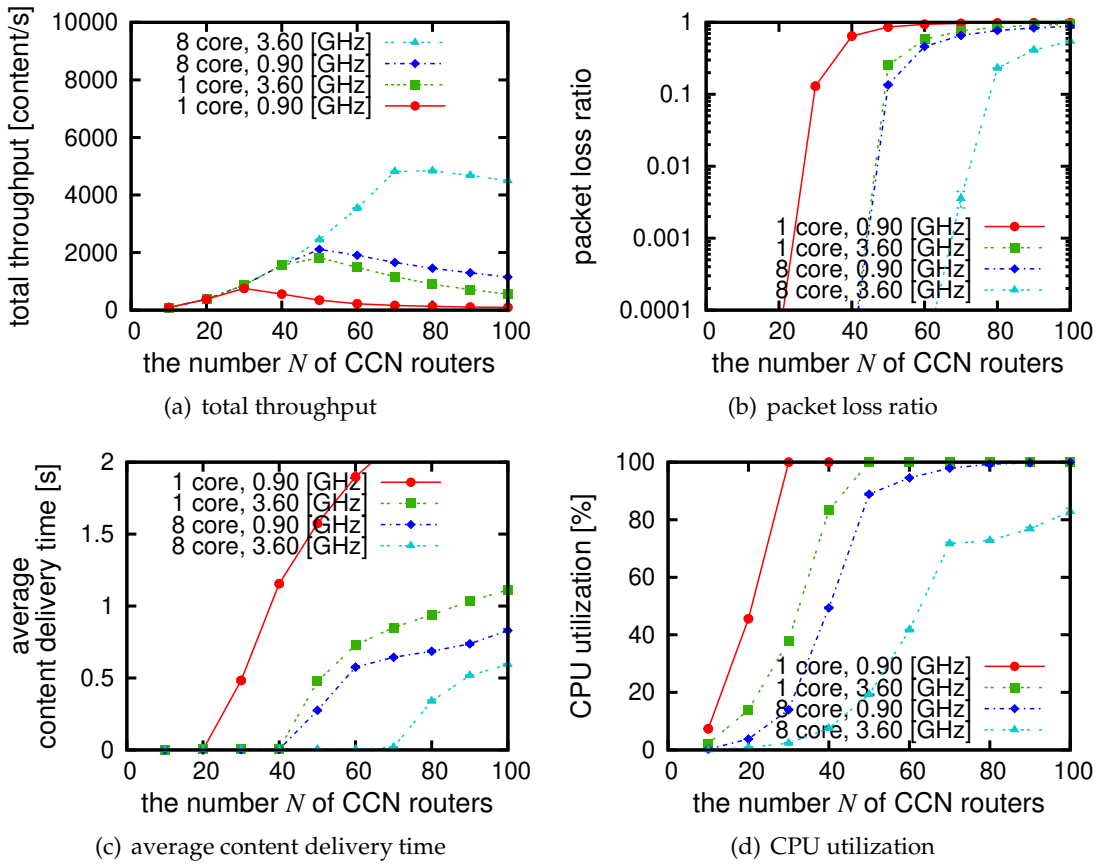
First, experiment results for linear network topologies are shown.

Total throughput, packet loss ratio, the average content delivery time, and CPU utilization when changing the number N of CCN routers are shown in Figs. 2.2 and 2.3.

The total throughput initially increases quadratically as the number N of CCN routers increases since the total number of packets sent from request generators is proportional to $N(N-1)$. The total number of Interest packets and Data packets processed at CCN routers increases exponentially, so that the CPU utilization also increases exponentially.

The larger number of CPU cores and the faster CPU clock speed results in larger throughput, which indicates that CCNx is CPU-intensive rather than I/O-intensive. This observations is confirmed by comparing the total throughput and CPU utilization; i.e., the total throughput decreases as the number N of CCN routers increases when the CPU utilization is close to 100%. In addition, the packet loss ratio and the average content delivery delay rapidly increase as the CPU utilization increases.

CCNx version 0.8.2 and version 1.0 generally show similar tendency. One can find differences in the average content delivery time and the CPU utilization. Namely, the average content delivery times in CCNx version 1.0 varies significantly depending on several factors such as the network size N, the number of CPU cores, and the CPU clock speed. Also, CCNx version 1.0 shows slightly larger CPU utilization than CCNx



(a) total throughput

(b) packet loss ratio

(c) average content delivery time

(d) CPU utilization

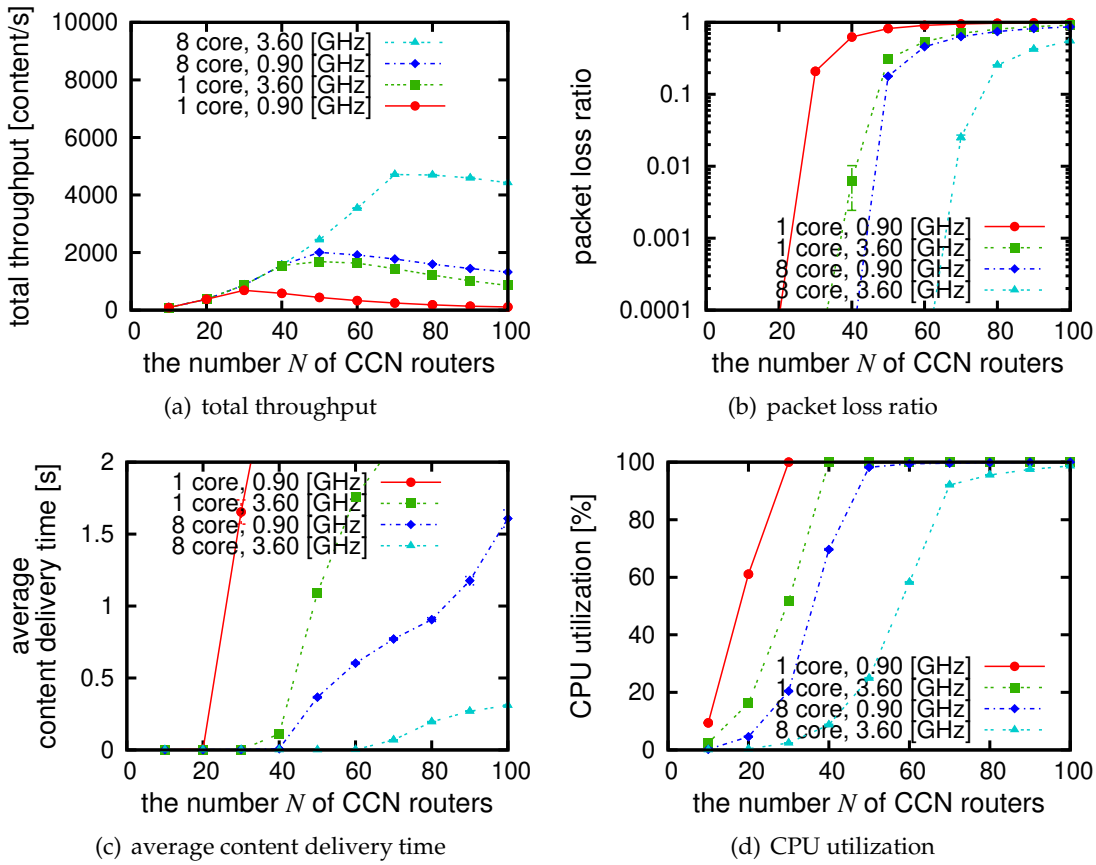
Figure 2.2: Experiment results in linear network topology with CCNx version 0.8.2

version 0.8.2.

Second, experiment results for random network topologies are shown.

Similar to Figs. 2.2 and 2.3, Figs. 2.4 and 2.5 show total throughput, packet loss rate, the average content delivery time, and CPU utilization for different numbers of CCN routers in a random network topology.

The linear network has a larger network distance than that of the random network. For instance, the average number of hops for CCN routers in the linear network topology and the random network with $N = 100$ is 30 and 2.3, respectively. This makes the number of packets processed at CCN routers in the random network topology much less than that in the linear network topology. Consequently, results in the random network topology show higher throughput and smaller average content delivery time than that in the linear network topology. Our observations in the linear network topology also apply to the random network topology.



(a) total throughput

(b) packet loss ratio

(c) average content delivery time

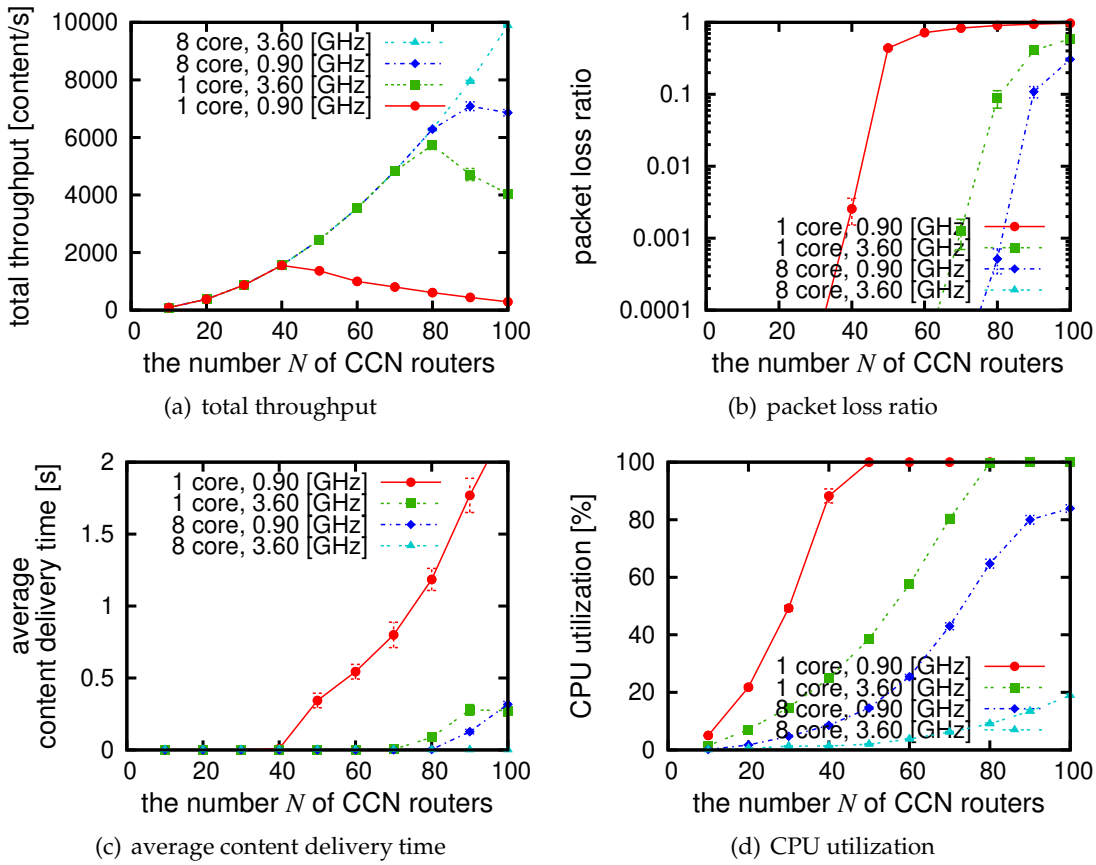
(d) CPU utilization

Figure 2.3: Experiment results in linear network topology with CCNx version 1.0

2.5 Estimating the Impact of Hardware Offloading

To investigate the performance bottleneck of CCNx, we performed a system-wide profiling using OProfile, a system-wide statistical profiling tool for Linux. Tables 2.1 and 2.2 are excerpts of the profiling result, which show ten most time-consuming functions for linear network topology with $N = 30$ and the CPU clock speed of 0.90 [GHz]. These tables indicate that `sha256_block_data_order` function, which is a part of OpenSSL library, utilizes approximately 12–20% of the CPU time. Namely, `sha256_block_data_order` seems to be the performance bottleneck in CCNx. In CCNx, `sha256_block_data_order` function is invoked for every Data packet reception to check the validity of the Data chunk using digest-based authentication.

One possible solution for improving the scalability of CCNx, when a number of



(a) total throughput

(b) packet loss ratio

(c) average content delivery time

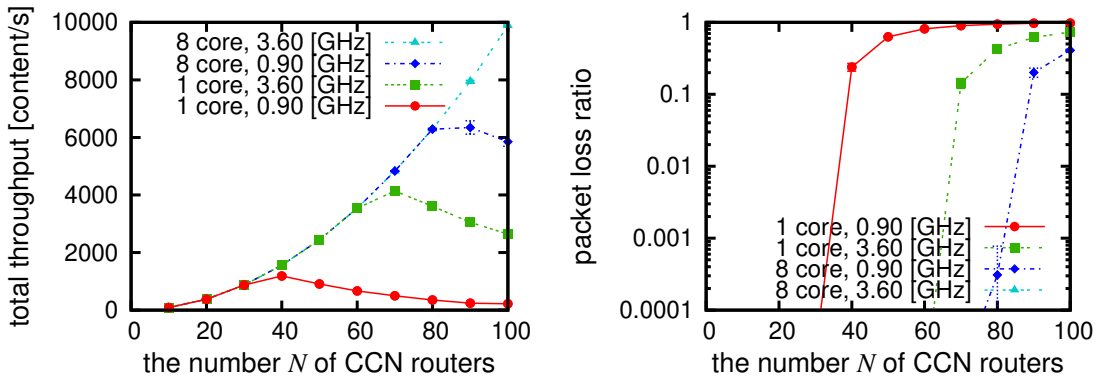
(d) CPU utilization

Figure 2.4: Experiment results in random network topology with CCNx version 0.8.2

CCN router slices are constructed on a single physical computer, is hardware offloading of such CPU-intensive processings.

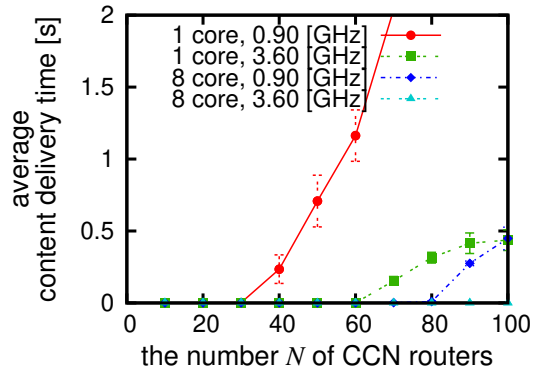
In what follows, we therefore investigate how the scalability of CCNx in terms of the number of nodes can be improved with hardware offloading of the Data-chunk digest computation at CCN routers.

To estimate the performance improvement with hardware offloading, processing of the Data-chunk digest computation at all CCN routers are disabled by bypassing a function invocation. In CCNx version 0.8.2, all invocations of `ccn_diges_create`, `ccn_digest_init`, and `ccn_digest_update` from `ccn_digest_ContentObject` are simply replaced with NOPs (No OPerations). Also, in CCNx version 1.0, `metisTlvSkeleton_ComputeContentObjectHash` from `metisMessage_GetContentObjectHashHash` is replaced with NOPs.

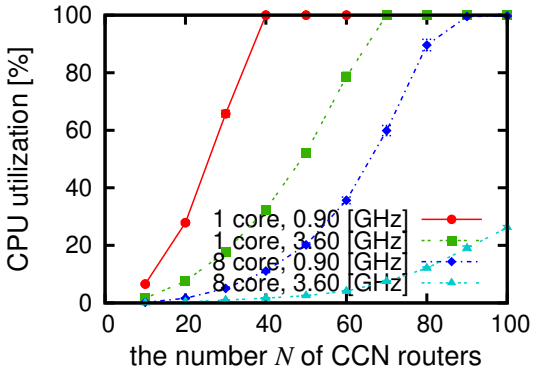


(a) total throughput

(b) packet loss ratio



(c) average content delivery time



(d) CPU utilization

Figure 2.5: Experiment results in random network topology with CCNx version 1.0

Total throughput, packet loss ratio, the average content delivery time, and CPU utilization when changing the number N of CCN routers in a linear network topology with one CPU core and the CPU clock speed of 0.90 [GHz] are shown in Fig. 2.6. In this figure, lines labeled as *offloading* show the results with bypassed Data-chunk digest computation, and ones labeled as *original* the results of the original CCNx.

These results show that, with hardware offloading of the Data-chunk digest computation, improvements in the throughput, the packet loss ratio, and CPU utilization are noticeable (i.e., approximately 30% in case of the throughput). In addition, one can find from these results that the average content delivery time decreases by approximately 60% at maximum with CCNx version 1.0, which should be significant reduction.

Table 2.1: An excerpt of system-wide profiling result with OProfile for $N = 30$ and the CPU clock speed of 0.90 [GHz] (CCNx version 0.8.2)

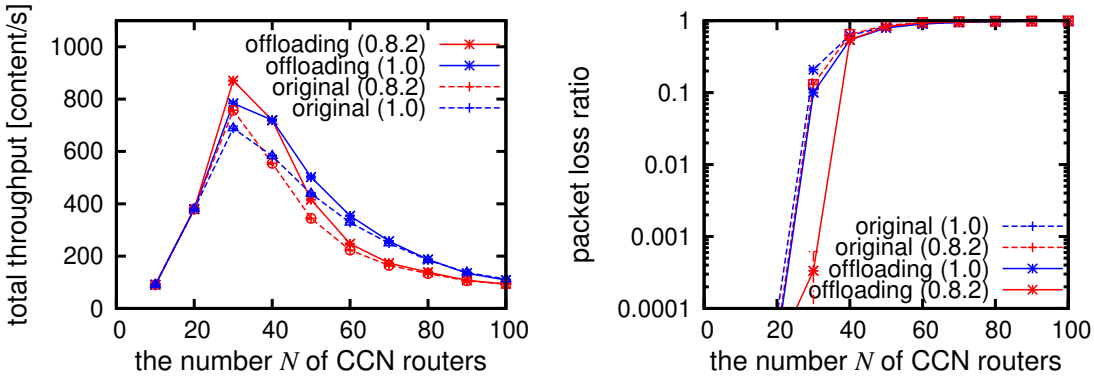
%	image name	app name	symbol name
20.0	libcrypto.so.1.0.0	ccnd	sha256_block_data_order()
7.99	ccnd	ccnd	ccn_skeleton_decode()
5.08	ccnd	ccnd	siphash_2_4()
2.34	ccnd	ccnd	hashtb_seek()
2.00	libc-2.19.so	ccnd	_int_malloc()
1.95	libc-2.19.so	ccnd	__memcmp_sse4_1()
1.48	ccnd	ccnd	ccny_skiplist_findbefore()
1.32	ccnd	ccnd	setpos()
1.14	ccnd	ccnd	hashtb_lookup()
1.12	ccnd	ccnd	heap_sift()

Table 2.2: An excerpt of system-wide profiling result with OProfile for $N = 30$ and the CPU clock speed of 0.90 [GHz] (CCNx version 1.0)

%	image name	app name	symbol name
12.2	libcrypto.so.1.0.0	metis_daemon	sha256_block_data_order()
5.46	libc-2.19.so	metis_daemon	_int_malloc()
3.11	libparc.so.1.0	metis_daemon	_findIndex()
2.68	libc-2.19.so	metis_daemon	_int_free()
2.09	libparc.so.1.0	metis_daemon	parcHashCode_HashImpl()
1.95	libc-2.19.so	metis_daemon	malloc_consolidate()
1.55	libparc.so.1.0	metis_daemon	_parcBuffer_CheckValidity()
1.41	libparc.so.1.0	metis_daemon	_pointerAdd()
1.15	libparc.so.1.0	metis_daemon	_parcStdlibMemory_IncrementOutstandingAllocations()
1.13	libparc.so.1.0	metis_daemon	_parcStdlibMemory_DecrementOutstandingAllocations()

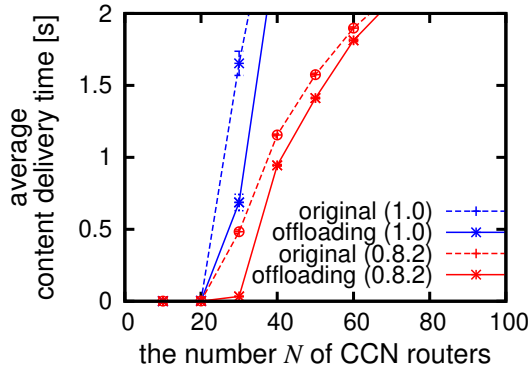
2.6 Summary

In this chapter, we have investigated the scalability of CCNx in terms of the number of nodes. Specifically, multiple CCNx daemons connected forming CCN networks were executed on a single physical computer, and CCN request generators were executed on the other one. As performance metrics, we have measured the total throughput of content deliveries, the packet loss ratio in the network, and the average content delivery time. Consequently, we have shown that, in our experiments, the CCNx performance degrades when the CPU utilization is close to 100%, which indicates that CCNx is CPU-intensive rather than I/O-intensive. Also, we have shown that the Data-chunk digest computation at CCN routers consumed approximately 20% of the total CPU time. In addition, we have revealed the effectiveness of hardware offloading of the Data-chunk digest computation. We found that the hardware offloading of the Data-chunk digest computation is effective of improving the total throughput and reducing the content delivery time.

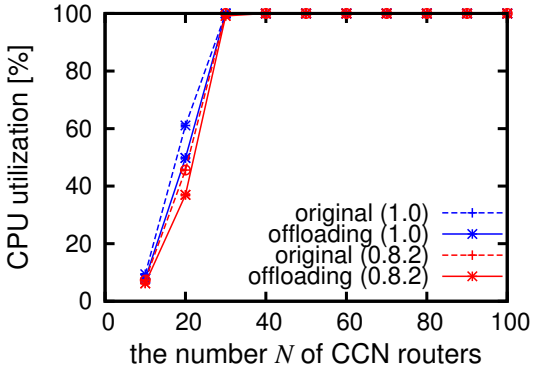


(a) total throughput

(b) packet loss ratio



(c) average content delivery time



(d) CPU utilization

Figure 2.6: Experiment results in linear network topology with/without virtual offloading

Chapter 3

Performance Analysis of Content-Centric Networking on Arbitrary Network Topology

3.1 Introduction

In recent years, Content-Centric Networking (CCN) [1] has been under the spotlight as a networking approach primarily focused on the content transmitted and received (information-centric networks), rather than on the hosts that transmit and receive the content (host-centric networks).

CCN is expected to deliver high availability, since multiple repositories maintain copies of an identical content, while also allowing reduction in traffic volume by caching content relayed by network routers. When using Internet Protocol (IP), one must communicate directly with a host that maintains the desired content in order to obtain the content. CCN, in contrast, does not require identification of the host that maintains the content, and the content can be obtained from anywhere so long

This chapter is a minor version of [31].

as it exists in the network. As a consequence, CCN is expected to reduce content delivery delays, reduce traffic volume transferred through the network, and improve availability as a result of the ability to disseminate content copies within the network in a natural way.

Many studies have investigated the effectiveness of CCN through simulation experiments. For example, in [7], the effectiveness of CCN in video streaming is investigated through simulation. The results of that study show that the performance of CCN is not greatly affected by topology, and that the effectiveness of CCN depends largely on the distribution of the content requested by users. In [32], power consumption for video streaming is examined when either IP or CCN is used. The results show that CCN can reduce overall power consumption by reducing traffic volume, which results from caching, although power consumption in network devices increases as a result of maintaining a high volume of cache data.

There have also been mathematical analyses of CCN [9, 11-15]. For example, the authors of [11] have developed a Markov model for a cache in single-router CCN, and have examined the performance of CCN in a tree topology by complementing their Markov model with simulation results. Caching performance for single-router CCN with Interest packet aggregation was analyzed in [12]. The authors of [9] calculated CCN throughput and content delivery delay on cascade-type and binary tree-type network topologies. As a result, it was shown that CCN throughput and content delivery delay depend on cache size, content size and content popularity distribution. In addition, the authors of [13] calculated the content delivery delay, throughput and download time in the case where an entity performed multipath access from multiple leaf routers in a tree network, by extending the analysis in [9]. However, these analytical studies were limited to simple network topologies, and the effectiveness of CCN in a more general network topology has yet to be ascertained.

The pioneering work in performance analysis of a multi-cache network is [33], which proposed the multi-cache approximation (MCA) algorithm for analytically calculating cache hit probability for intermediate nodes. The authors of [33] mostly focus on link-level performance (i.e., cache hit probability) instead of on network-level per-

formance (e.g., delivery delay, throughput, and availability).

In [14, 15], the performance of CCN in a general network topology was analyzed. The author of [15] developed a modeling framework for a general network based on Information-Centric Networking (ICN), and obtained several performance metrics. However, in this study, application-level performance metrics (e.g., content delivery delay and throughput) were not analyzed. In this study, cache hit probability at routers and content delivery delay (referred to as virtual round-trip time in [14]) in a general network topology were obtained by MCA [33]. We analyze CCN performance using the MCA algorithm in a similar manner to [14], but we also analytically calculate throughput and availability, assuming link failures in addition to content delivery delay.

In this chapter, we analytically derive content delivery delay, throughput, and availability with CCN for an arbitrary network topology. We model CCN with multiple routers and multiple repositories. Performance when multiple entities request a content stored in repositories is analyzed. Time required until an entity obtains the content after making a request (content delivery delay), throughput for content acquisition, and probability for an entity to successfully obtain the content under probabilistic link failures are also analytically calculated. Furthermore, the effect of network topology on the effectiveness of CCN is also studied through several numerical examples.

The contributions of this chapter are summarized as follows.

- We propose a modeling approach for large-scale CCN on arbitrary network topology.
- We analytically derive content delivery delay, availability, and throughput in CCN by using MCA algorithm.
- We show the effect of network topology on the performance of CCN.

This chapter is organized as follows. First, Section 3.2 describes the analytical model used in this chapter. Section 3.3 analyzes content delivery delay, throughput, and availability in CCN with an arbitrary network topology. Section 3.4 looks into

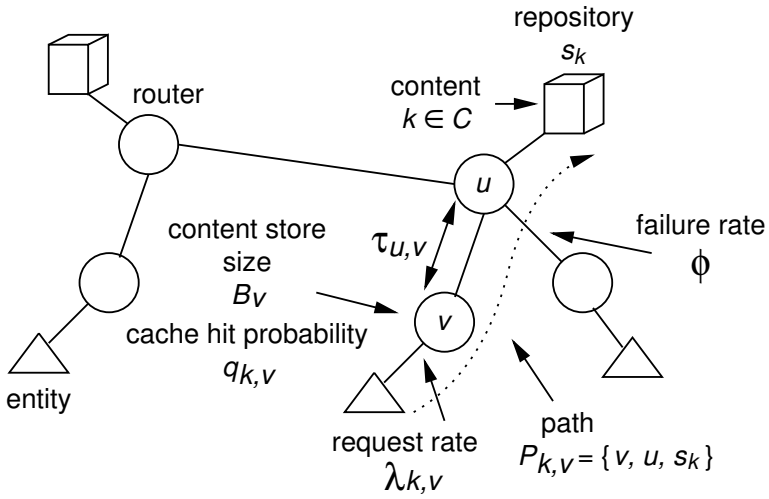


Figure 3.1: Analytic model

the effects of network topology on the effectiveness of CCN using several numerical examples. Section 3.5 examines the validity of our estimates by comparing our analytical results with those by simulation. Finally, Section 3.6 provides a summary of this chapter.

3.2 Analytic Model

The topology for a CCN network comprising multiple routers (CCN routers) and multiple repositories is expressed as an undirected graph $G = (V, E)$ (Fig. 3.1). Hereinafter, these routers and repositories are referred to as *nodes*.

C represents a collection of all contents present in the network. For simplicity, it is supposed that all contents are of the same size. The space needed to store content in router v is expressed as B_v, and the communication delay for the link between node u and node v (i.e., propagation delay plus all processing delays) is $\tau_{u,v}$. The failure rate for links is set to ϕ for all links.

It is supposed that each content exists in a single repository, and that the Forwarding Information Base for each router is properly set up by the routing.

The path from node $v \in V$ to the repository that stores content $k \in C$ is written $P_k^v = (v, \ldots, s_k)$. Here, s_k indicates the repository that stores content k. The nth node in the path P_k^v is notated $P_k^v[n]$. Therefore, $P_k^v[1] = v$ and $P_k^v[|P_k^v|] = s_k$.

It is assumed that each entity is connected directly to a router, that the bandwidth between an entity and a router is sufficient, and that entity–router communication delays are negligible.

The arrival rate of Interest packets for content k received by node v from directly connected subordinate entities is expressed as $\lambda_{k,v}$. In addition, the cache hit probability for content k at node v is expressed as $q_{k,v}$. For a repository s_k that stores content k, we define $q_{k,s_k} = 1$.

3.3 Analysis

3.3.1 Content Delivery Delay

First, the content delivery delay (i.e., the expected time difference between request and receipt) in CCN when there is no link failure (i.e., when $\phi = 0$) is obtained.

Since the router caches content within the network in CCN, content delivery delay is reduced, and traffic volume is likely to be reduced. Since each content is distinguished by a unique identifier in CCN, content is recycled at the network level. A router returns a Data packet without further relaying a corresponding Interest packet if it has the Data packet corresponding to the Interest packet in its Content Store.

When an entity sends an Interest packet to the network, the router forwards the Interest packet to the nearest repository, determined according to a routing table called the Forwarding Information Base. If the content corresponding to the Interest packet is cached in a router along the path, the router returns the corresponding content to the entity as a Data packet rather than forwarding the Interest packet. If the content corresponding to the Interest packet is not cached in any of the routers along the path, then the Interest packet arrives at the repository, which returns the requested content as a Data packet.

The cache hit probability $q_{k,v}$ for content k at router v can be approximately obtained using MCA [33] or multi-cache with aggregation approximation (MCAA) [14] for partial networks comprising only routers.

MCA is an approximation algorithm to analytically calculate the cache hit proba-

bility in a multi-cache network [33]. MCA uses single-cache approximation (SCA) [34] to calculate the cache hit probability at a single node that has finite buffer size uses either first-in first-out (FIFO) or least-recently used (LRU) for cache expiration. It repeatedly applies SCA to each node in the network, and calculates the cache hit probability at each node.

MCA calculates $m_{k,v}$ so that it satisfies the following equations by iterative calculation [33].

$$r_{k,v} = \lambda_{k,v} + \sum_{v':k \in R(v',v)} m_{k,v'} \tag{3.1}$$

$$p_{k,v} = \frac{r_{k,v}}{\sum_{i=1}^{|C|} r_{i,v}} \tag{3.2}$$

$$\vec{q_v} = contents(\vec{p_v}, B_v) \tag{3.3}$$

$$m_{k,v} = r_{k,v}\,(1 - q_{k,v}) \tag{3.4}$$

Here, $m_{k,v}$ indicates the rate of misses (i.e., the number of misses per unit of time) at node v for content k. In this, $r_{k,v}$ indicates the request rate (i.e., the total request rate flowing in from upstream nodes and the request rate received from entities directly connected to the node) for content k at node v; $R(v, v')$ is the collection of content for which node v is the next hop from node v' on the path. For example, if $R(v, v') = \{k\}$, then node v is located at the next hop in the path for content k from node v'. Further, $p_{k,v}$ and $q_{k,v}$ are the relative request rate and the cache hit probability, respectively, for content k at node v. The vectors $\vec{p_v}$ and $\vec{q_v}$ aggregate $p_{k,v}$ and $q_{k,v}$, respectively, for all k (i.e., all contents).

In [14], the MCAA algorithm, which extends the MCA algorithm and models the aggregation of Interest packets at a router, was proposed.

By utilizing either the MCA or the MCAA algorithm, the cache hit probability $q_{k,v}$ for content k at router v can be calculated.

The probability for the Data packet corresponding to an Interest packet sent from node v for content k to be returned by the nth node on the path P_k^v (i.e., hitting the cache at the nth node, or the nth node being the repository) is expressed as $\eta_{k,n}^v$. Since

content k is always returned from one of the nodes on the path P_k^v, $\sum_n \eta_{k,n}^v = 1$ when link failure is not included.

Since the cache hit probability at the nth node is $q_{k,P_k^v[n]}$, the value of $\eta_{k,n}^v$ is given by

$$\eta_{k,n}^v = q_{k,P_k^v[n]} \prod_{i=1}^{n-1} (1 - q_{k,P_k^v[i]}). \tag{3.5}$$

Therefore, the expected delay for content delivery given an Interest packet received by node v for content k is

$$D_k^v = \sum_{n=2}^{|P_k^v|} \left(\eta_{k,n}^v \sum_{m=1}^{n-1} 2\,\tau_{P_k^v[m],P_k^v[m+1]} \right). \tag{3.6}$$

Since the arrival rate of Interest packets received by node v from directly connected entities for content k is $\lambda_{k,v}$, the expected content delivery delay, D^v, at node v for any content is given by

$$D^v = \sum_{k \in C} \frac{\lambda_{k,v}}{\sum_{k' \in C} \lambda_{k',v}} D_k^v. \tag{3.7}$$

3.3.2 Throughput

Next, the throughput for content retrieval in CCN when link failure does not occur (i.e., $\phi = 0$) is obtained.

In general, Interest packets are very small relative to Data packets. We therefore neglect Interest packets in calculating traffic. The size of each Data packet is assumed to be S.

The rate at which the Data packet is returned by node v for content k is expressed as $x_{k,v}$. The rate at which node v receives Interest packets for content k is $r_{k,v}$, and the cache hit probability is $q_{k,v}$, so we have

$$x_{k,v} = r_{k,v}\, q_{k,v}\, S + \sum_{v' : k \in R(v',v)} \xi_{k,v}(v'), \tag{3.8}$$

where $\xi_{k,v}(v')$ indicates the reception rate for Data packets for content k flowing from node v' to node v.

The rate of Data packet transmission for content k from v' to v is expressed as $\zeta_{k,v'}(v)$. Note that $\zeta_{k,v'}(v)$ is the rate of transmission from v' to v, and $\xi_{k,v}(v')$ is the rate of reception at node v from node v'.

If the bandwidth between node v and node v' is $\mu_{v,v'}$, then Data packets exceeding bandwidth $\mu_{v,v'}$ are discarded at transmission time from node v'. Therefore, $\xi_{k,v}(v') \leq \zeta_{k,v'}(v)$.

The rate at which node v' returns Data packets for content k is $x_{k,v'}$, and only a fraction $m_{k,v}/r_{k,v'}$ of that is transmitted to node v. So, we have

$$\zeta_{k,v'}(v) = \begin{cases} x_{k,v'} \frac{m_{k,v}}{r_{k,v'}} & \text{if } k \in R(v,v') \\ 0 & \text{otherwise} \end{cases} . \tag{3.9}$$

Assuming that fair queuing is enforced by all routers and that the rate of loss for Data packets is proportional to the transmission rate for Interest packets, $\xi_{k,v}(v')$ is given by

$$\xi_{k,v}(v') = \min(\mu_{v,v'}, \sum_{i \in C} \zeta_{i,v'}(v)) \frac{\zeta_{k,v'}(v)}{\sum_{i \in C} \zeta_{i,v'}(v)}. \tag{3.10}$$

Therefore, the rate of transmission for Data packets for content k returned by node v to entities connected directly to the node T_k^v (i.e., the throughput) is given by

$$T_k^v = \frac{\lambda_{k,v}}{r_{k,v}} x_{k,v}. \tag{3.11}$$

The total throughput T^v for Data packets for node v to return to entities connected directly to the node is the sum of throughputs for all content.

$$T^v = \sum_{k \in C} T_k^v \tag{3.12}$$

3.3.3 Availability

Finally, content availability (i.e., the probability that an entity can successfully obtain a requested content) with CCN is derived.

Since the routers cache content in CCN, it is possible to obtain a content so long as all links to the router caching the content are functioning properly, even when other links in the network temporarily fail.

The availability of content k at node v is expressed as A_k^v. That is, A_k^v is the probability that the Data packet corresponding to an Interest packet can be obtained properly when the Interest packet is sent from node v to request content k.

If a cache is hit at the nth node on the path P_k^v from node v to repository s_k (which maintains content k), then the content can be properly obtained if the path from the 2nd to nth node on the path is properly functioning. Since the cache hit probability at the nth node is $q_{k,P_k^v[n]}$ and the failure rate for each link is ϕ , we have

$$A_k^v = \sum_{n=1}^{|P_k^v|} \left(\eta_{k,n}^v \left(1-\phi\right)^{2\,(n-1)} \right). \tag{3.13}$$

Because the rate of arrival for Interest packets received by node v from entities directly connected to the node for content k is $\lambda_{k,v}$, the availability A^v for all content at node v is obtained as

$$A^v = \sum_{k \in C} \frac{\lambda_{k,v}}{\sum_{k' \in C} \lambda_{k',v}} A_k^v. \tag{3.14}$$

3.4 Numerical Examples

First, content delivery delay for content k at router v (Eq. (3.6)) in a linear network topology where five routers and one repository are connected in series (see Fig. 3.2) is shown in Fig. 3.3. The repository (node 6) stores 500 items of content $C = \{1, \ldots, 500\}$. The arrival rate of Interest packets for content k at router v $(1 \leq v \leq 5)$ from directly connected entities, $\lambda_{k,v}$, is given by a Zipf distribution with a mean of 20 [requests/s] and an exponent parameter of 1.0. Therefore, the arrival rate of Interest packets for

content at a specific router is heavy-tailed. For instance, content 1 is the least popular content, and content 500 is the most popular content. Further suppose that the content store size B_v is set equally to 50 [content] in all routers, and communication delay $\tau_{u,v}$ is equally 1 [ms] for all links. Link failure rates at all links are set to $\phi = 0$ unless stated otherwise. The packet size S for Data packets is 8 [Kbytes], and the bandwidth $\mu_{v,v'}$ between nodes (i.e., routers and repositories) is set to 100 [Mbits/s]. The bandwidth between an entity and its neighboring router is unlimited; that is, links at network edges never become a performance bottleneck.

One can see from Fig. 3.3 that delivery delay becomes lower for more frequently accessed content (i.e., for larger k). It can also be seen that the delivery delay is longer for routers further from the repository (i.e., for smaller v). There are five hops from the router on the left end (node 1) to the repository (node 6), and the delivery delay when there is no content caching (when it is directly obtained from the repository) is $1 \times 5 \times 2 = 10$ [ms]. From Fig. 3.3, it is seen that the content delivery delay is about 9 [ms] at maximum for $k = 1$, and nearly zero at minimum for $k = 500$ as content caching is done.

Second, the throughput for content k at router v, T_k^v, is shown in Fig. 3.4. Note that the y-axis is plotted on a logarithmic scale. This figure shows that throughput significantly differs for every content since the arrival rate of Interest packets is given by a Zipf distribution in our numerical examples. The throughput for popular content (i.e., large k) exceeds 10 [Mbits/s], but that for unpopular content is less than 0.1 [Mbits/s]. One can see from this figure that, unlike with content delivery delay in Fig. 3.3, routers far from the repository (i.e., smaller v) achieve *higher* throughput than those close to the repository (i.e., larger v). This phenomenon can be explained by the *filtering effect* in multi-stage caching. Namely, in multi-stage caching, popular content is likely to be hit at an earlier stage. Hence, popular content is less likely to be accessed at a later stage. The link bandwidth at the later stage competes with requests for different content. However, because of the filtering effect, popular content is less likely to be accessed, so that requests for unpopular content are likely to achieve higher throughput.

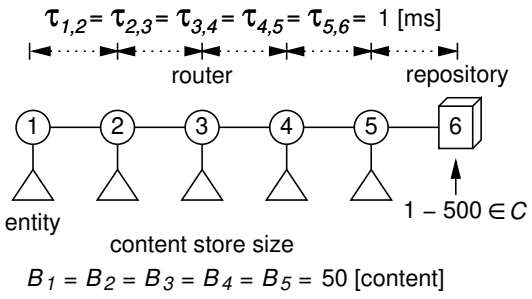


Figure 3.2: Linear network topology: five routers and one repository are connected in series.

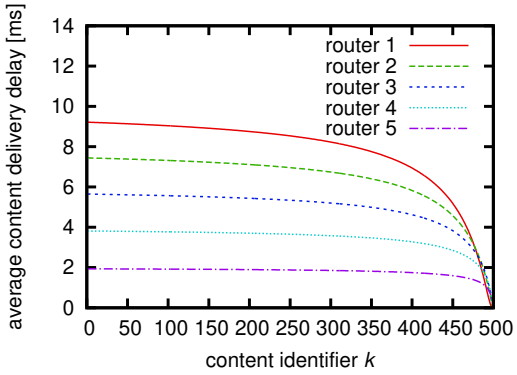


Figure 3.3: Content delivery delay in linear network topology

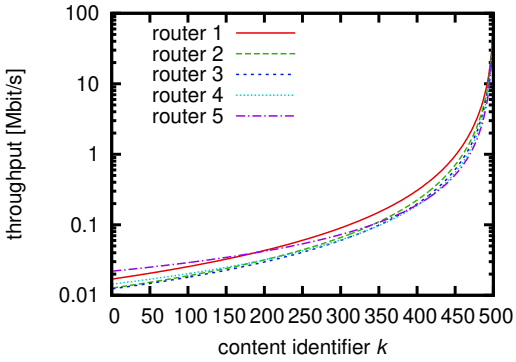


Figure 3.4: Throughput in linear network topology

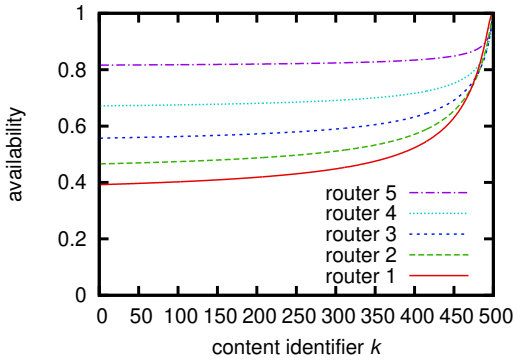


Figure 3.5: Availability in linear network topology

The availability of content k at router v (Eq. (3.13)) when the link failure rate is set to $\phi = 0.1$ is shown in Fig. 3.5. This figure shows that content availability improves dramatically with content caching. Since the link failure rate between router 5 and the repository (node 6) is also ϕ, the availability for router 5 is $(1-\phi)^2 = 0.81$. As is shown in Fig. 3.5, the availability exceeds 0.5 except for content with low popularity, even for routers that are far from the repository (such as routers 1 and 2). It is possible to obtain a content in CCN if one of the routers on the path has cached the content (and all links to the router are functioning properly).

Next, content delivery delay for content k at router v in the simple network topology shown in Fig. 3.6, where five routers and two repositories are connected, is shown in Fig. 3.7. Three entities are connected, one to each of router 1, router 2, and router 3. Similar to Fig. 3.3, there are 500 contents $C = \{1, \ldots, 500\}$ in the network. One

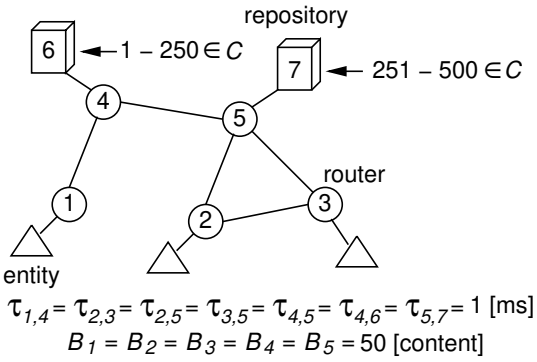


Figure 3.6: Simple network topology: five routers and two repositories are connected, with each of the two repositories keeping 250 contents.

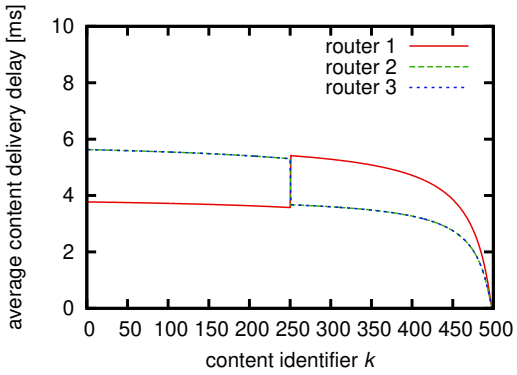


Figure 3.7: Content delivery delay in simple network topology

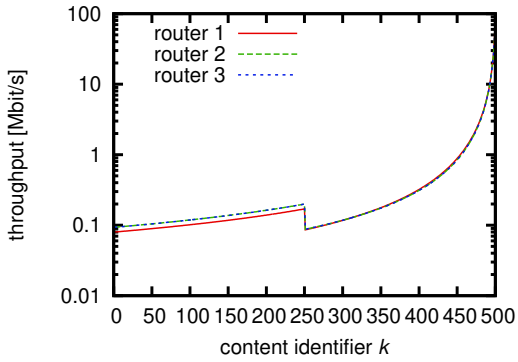


Figure 3.8: Throughput in simple network topology

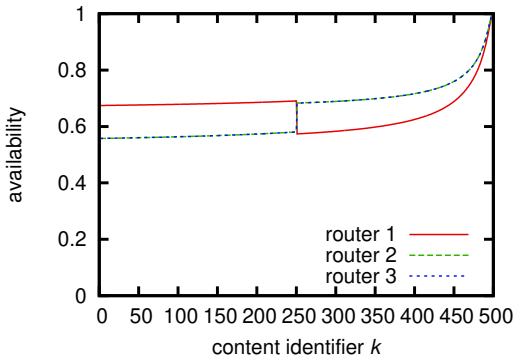


Figure 3.9: Availability in simple network topology

repository (node 6) has contents $1, \ldots, 250$, and the other (node 7) has the remaining content. The other conditions are the same as in Fig. 3.3. Hereafter, the network topology shown in Fig. 3.6 is called a *simple network topology*. Note that in Fig. 3.7, content delivery delays at router 2 and router 3 are indistinguishable.

Figure 3.7 shows that the content delivery delay is smaller when the requesting router is closer to the repository holding the content. The small delivery delay is caused by higher cache hit probability at routers near the repository, as well as a lower number of hops from the requesting router to the corresponding repository.

Throughput for content k at router v, T_v^k, in the simple network topology is shown in Fig. 3.8. Again, this figure shows that throughput significantly differs for every content. Namely, throughput for popular content is quite high, and throughput for

unpopular content is very low. However, the difference is caused by a difference in Zipf-distributed request rates. A notable difference from Fig. 3.4 is that throughputs at routers 1, 2, and 3 are almost the same in Fig. 3.8, even for unpopular content. This phenomenon can also be explained by absence of the filtering effect in multi-stage caching. The number of hops in the simple network topology is either 2 or 3, and therefore, any filtering effect is unlikely to be strong. Thus, unpopular content is not likely to benefit from higher throughput caused by the filtering effect.

Availability of content k at router v in the simple network topology is shown in Fig. 3.9. The link failure rate is set to $\phi = 0.1$, similar to Fig. 3.5. This figure shows that the availability for the corresponding content is higher when the router is closer to the repository that has the content, just as shown in the results for content delivery delay.

From these observations, we conclude that the benefit of performance improvement from content caching in terms of delivery delay and availability is higher for entities *closer* to the repository. In contrast, the benefit in terms of throughput is the opposite: entities *further* from the repository see higher throughput.

Finally, to demonstrate the usability of our analysis, we examine the performance of CCN with a real network topology, the Abilene network topology [35], which is shown in Fig. 3.10. For demonstration purposes, a single repository (node 12) with 500 contents is connected to router 5. A single entity is connected to all routers. Other conditions are the same as those with the linear network topology and the simple network topology. We note that our analysis places no limitation on the number of repositories in the network and does not assume heterogeneity in arrival rates of Interest packets at routers. We used a simple scenario because a complicated scenario makes interpretation of numerical examples difficult.

Content delivery delay, throughput, and availability for content k at router v are shown in Figs. 3.11, 3.12, and 3.13, respectively. Because of space limitations, these figures show results for only routers 1, 5, 7, and 11. The results for routers 3 and 9 are almost the same in those for router 1. In these figures, we can see similar tendencies to those observed for the linear network topology and the simple network topology.

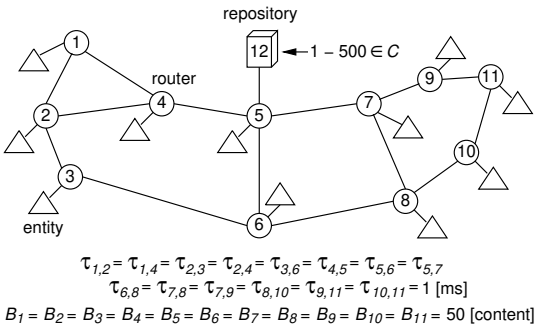


Figure 3.10: Abilene network topology

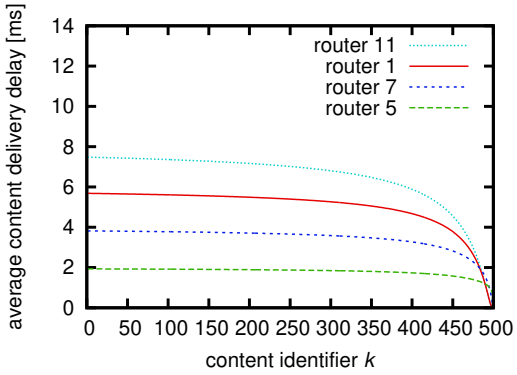


Figure 3.11: Content delivery delay in abilene network topology

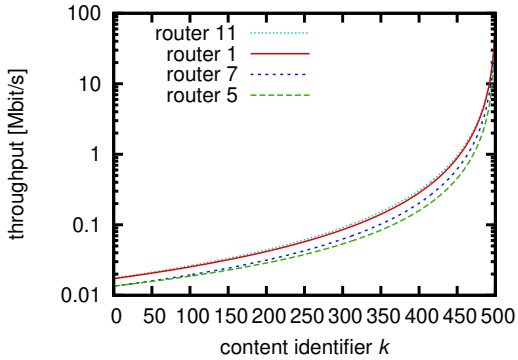


Figure 3.12: Throughput in abilene network topology

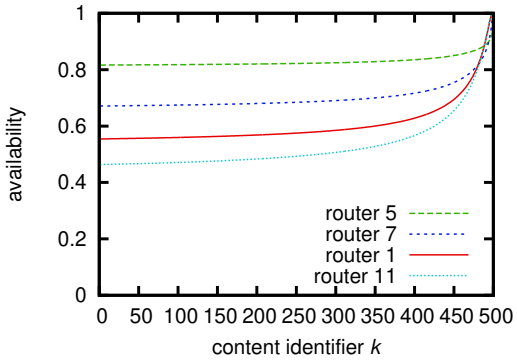


Figure 3.13: Availability in abilene network topology

However, because of complexity in the network topology, Figs. 3.11, 3.12, and 3.13 exhibit more complex patterns, which implies that the performance of CCN depends strongly on the network topology and that performance analysis should explicitly take account of the network topology to be studied.

3.5 Validation

Finally, the validity of our estimation is examined by comparing analytical results with those by simulation.

We have developed a chunk-level CCN simulator, written in the Perl programming language, and measured content delivery delay, throughput, and availability in three network topologies shown in Fig. 3.2, Fig. 3.6, and Fig. 3.10. The parameters

are the same as those used in Section 3.4. Interest packets were randomly generated from entities at a specified rate, $\lambda_{k,v}$. The queueing discipline at all routers was FIFO. The cache replacement algorithm at all routers was LRU. Every simulation run lasted for 30 [s]. For a single parameter setting, simulations were repeated 10 times, and the average and 95% confidence interval of all measurements were obtained. For better readability, 95% confidence intervals are shown sparsely (i.e., for every 50 contents) in the following figures.

Simulation results for content delivery delay, throughput, and availability are shown in Figs. 3.14 through 3.16. These figures show good agreement between analytical and simulation results in content delivery delay, throughput, and availability, which clearly shows the validity of our analysis, even in a cascaded network topology.

Next, simulation results for content delivery delay, throughput, and availability in a simple network topology and the Abilene network topology are shown in Figs. 3.17 through 3.19 and Figs. 3.20 through 3.22, respectively. These figures show that analytic results and simulation results agree even in general networks.

3.6 Summary

In this chapter, we have analyzed the performance of CCN on an arbitrary network topology by utilizing the MCA algorithm, which is an approximation algorithm which analytically calculates cache hit probability in a multi-cache network. Content delivery delay, throughput, and availability in a network comprising multiple routers and multiple repositories have been analytically calculated. Through several numerical examples, we have shown that the benefits of performance improvement by content caching (i.e., reduction in content delivery delay and improvement in availability) were more pronounced when the router was closer to the repository in CCN. We have also shown the validity of our analysis through simulation experiments.

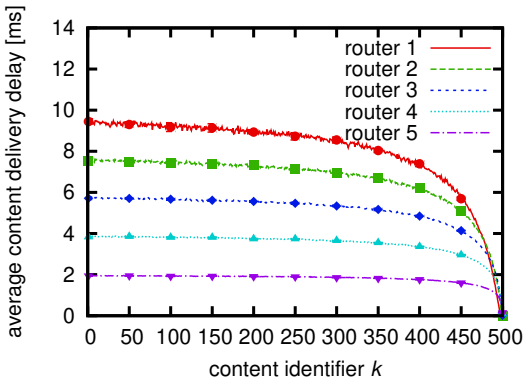


Figure 3.14: Simulation results of content delivery delay for content k at router v in linear network topology

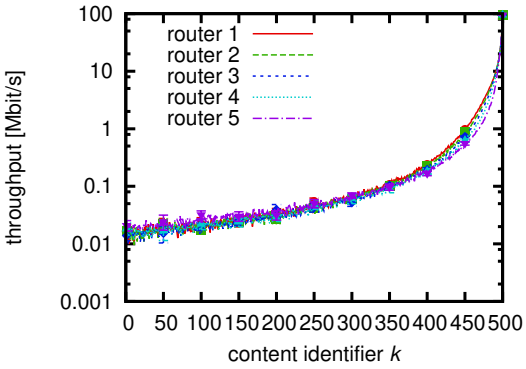


Figure 3.15: Simulation results of throughput for content k at router v in linear network topology

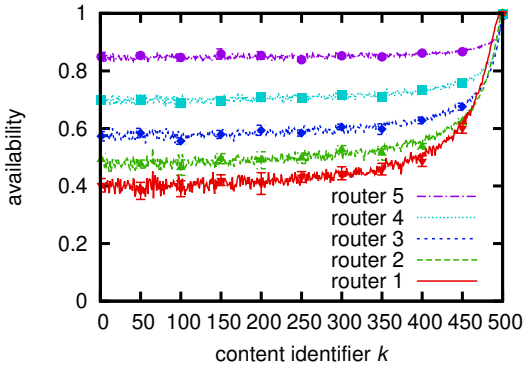


Figure 3.16: Simulation results of availability for content k at router v in linear network topology

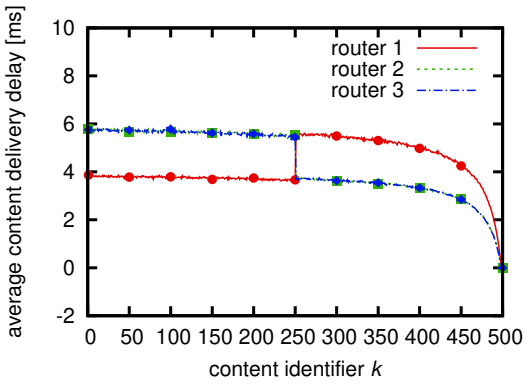


Figure 3.17: Simulation results of content delivery delay for content k at router v in simple network topology

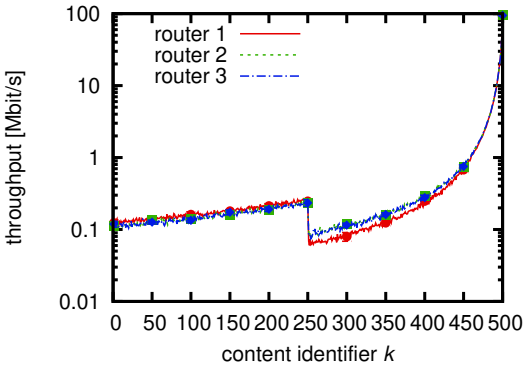


Figure 3.18: Simulation results of throughput for content k at router v in simple network topology

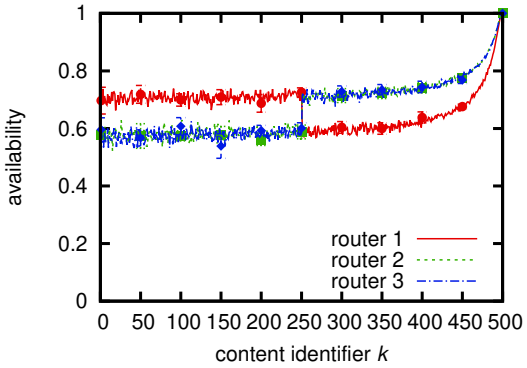


Figure 3.19: Simulation results of availability for content k at router v in simple network topology

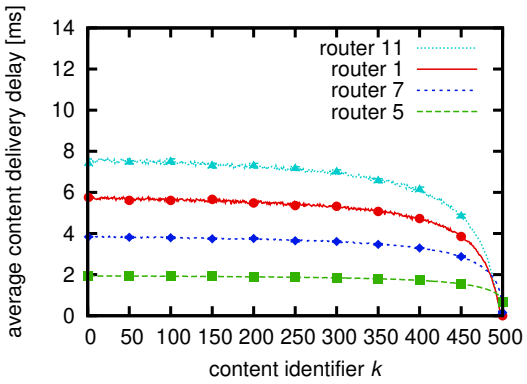


Figure 3.20: Simulation results of content delivery delay for content k at router v in abilene network topology

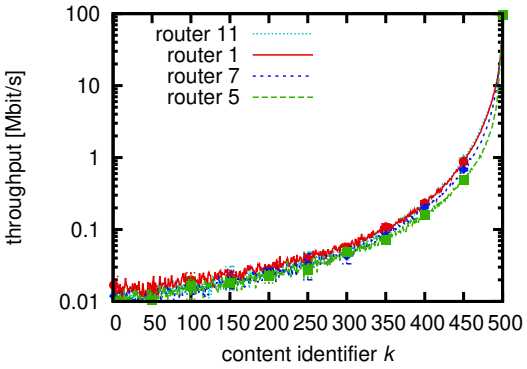


Figure 3.21: Simulation results of throughput for content k at router v in abilene network topology

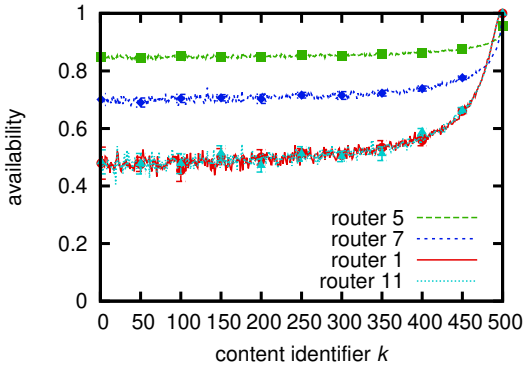


Figure 3.22: Simulation results of availability for content k at router v in abilene network topology

Chapter 4

Performance Analysis of Large-Scale Information-Centric Networking

4.1 Introduction

In the recent years, Information-Centric Networking (ICN) that mainly focuses on contents that are transmitted and received instead on end hosts that transmit and receive contents has been under the spotlight [1-3].

Notable features of ICN architectures compared with the conventional TCP/IP network are adoption of unique content identifiers, location independence, in-network content caching, native support of anycast, multicast and broadcast communications [3]. In an ICN, *contents* are stored in one or more *content providers*. The primary objective of ICNs are efficient content delivery from content providers to content consumers called *entities*. A requesting entity injects a *content request* into the ICN, which tries to deliver the content request to nearby content provider(s) through *routers*. The content

This chapter is a minor version of [36].

is sent back from the content provider to the requesting entity. Because of in-network caching, the content might be directly sent back from one of *caching routers*.

In the literature, several ICN architectures have been proposed, each of which has commonalities and differences with others. Two of the most popular ICN architectures are CCN (Content-Centric Networking) [1] and NDN (Named Data Networking) [2]. In CCN and NDN, routers in a network can cache the forwarded contents to its buffer memory (called *Content Store* in [1, 2]), and can return requested contents from the buffer memory. As a consequence, CCN and NDN are expected to reduce content delivery delays from a repository (i.e., content provider) to entities and traffic volume transferred through the network.

Because of in-network content caching, ICN is a sort of *cache networks* where multiple content caches are mutually connected residing in between a content provider and a content consumer. One of the most popular and widely-deployed cache networks is a hierarchical Web caching [37] where Web proxy servers with page caches are hierarchically connected to serve Web page requests from many clients, as well as CDN (Content Delivery Network). A node in a cache network (e.g., a proxy server in Web and a content router in ICNs) is provided with a cache storage, which stores copies of content recently served from the node. Those nodes with caches are logically interconnected. If a content request (e.g., an HTTP GET request in Web and a content request in ICN) *hits* the cache of a node (cache hit), it immediately returns the content to the requesting entity. Otherwise (cache miss), the node forwards the content request to upstream node(s). If the content request never experiences cache hit at all nodes along the path, the content request is eventually forwarded to the content provider (e.g., an HTTP server in Web and a content provider in ICN).

Both ICN and hierarchical Web caching have high commonality (e.g., receiver-driven network and hierarchical content caching), but the essential difference is in their *scales* — the targeted scale of ICN is significantly larger than that of hierarchical Web caching. For instance, in CCN and NDN, nodes of the cache network are *routers* whereas nodes are *application gateways* in hierarchical Web caching. The Web is an extremely huge network and a vast number of Web proxy servers have been operating

on the current Web. However, those Web proxy servers are mostly independent, so that a single hierarchical Web cache network is rather small.

To realize a global-scale ICN, it is crucial to understand the fundamental properties of such large-scale cache networks. To the best of our knowledge, it is still an open issue how scalable ICNs are in terms of the network size (e.g., the number of entities, routers, and content providers), the number of contents in the network, the cache size of routers, and the processing speed of routers. Among those scaling factors, scalability of ICN regarding the network size should particularly important since impacts of increasing other three factors (i.e., contents diversity, router cache size, processing speed) are expectable. For instance, increase in cache sizes and processing speeds of routers should have positive effect. However, it is unclear whether increase in the size of an ICN is beneficial or malicious.

The scaling property of ICN has not been well understood due to the lack of theoretical foundations and analysis methodologies. In the literature, the characteristics of *small-scale* cache networks have been investigated [38, 39]. One interesting phenomenon observed in a cache network is a *filter effect* — in multi-stage caching, a content with high popularity (i.e., frequently-requested content) is likely to be hit at earlier stage so that the content popularity distribution at later stage tends to be smoothed. However, to the best of our knowledge, the implication of filter effect on a large-scale ICN has not been well understood. Due to the complexity of a large-scale ICN, it is not trivial to investigate its characteristics as well as its performance through conventional numerical solutions [9, 10, 13-15, 31, 40, 41] and computer simulations [7, 42].

This chapter addresses the following research questions regarding the scaling property of ICN.

- How scalable an ICN as a large-scale cache network is for its network size (i.e., the number of routers) in terms of, in particular, user-level performance?

  Major drawback of past studies in the literature is in their emphasis on cache hit ratios. Estimating cache hit ratios at intermediate routers are quite important to understand the characteristics of a cache network. However, cache hit ratios

of routers are one of *system-level* performance metrics, so that higher cache hit ratios do not necessarily imply better *user-level* performance.

In this chapter, we try to reveal the relation between the network size and one of major user-level performance metrics — the average content delivery delay of every entity. The content delivery delay is composed of the two-way (i.e., end-to-end) transfer delay between an entity and the repository and the processing delay at the repository.

- How the cache hit ratios of routers are distributed on a large-scale ICN?

  As we have explained above, understanding of cache hit ratios themselves are not sufficient, but those are still important to capture the behavior of a large-scale cache networks.

  On a complex and large-scale ICN, even if every router is provided with the equal amount of content caches, utilization of content caches should differ at one router and another. For instance, content caches of routers at the network edges might or might not be better utilized compared with content caches of router at the network cores. Different routers (and their content caches) works differently. Thus, it is quite important to understand the distribution of cache hit ratios of routers on an ICN. If we could identify the most *beneficial (contributing)* routers in an ICN, such understanding would be helpful to, for instance, router-capability dimension (e.g., how much content caches should be provided to different routers).

- What scaling properties does an ICN exhibit when the network size grows indefinitely?

  If the size of an ICN would evolve continuously (i.e., the network size would reach and go beyond global-scale (e.g., Internet-scale)), what does happen? This might be a non-practical question since chances that a single ICN architecture would replace the majority of Internet routers and dominate the globe are not that high. However, understanding such scaling property of an ICN is beneficial to *understand* fundamental properties of ICN.

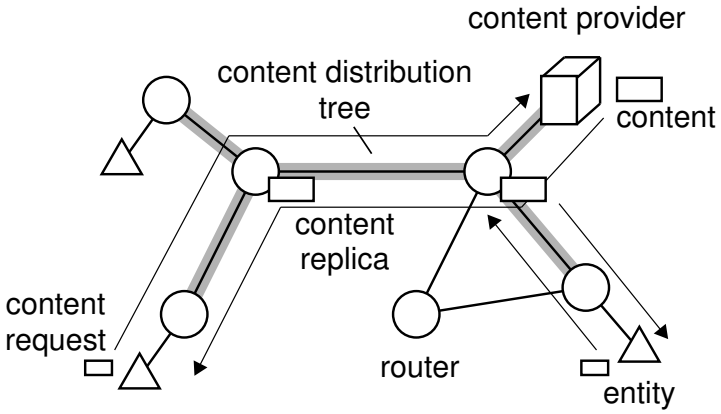


Figure 4.1: An example of content distribution tree comprised of a set of paths from an entity to the content provider

In this chapter, we focus on a large-scale ICN, and derive the cache hit probability at each router, the average content delivery delay for each entity, and the average content delivery delay of all entities over *a content distribution tree* comprised of a single content provider, multiple routers, and multiple entities (see Fig. 4.1).

This chapter is organized as follows. First, Section 4.2 summarizes previous works related to mathematical analyses of ICNs. Section 4.3 explains our analytic model used throughout this chapter. Section 4.4 approximately derives the cache hit probability at each router and the average content delivery delay for each entity over a content distribution tree. Section 4.5 investigates, through several numerical examples, the effect of the topology and the size of the content distribution tree and the cache size at routers on the average content delivery delay of all entities. Finally, Section 4.6 provides the summary of this chapter.

4.2 Related Work

In the literature, a number of analytical studies on ICNs with different network topologies have been performed [9, 10, 13-15, 31, 40]. Different from TCP/IP networks, ICNs generally have in-network caching, so that analytic models of ICNs are rather complicated. Hence, performance analyses of ICNs with rather simple network topologies have been extensively performed [9, 10, 13].

For instance, Carofiglio *et al.* analyze the performance of CCN in a cascaded net-

work topology and a perfect $m-$ary tree [9]. In [9], the authors focused on two types of network topologies: a cascaded network topology in which routers are connected in serial and a perfect m-ary tree. The authors derived cache hit probability at each router, content delivery delay, and throughput of each entity. As a result, the authors showed that the content delivery delay and throughput depends on the cache size at router, the size of content, and a content popularity distribution.

Performance analyses of cache networks [33] and for CCNs [14, 31] with a *general* network topology have been performed. For instance, the authors of [31] analytically obtained the content delivery delay, throughput, and availability for each entity in a general network comprised of multiple routers, multiple repositories, and multiple entities. Furthermore, through several numerical examples, the authors investigated how network topology affects the performance of ICN and showed that the benefits of performance improvement by contents caching (i.e., reduction in content delivery delay and improvement in availability) were more pronounced when the router was closer to the repository.

The limitation of those performance analyses is in computational complexity; i.e., it is difficult to analyze the performance of large-scale ICNs with those numerical approaches. MCA algorithm used in those performance analyses calculates the cache hit probability at each node by repeatedly applying SCA (Single-Cache Approximation) algorithm [34], which calculates the cache hit probability at a single node, to each node. To calculate the cache hit probability by MCA algorithm, iterative calculation is required [33]. For this reason, the computation complexity of MCA algorithm depends on the network size as well as the number of contents in a network and the cache size at node.

An approximate solution with reduced computational complexity has been proposed in [40]. Specifically, by extending Che's approximation [39], the authors of [40] obtained the cache hit probabilities at nodes, considering several content replacement algorithms (e.g., LFU and LRU) and several content replication strategies (e.g., LCE (Leave Copy Everywhere) and LCD (Leave Copy Down)). Moreover, the authors evaluated the performance of the content replacement algorithms and content

replication strategies in terms of the cache hit probability. The authors of [40] focuses on system-level performance such as cache hit probability. On the contrary, in this chapter, we focus on user-level performance such as average content delivery delay.

4.3 Analytic Model

Our analytic model is a content distribution tree composed of N routers excluding entities and the repository (see Fig. 4.2). All contents in a network are stored in the single repository. The repository is connected to the root node of the content distribution tree. For simplicity, a single entity is directly connected to each of other $(N-1)$ routers. We intentionally assume that only a single repository exists in a network to make our model analytically tractable. In the field of performance analyses of ICN (e.g., [9, 13, 15]), this assumption has been widely adopted.

Generally, the topology of an ICN is not a tree, but a set of content distribution paths from a single repository to all entities can be regarded as a tree. In ICN, a path between an entity and a repository is determined by a routing protocol. In this chapter, we consider the case with a static routing protocol; i.e., a static content distribution tree from the single repository to all entities is assumed. A router decides the direction to which a request packet is forwarded based on its routing table (in CCN and NDN, it realized with FIB (Forwarding Information Base)). Because the single repository stores all contents in the network, the router forwards the request packet to its upstream router.

In this chapter, we focus on a general content distribution tree defined as follows. The root node connected to the repository is called as the 1st level router, and the degree distribution of routers located at the $h\,(\geq 1)$-th level (i.e., nodes located at $(h-1)$ hops away from the root of the content distribution tree) is denoted as $P_h(d)$.

A set of contents stored in the repository is denoted by $\mathcal{C}$. For simplicity, the size of all contents is assumed to be identical. In this chapter, we assume a content delivery service on ICNs as like Web on the Internet, so that we assume that every content is comprised of a single packet. In [43], it has been reported that the size of popular

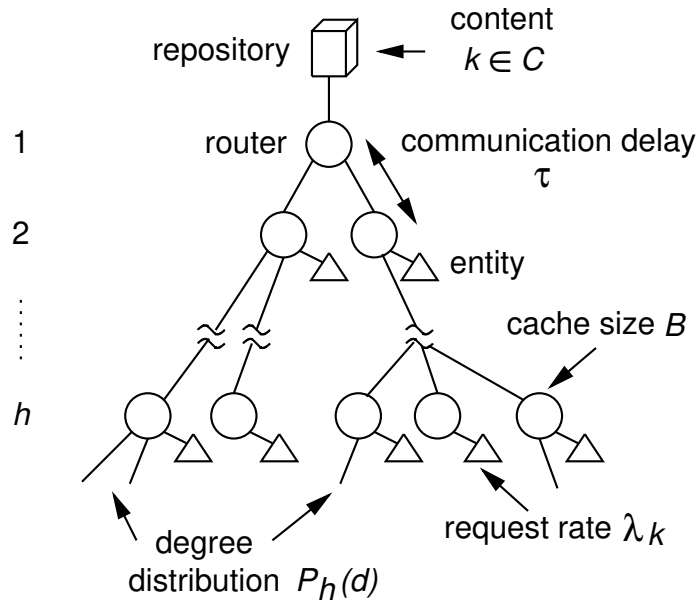


Figure 4.2: Analytic model

requested contents in HTTP traffic on Web is approximately 2 [Kbyte]. Also, in CCN and NDN, each of which is one of ICN architectures, it is specified that the size of a Data packet (i.e., response packet) is 1.5 [Kbyte]. Thus, we assume that an entity can acquire a content with a single request packet.

The cache size at all routers is equally denoted by B. In this chapter, for simplicity, we assume that all cache sizes are identical. However, our analytic model is expandable for different cache sizes at h-th level router. The content replacement algorithm and content replication strategy at all routers are respectively assumed to be LRU (Least-Recently Used) and LCE (Leave Copy Everywhere). The rationale behind this assumption is that the combination of LRU and LCE can be regarded as the baseline for other content replacement algorithms and content replication strategies. Also, this combination is widely used in existing performance analyses of ICNs [9, 10, 44].

All communication delays between adjacent routers (i.e., the sum of the transfer delay and the processing delay) are equally denoted by τ. We assume that other communication delays (i.e., router-to-entity, entity-to-router, router-to-repository, and repository-to-router communication delay) are negligibly small. We assume that bandwidths of all links are sufficiently large. Thus, it is assumed that request and content

Table 4.1: Definition of symbols

N	the number of routers
$P_h(d)$	the degree distribution of h-th level router
$\overline{d}_h$	the average degree of h-th level routers
$\nu(h)$	the average number of h-th level routers
$\mathcal{C}$	A set of contents stored in the repository
τ	communication delay
λ_k	request rate for content k
B	cache size at a router
$q_k(h)$	cache hit probability of content k at h-th level router
$r_k(h)$	arrival rate of requests for content k at h-th level router
$D_k(h)$	average content delivery delay of an entity connected to a h-th level router for content k
$D(h)$	average content delivery delay of an entity connected to a h-th level router for all contents
D	average content delivery delay of an entire network

packets are never discarded in the network due to buffer overflow.

Every entity randomly requests content $k\,(\in \mathcal{C})$ at the rate of λ_k. Namely, every entity continuously sends request packets for content k to the nearest router at the rate of λ_k.

Definition of symbols used throughout our analysis is summarized in Table. 4.1.

4.4 Analysis

$(N-1)$ entities send request packet for content $k\,(\in \mathcal{C})$ at the rate of λ_k, which will be, if not cached at an intermediate router, delivered to the repository connected to the root of the content distribution tree. A router receives request packets from either the directly-connected entity or downstream routers in the content distribution tree. When a router receives a request packet, it checks whether a requested content exists in its own cache. When the corresponding content is cached in the content cache, the router immediately returns the content as a content packet. Otherwise, the router forwards the received request packet to the upstream router in the content distribution tree.

The arrival rate of request packet for content k at the h-th level router in the content distribution tree is denoted by $r_k(h)$. Also, the cache hit probability of request

packets for content k is denoted by $q_k(h)$.

Utilizing the relative request rate of content k among all contents, $r_k(h)/\sum_{l\in\mathcal{C}} r_l(h)$, the cache hit probability of content k, $q_k(h)$, is approximately given by

$$q_k(h) \simeq B\,\frac{r_k(h)}{\sum_{l\in\mathcal{C}} r_l(h)}. \tag{4.1}$$

Note that more accurate cache hit probability has been derived in, for instance, [34] than that of the above equation. However, for simplicity, we intentionally use Eq.(4.1) throughout this chapter.

The arrival rate of request packets for content k at the h-th level router in the content distribution tree, $r_k(h)$, is given by the summation of (1) the transmission rate λ_k of request packets from a directly-connected entity and (2) the total reception rates of request packets from the $(h+1)$-th level routers. Let $\overline{d}_h \equiv \sum_d d\,P_h(d)$ be the average degree of the h-th level routers. The arrival rate $r_k(h)$ of request packets is given by

$$r_k(h) = \lambda_k + (\overline{d}_h - 1)\,(1 - q_k(h+1))\,r_k(h+1). \tag{4.2}$$

The average content delivery delay of an entity connected to a h-th level router in the content distribution tree for content k is denoted as $D_k(h)$. The content delivery delay is the time elapsed since an entity connected to a h-th level router emits a request packet for content k until the entity receives the corresponding content packet. Note that the content delivery delay is also referred to as VRTT (Virtual Round-Trip Time) [9, 14].

A request packet for content k, which is sent from an entity connected to a h-th level router in the content distribution tree, is resolved by (1) any router caching content k along the path between the entity and the repository or (2) the repository storing content k.

Let $p_k(h,i)$ be the probability that a $i\,(\leq h)$-th level router returns the corresponding content packet when the router receives a request packet for content k originated from an entity at a h-th level router in the content distribution tree. $p_k(h,i)$ is given

by

$$p_k(h,i) = \left(\prod_{j=i+1}^{h} (1 - q_k(j)) \right) q_k(i). \tag{4.3}$$

Therefore, the average content delivery delay $D_k(h)$ of an entity connected to a h-th level router for content k is given by

$$\begin{aligned} D_k(h) = & \sum_{i=1}^{h} p_k(h,i)\, 2\,\tau\,(h-i) \\ & + \left(1 - \sum_{i=1}^{h} p_k(h,i)\right) 2\,\tau\,(h-1). \end{aligned} \tag{4.4}$$

The average content delivery delay $D(h)$ of an entity connected to a h-th level router for all contents is given by the weighted average of $D_k(h)$'s. Thus, we have

$$D(h) = \sum_{k \in C} \frac{\lambda_k}{\sum_{l \in C} \lambda_l} D_k(h). \tag{4.5}$$

Let $\nu(h)$ be the average number of h-th level routers in the content distribution tree.

$$\nu(h) = \overline{d}_1 \prod_{i=2}^{h-1} (\overline{d}_i - 1) \tag{4.6}$$

The average content delivery delay D of $(N-1)$ entities (i.e., the average content delivery delay of the entire network) is given by

$$D = \frac{\sum_{h=2}^{\infty} \nu(h)\, D(h)}{N-1}. \tag{4.7}$$

Finally, we investigate the asymptotic behavior of content delivery when the network size N increases.

For simplicity, we assume that the average degree at all levels in the content distribution tree is equal; i.e., $\overline{d}_h = \overline{d}$.

Since the average degree at all levels except the root node and leaf nodes is $\overline{d} \geq 2$,

the arrival rate $r_k(h)$ of request packets for content k at h-th level routers increases exponentially as the level h decreases. Thus, $\lambda_k \ll r_k(h)$ except near-leaf routers in the content distribution tree. So, Eq. (4.2) can be approximated by

$$r_k(h) \simeq (\overline{d}_h - 1)\,(1 - q_k(h+1))\,r_k(h+1). \tag{4.8}$$

Let us focus on two contents, a and b. Without loss of generality, we assume $0 < \lambda_a \leq \lambda_b$. We also assume that the request rate of content b is the highest among all contents (i.e., $\lambda_b = \max_{k \in \mathcal{C}} \lambda_k$).

Let $\Lambda(h)$ be the aggregate arrival rate of request packets at a h-th level router.

$$\Lambda(h) \equiv \sum_{k \in \mathcal{C}} r_k(h) \tag{4.9}$$

Let $\psi_k(h)$ be the relative request rate of content k among all contents at a h-th level router.

$$\psi_k(h) \equiv \frac{r_k(h)}{\Lambda(h)} \tag{4.10}$$

We denote the difference in relative request rates of contents a and b by $\Delta(h) \equiv \psi_b(h) - \psi_a(h)$. We have

$$\begin{aligned}
\Delta(h) &= \frac{r_b(h) - r_a(h)}{\Lambda(h)} \\
&< \frac{r_b(h) - r_a(h)}{(\overline{d}_h - 1)(1 - q_b(h+1))\Lambda(h+1)} \\
&\simeq \Delta(h+1)\frac{1 - B(\psi_a(h+1) + \psi_b(h+1))}{1 - B\,\psi_b(h+1)} \\
&< \Delta(h+1)
\end{aligned} \tag{4.11}$$

In particular, when $1 - B(\psi_a(h+1) + \psi_b(h+1)) > 0$, that is, $\psi_a(h+1) + \psi_b(h+1) < \frac{1}{B}$, $r_b(h) \geq r_a(h)$ is satisfied. Thus, we have

$$\Delta(h) \geq 0. \tag{4.12}$$

Therefore, when the network size N is sufficiently large, the relative request rates of contents a and b are asymptoted.

$$\lim_{h\to 0} \psi_a(h) = \lim_{h\to 0} \psi_b(h) = \psi^* = \frac{1}{|\mathcal{C}|} \tag{4.13}$$

Similarly, the cache hit probability of content k at h-th level router in the content distribution tree is asymptoted.

$$\lim_{h\to 0} q_k(h) = q^* = B\,\psi^* = \frac{B}{|\mathcal{C}|} \tag{4.14}$$

4.5 Numerical Examples

In this section, we investigate the effect of the topology and the size of the content distribution tree and the cache size at routers on the average content delivery delay of all entities.

To clarify how the topology of a large-scale ICN affects the average content delivery delay, we use the three types of content distribution trees — perfect m-ary tree, linearly-shrinking tree, and reciprocally-shrinking tree.

- Perfect m-ary tree

  A content distribution tree is given by the perfect m-ary tree, in which all nodes except leaf nodes have the identical number m of child nodes. The degree distribution of a h-th level node $P_h(d)$ is defined as follows. For internal (i.e., neither root nor leaf) nodes, $P_h(d)$ is 1 if $d = m + 1$ and is 0 otherwise. In the case with a root node, $P_h(d)$ is 1 if $d = m$ and is 0 otherwise. In the case with a leaf node, $P_h(d)$ is 1 if $d = 1$ and is 0 otherwise. In the following numerical examples, we use $m = 2$ (perfect 2-ary tree) which is widely used in performance evaluations of ICN [9, 42]. Also, we used $m = 3$ (perfect 3-ary tree) as a more densely-connected tree than perfect 2-ary tree.

- Linearly-shrinking tree

  In linearly-shrinking tree, the number of child nodes decreases linearly to the

number of hops from the root node. Namely, the number of child nodes at the 1st-level node in a content distribution tree is m_{max}, and that of the h-th level node is $\max(0, m_{max}-(h-1))$. In the following numerical examples, $m_{max} = 10$ is used.

- Reciprocally-shrinking tree

  In reciprocally-shrinking tree, the number of child nodes decreases inverse-proportionally to the number of hops from the root node. Namely, the expected value of the number of child nodes at the 1st-level node in the content distribution tree is m_{max}, and that of the h-th level node is m_{max}/h. In the following numerical examples, $m_{max} = 20$ is used.

Because we assume Internet-scale ICNs, in linearly-shrinking tree and reciprocally-shrinking tree, we selected m_{max} as 10 and 20, respectively, so that one million nodes (routers) at most exist.

In our numerical examples, we consider the application such as Web browsing on ICN. In [45] (published in 2004), it is reported that the delay (i.e., the content delivery delay) required for Web browsing is the order of several hundred milliseconds. So, in the future when ICN will be realized, the required delay for Web browsing should be much smaller than that reported in [45].

The repository is provided with 100 types of contents, $\mathcal{C} = \{1, \ldots, 100\}$, each of which has different popularity. The content request rate at each entity is set to $\lambda_k = k/\sum_{k\in\mathcal{C}} k$. Hence, content $k = 100$ is the most popular content and content $k = 1$ is the least popular content. Recall that, as Eqs. (4.1) and (4.5) imply, the cache hit probability and the content delivery delays in our analysis are determined not by *the absolute value* of the request rate for a content but by *the relative value* of the request rate for all contents.

Unless explicitly stated, we use the following parameters: cache size at router $B = 10$ [content] and the communication delay $\tau = 1$ [ms].

Note that we intentionally use a small number of contents (i.e., 100) and a small cache size (i.e., $B = 10$ [content]) since, as Eq. (4.14) implies, the average content de-

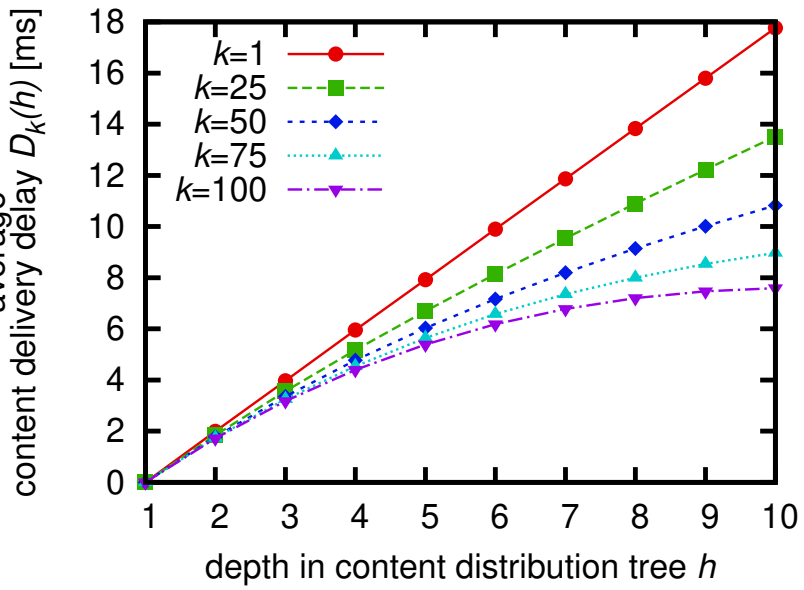


Figure 4.3: Average content delivery delay of an entity connected to h-th level router for content k, $D_k(h)$, in perfect 2-ary tree

livery delay is dominated by the *ratio* of the cache size to the number contents. For this reason, even if the number of contents and the cache size increases, our observations presented hereafter will not change significantly as long as the ratio is maintained.

First, we investigate the average content delivery delay of entities connected at the different levels of the content distribution tree.

Figure 4.3 shows the average content delivery delay $D_k(h)$ of entities connected at the h-th level router and requesting for content k (Eq. (4.4)). In this figure, the results for perfect 2-ary tree are shown.

One can find from this figure that (1) the average content delivery delay of entities *far away* from the repository is significantly affected by the content popularity, and that (2) the average content delivery delays of entities *near* the repository is almost independent of the content popularity. Such phenomenon is resulted from *filter effect* in cache networks — the content popularity distribution is gradually *smoothed* as the request packet passes through multi-stage caches.

Figure 4.4 exhibits the filter effect (Eq.(4.1)) in a large-scale ICN. Similar to Fig. 4.3, the results for perfect 2-ary tree are shown in Fig. 4.4. This figure shows cache hit probabilities for contents $k = 1, 25, 75, 50, 100$ at the h-th level router in a content

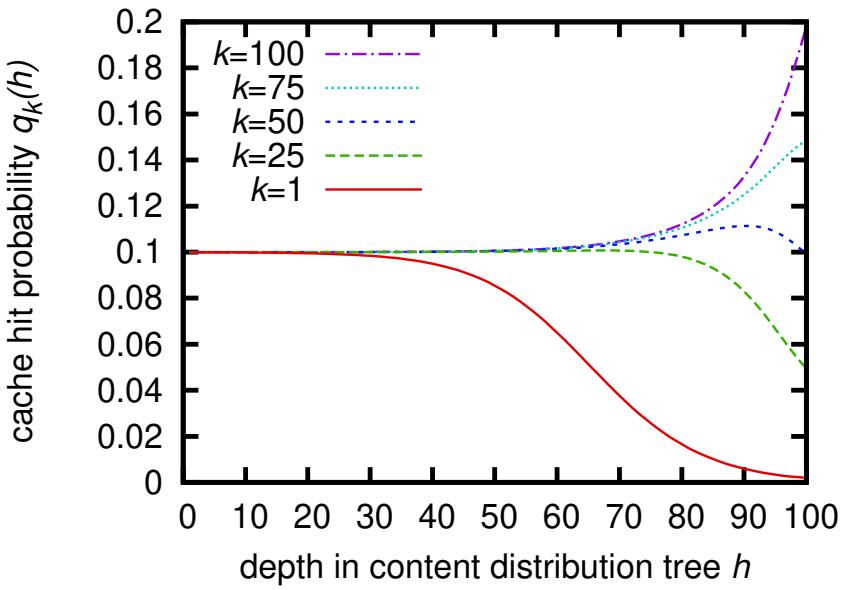


Figure 4.4: Cache hit probability for content k at h-th level router in a content distribution tree $q_k(h)$ in perfect 2-ary tree

distribution tree. From this figure, it is found that the variation in cache hit probabilities for each content is suppressed as the level in the content distribution tree changes from leaf nodes to the root node. It is also found that the cache hit probabilities for all contents equally converge to 0.1 (= $B/|\mathcal{C}|$) regardless of the popularity of contents as the level in the content distribution tree decreases. Furthermore, it is found that, at earlier stage (i.e., $h = 80$), cache hit probabilities for popular contents ($k = 25, 50, 75, 100$) are nearly 0.11. This implies that cache hit probabilities for most of popular contents except for few unpopular contents are converged in medium-scale ICNs.

To examine the validity of our approximate analysis, Fig. 4.5 shows average content delivery delays obtained with our approximate analysis and performance analysis of CCN in Chapter 3. In this figure, since the performance analysis of CCN, which is plotted as "MCA-based analysis", is not scalable technique in terms of the network size, results when changing the number of routers N to 1,000 are shown. From these results, it is found that our approximate analysis shows good agreement with the performance analysis of CCN in perfect 2-ary tree.

Next, we investigate the average content delivery delay of all entities in different

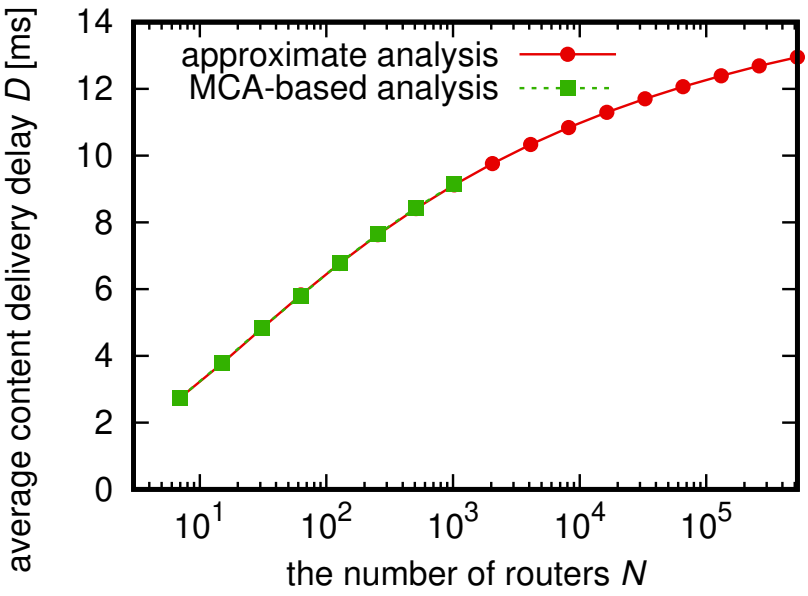


Figure 4.5: Comparison of average content delivery delays obtained with our approximate analysis and performance analysis of arbitrary CCN network

types of content distribution trees when the network size is varied. In the following numerical examples, we obtain the average content delivery delay of all entities using the cache hit probabilities at a routers obtained from Eq. (4.14).

Figures 4.6 through 4.9 show the average content delivery delays of all entities (Eq. (4.7)) in perfect 2-ary tree, perfect 3-ary tree, linearly-shrinking tree, and reciprocally-shrinking tree. In these figures, the cache size B at a router is changed between 0 and 30 [content].

These results indicate that (1) when the cache size is not small (e.g., $B = 10$), the larger the network size becomes, the larger the average content delivery delay becomes, and that (2), on the contrary, when the cache size is large (e.g., $B \geq 20$), the increase in the average content delivery delay can be suppressed even when the network size becomes large. From performance analyses of ICN, the ratio of the cache size to the number of contents in a network (i.e., $B/|C|$) should be between 10^{-5} and 10^{-1} [42]. Therefore, with the current technology, the large cache size (e.g., $B \geq 20$) might not be practical. In the future (e.g., 20 years later), however, because of the development of memory technology, the above observations (e.g., the convergence occurrences of the cache hit probability and the content delivery delay) might be valid.

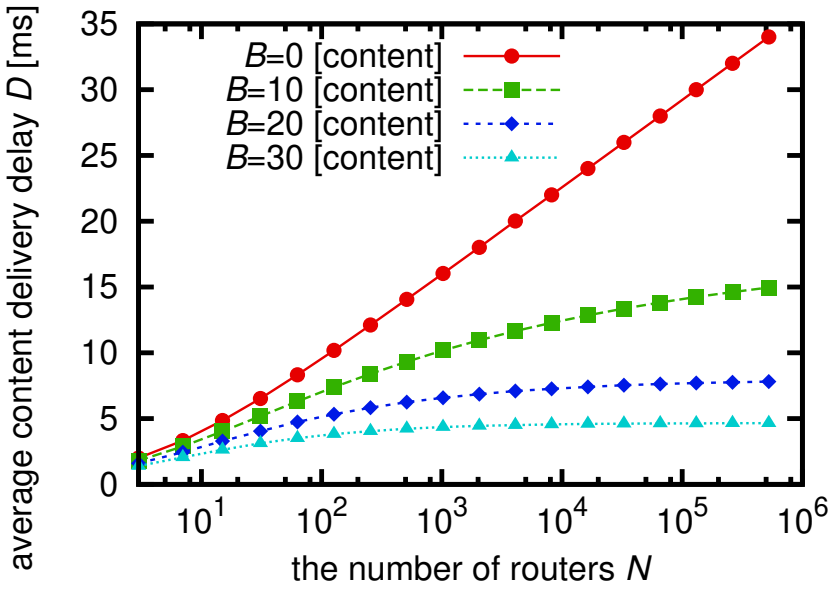


Figure 4.6: Effect of the cache size at a router on the average content delivery delay of an entire network in perfect 2-ary tree

Furthermore, we examine in detail the relation between the cache size and the average content delivery delay. Figure 4.10 shows the average content delivery delay of an entire network (Eq. (4.7)) in prefect 2-ary tree when the cache size is varied from 0 to 30 [content]. In this figure, results for different network sizes (i.e., the depth of content distribution tree h_{max}) are plotted. From these results, it is found that the average content delivery delays except for the case of a small-scale network (e.g., $h_{max} = 5$) are inversely proportional to the cache size. In particular, the average content delivery delays converge at approximately $B = 20$ [content], regardless of the network size.

We discuss the effect of assumptions in our analytic model on numerical examples. First, we discuss the effect of the difference between real network topologies and our simplified network topologies on the average content delivery delay. In this chapter, we assume a heterogeneous content distribution tree where nodes in a level have the different numbers of child nodes like realistic network topologies, rather than a homogeneous content distribution tree. The difference in the average content delivery delays in homogeneous and heterogeneous content distribution trees is mainly caused by the difference of the average numbers of hops from entities to the

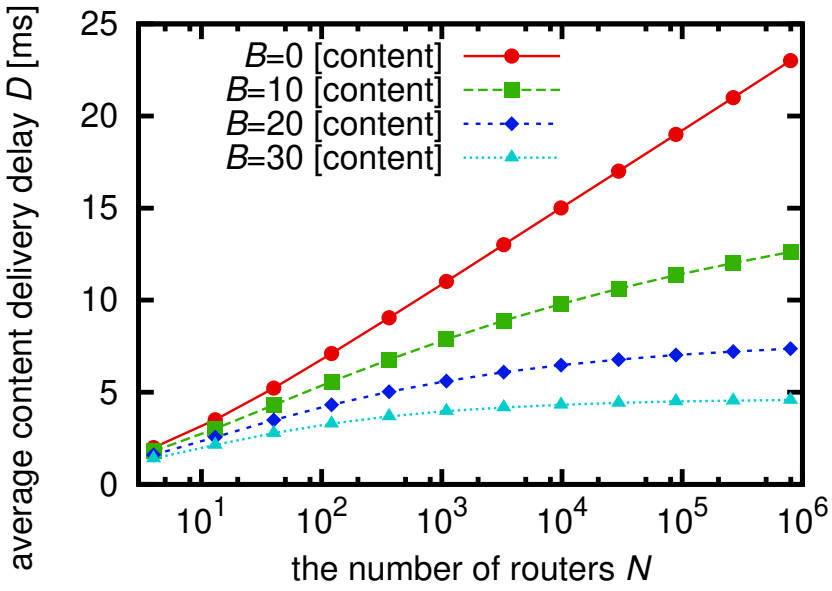


Figure 4.7: Effect of the cache size at a router on the average content delivery delay of an entire network in perfect 3-ary tree

repository. This can be explained by the following reasons; (1) the average content delivery delay can be estimated from the average number of hops and the cache hit probability; (2) as shown in Fig. 4.4, cache hit probabilities can be expected to be converged because of the filter effect in (large-scale) heterogeneous content distribution tree.

Next, we discuss the effect of the queuing delay at a router. If the cache size at a router is small (e.g., $B = 0, 10$), the queuing delay occurred at a router might highly affect the average content delivery delay. Specifically, the queuing delays at routes (in particular, nearly located at the repository) might increase, which leads to the increase in the average content delivery delay of an entire network. On the other hand, if the cache size is large (e.g., $B = 20, 30$), the transmission of request and response packets can be suppressed because of content caching. Therefore, it can be expected that the queuing delay at routes does not highly affect the average content delivery delay.

Finally, based on our observations, we answer the research questions explained in Section 4.1.

- How scalable an ICN as a large-scale cache network is for its network size (i.e., the number of routers) in terms of, in particular, user-level performance?

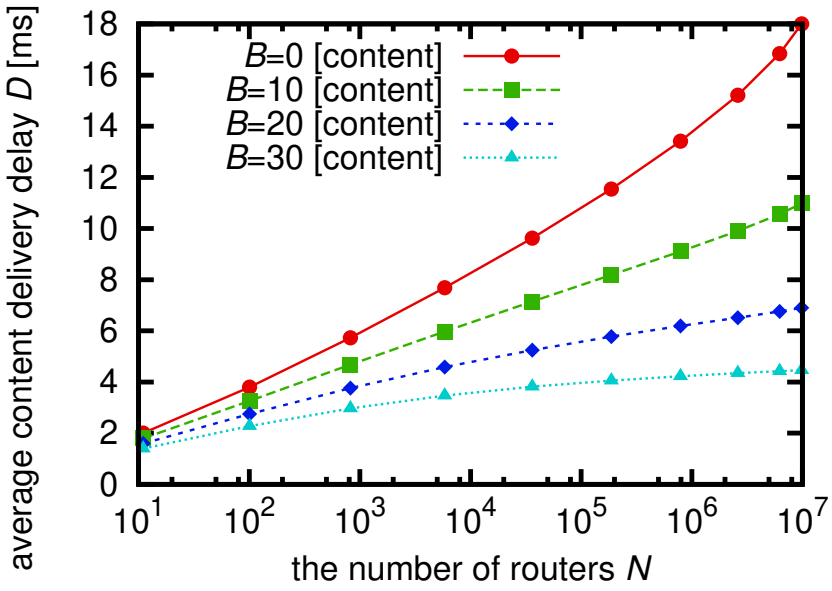


Figure 4.8: Effect of the cache size at a router on the average content delivery delay of an entire network in linearly-shrinking tree

As Figs. 4.6 through 4.9 indicate, the average content delivery delay of ICNs converges to a constant value (i.e., does not increase indefinitely) if the cache size of routers are not small, which implies *high scalability* of ICNs in terms of the network size.

- How the cache hit ratios of routers are distributed on a large-scale ICN?

  The cache hit ratios of routers are significantly affected by the *distance* from requesting entities. Consequently, in a large-scale ICN, the content popularity distribution observed by *edge routers* are diverse whereas that by *core routers* are rather uniform, which implies that caches at edge routers are more advantageous (i.e., effective) than those at core routers.

- What scaling properties does an ICN exhibit when the network size grows indefinitely?

  Even when the network size would grow indefinitely, the average content delivery delay is upper-bounded by a constant value if routers in the network are provided with a fair amount of content caches. The rationale behind this behavior is caused by the filter effect in multi-stage caching.

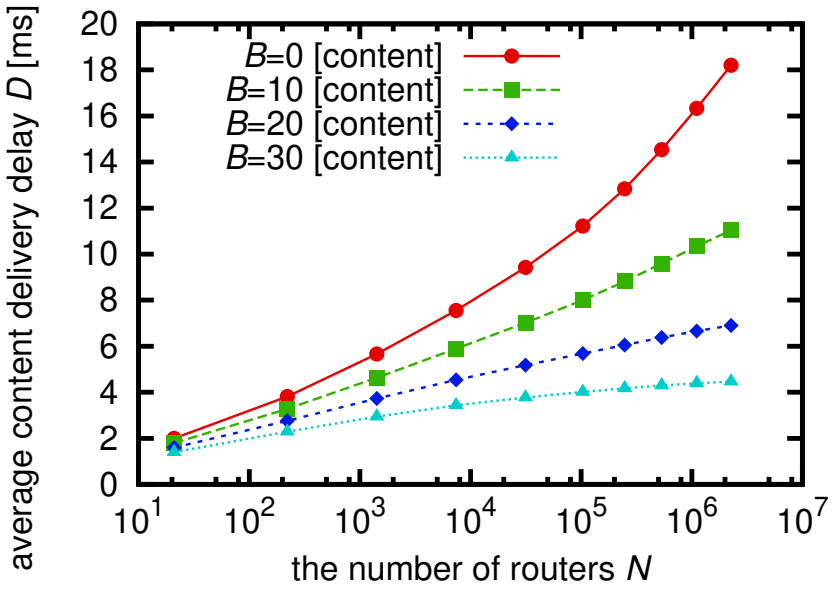


Figure 4.9: Effect of the cache size at a router on the average content delivery delay of an entire network in reciprocally-shrinking tree

4.6 Summary

In this chapter, we have focused on a large-scale ICN and derived the cache hit probability at each router, the average content delivery delay for each entity, and the average content delivery delay of all entities over a content distribution tree comprised of a single repository, multiple routers, and multiple entities. Furthermore, through several numerical examples, we have investigated the effect of the topology and the size of the content distribution tree and the cache size at routers on the average content delivery delay of all entities. Our findings include that the average content delivery delay of ICNs converges to a constant value if the cache size of routers are not small, which implies high scalability of ICNs in terms of the network size, and that even when the network size would grow indefinitely, the average content delivery delay is upper-bounded by a constant value if routers in the network are provided with a fair amount of content caches.

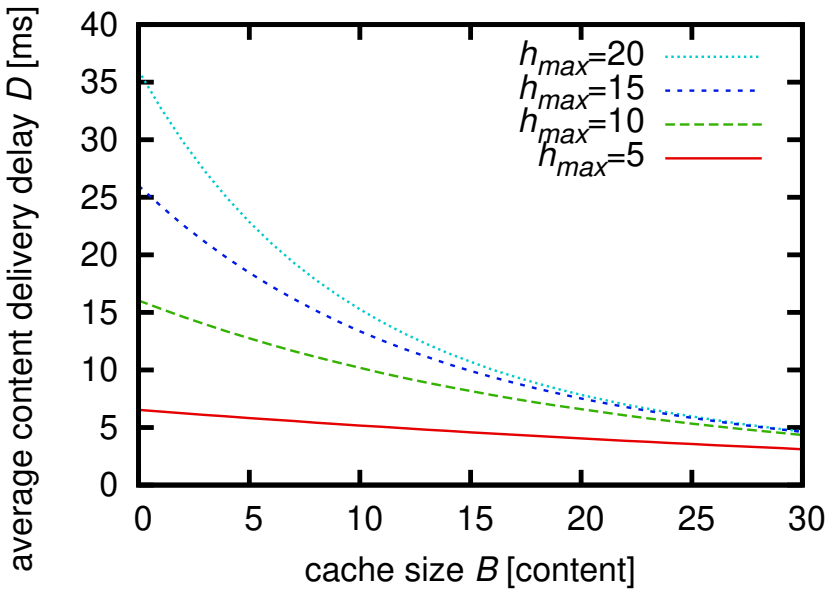


Figure 4.10: Relation between the cache size at a router and the average content delivery delay of an entire network in perfect 2-ary tree

Chapter 5

On the Optimality of Shortest-Path Routing in Information-Centric Networking

5.1 Introduction

In the recent years, Information-Centric Networking (ICN) that mainly focuses on contents that are transferred and received instead on end hosts that transmit and receive contents has been under the spotlight [1-3]. Two major network architectures for realizing ICNs are CCN (Content-Centric Networking) [1] and NDN (Named Data Networking) [2]. In ICNs, the content request packet sent from an entity (i.e., content consumer) is forwarded to a repository (i.e., content provider) that stores the content based on its content identifier and routing tables of routers along the path. The requested content is returned to the entity from the repository as a response packet by retracing the trajectory of the request packet.

Two major challenges in the ICN architecture design are *content caching* and *content routing* [16]. Content caching improves the way a router in a network caches contents for performance improvement in terms of the reduction in the traffic volume transferred through the network. On the other hand, content routing is aimed at effectively discovering the content by appropriately selecting the path from a router to

the nearest repository which stores the content.

A large number of studies have been devoted for designing content caching mechanisms in ICNs [46, 47]. In those studies, several content caching mechanisms to effectively utilize caches of intermediate routers on the path are proposed. For instance, authors of [46] proposed an in-network caching scheme, in which a router caches contents based on the probability calculated from a distance to its destination and the caching capability of other routers along a path. Also, authors of [47] focused on round-trip times for contents measured at a router, and proposed a caching algorithm based on the probability calculated from observed round-trip times.

On the other hand, not many but several studies have been devoted for content routing in ICNs [17, 18, 48]. A few studies focus on routing mechanisms for request and content packets to achieve better performance than that with the shortest-path routing.

A straightforward approach for content routing in ICNs is to utilize a class of shortest-path routing mechanisms. In similar fashion to the conventional IP network, usage of the shortest-path routing simply based on the number of hops or the link-level metrics between a router and a repository, is considered in CCN [1]. However, since routers on the path cache contents in ICNs, the shortest-path routing might not be always optimal.

In the literature, several cache-aware routing mechanisms to take advantage of content caches at intermediate routers have been proposed (see, e.g., [17, 18]). Generally, cache-aware routing mechanisms determine the path through which a request packet is forwarded by taking account of both the proximity of content replicas and the likelihood of cache hits at intermediate routers. Cache-aware routings are expected to reduce the server load as well as the content delivery delay.

In those studies, however, only link-level performance metrics (e.g., cache hit ratio and server load) of content routing have been investigated. Hence, it has been still unclear how the shortest-path routing is effective (or ineffective) in terms of application-level performance metrics (e.g., content delivery delay and throughput). Also, cache-aware routing mechanisms generally rely on the availability of cache hit

ratios and/or cache sizes of routers, which might not be easy to obtain accurately in a timely fashion.

For a given scenario, the shortest-path routing might not provide the best application-level performance. A sophisticated content routing should provide better application-level performance than that with the shortest-path routing. However, under realistic scenarios where many factors (e.g., network topology, bandwidths and propagation delays of links, cache and buffer sizes of routers, and workloads generated from entities) are varying and/or uncertain, an overly-optimized content routing could result in a poor performance.

In this chapter, we investigate the optimality of the shortest-path routing by comparing the performances with shortest-path routing and with an optimal routing obtained by searching all detour paths existing in the vicinity of the shortest-path routing (*optimal k-hop detour routing*). We focus on the average content delivery delay, which is one of the key application-level performance metrics. Through a number of experiments, we compare average content delivery delays with the shortest-path routing and with the optimal k-hop detour routing.

This chapter addresses the following research questions regarding the optimality of shortest-path routing in ICNs.

Q1. Under a given condition, which of the shortest-path routing or the optimal k-hop detour routing is suitable in terms of the average content delivery delay?

Q2. How robust are the shortest-path routing and the optimal k-hop detour routing against measurement errors in cache hit ratios at routers?

In this chapter, we investigate the optimality of the shortest-path routing in diverse scenarios. We quantitatively investigate the optimality of the shortest-path routing in ICNs by comparing the average content delivery delay under the shortest-path routing with that under the optimal two-hop detour routing in several networks (triangular network, seven-node network, grid network, and cluster network). Furthermore, we also investigate the robustness of the shortest-path routing against measurement errors in cache hit ratios at routers.

This chapter is organized as follows. First, Section 5.2 introduces previous works related to content routings in ICNs. Section 5.3 explains the methodology to investigate the optimality of the shortest-path routing in ICNs. Section 5.4 presents experiment results and discusses the optimality of the shortest-path routing. Finally, Section 5.5 provides the summary of this chapter.

5.2 Related Work

In the literature, it is known that sophisticated content routings including the cache-aware routing achieve better performance than the shortest-path routing in ICN [17, 18, 49, 42, 48]. Authors of [17] proposed a cache-aware routing that dynamically selects the path so that the number of hops to retrieve the content can be minimized [17]. They reported that, with their cache-aware routing, the server load can be reduced by approximately 18% from the shortest-path routing. Another cache-aware routing is proposed in [18]. The authors proposed a weight-based cache-aware routing that minimizes the content access delay based on the existence of content cache at routers. Authors of [48] proposed an efficient content routing by adopting a different approach than cache-aware routing. Specifically, in the proposed content routing, a router measures round-trip times for contents returned from repositories, and it probabilistically determines a next node to forward a request packet based on measured round-trip times. Through simulations, it was shown that, compared to the shortest-path routing, the proposed content routing can improve the content delivery delay because of reducing loads occurred at the repository.

However, a few studies reported that there is no significant difference in the performance between the shortest-path routing and the cache-aware routing [50]. Authors of [50] compared the shortest-path routing and the nearest-replica routing, which is one of cache-aware routings, while changing several factors such as the network topology and content request pattern. In nearest-replica routing, a request packet for a content from an entity is delivered to the nearest router/repository which storing the requested content. Although the nearest-replica routing is not a practical rout-

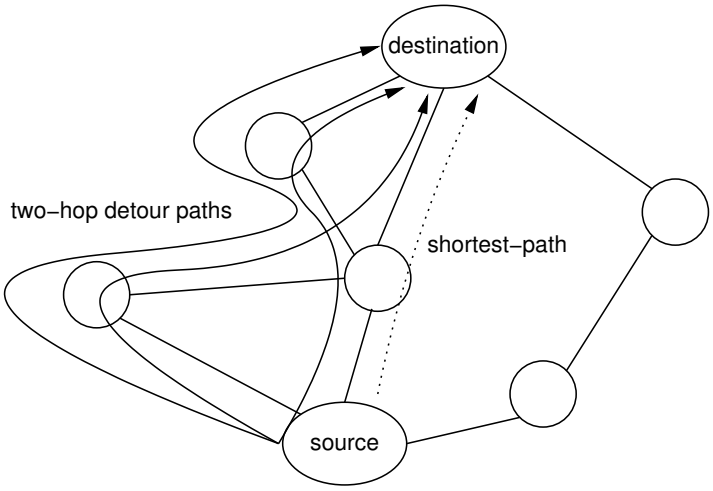


Figure 5.1: Examples of two-hop detour path in the optimal two detour routing

ing mechanism, it can be regarded as a baseline for other content routings in ICNs. In [50], improvements in all performance metrics (e.g., the number of hops required for contents delivery) with the nearest-replica routing is at most 2%, compared to the shortest-path routing.

5.3 Method

5.3.1 Optimal k-Hop Detour Routing

In this chapter, we perform three types of experiments to investigate the optimality of shortest-path routing. In all experiments, we compare the average content delivery delay under the shortest-path routing with that under the *optimal two-hop detour routing*.

The optimal k-hop detour routing is defined as the content routing with the least average content delivery delay among all possible k-hop detour paths obtained from the shortest path (Fig. 5.1). In our experiments, we obtained the optimal two-hop detour routing using an exhaustive search: (1) obtain all two-hop detour paths regarding arbitrary single hops in the shortest-path, (2) calculate average content delivery delays of all two-hop detour paths using our performance analysis, and (3) select the path with the least average content delivery delay.

Note that the optimal k-hop detour routing is not for practical purposes, but for theoretical analyses. The optimal k-hop detour routing is based on the idea that, even with caches at routers, the optimal path should be more or less similar to the shortest-path. Namely, the optimal k-hop detour routing exhaustively searches solution space around the shortest-path to hopefully find a reasonably better path than the shortest-path. However, in reality, near-optimal k-hop detour routing might be realized by shrinking solution space of detour paths.

In what follows, the methodologies of three experiments are explained.

5.3.2 E1: Effect of Giant Cache

In this experiment, the cache size of a specific router is varied to examine how the existence of a *giant cache* affects the effectiveness of the shortest-path routing as well as the optimal two-hop detour routing.

As network topologies, we use two different network topologies: triangular network and seven-node network shown in Figs. 5.2(a) and 5.3(a). The communication delays of links between an entity and a router are negligibly small and the communication delays of all other links are equally set to 1 [ms].

Cache sizes of all routers are equally set to C = 5 or 10 [content]. The cache size of a specific router (router 1, shaded router in Figs.5.2(a) and 5.3(a)) is varied between 0–40 [content] for investigating how the cache size affects the optimality of the shortest-path routing. The cache replacement algorithms at all routers are LRU (Least-Recently Used), which is widely used for the performance evaluation and analyses of cache networks [51].

50 contents are placed at one or two repositories, and every entity randomly and continuously requests contents of either the rate k [request/ms] for content k ($1 \leq k \leq 50$), or the popularity following Zipf distribution with the mean of 200 [request/s] and the exponent parameter of 1.0.

Utilizing our ICN performance analysis [31], the average content delivery delay (i.e., the average time required for an entity to retrieve a content) for a given content routing is calculated.

5.3.3 E2: Effect of Cache Sparseness

In this experiment, different from experiment E1, the density of cache-equipped routers (*cache sparseness*) is varied. Namely, instead of changing the cache size of a specific router, cache sizes of all routers in the network are uniformly changed.

As network topologies, we use three network topologies: triangular network, grid network, and cluster network shown in Figs. 5.4(a),5.5(a), and 5.6(a), respectively. Every network topology has a single repository. Similar to experiment E1, the communication delays of links between an entity and a router are negligibly small and the communication delays of all other links are equally set to 1 [ms]. The request rate is given by a Zipf distribution with the exponent parameter of 1.0.

To adjust the density of caches in the network, we define the parameter M called cache sparseness. Cache sparseness M is a positive integer that control the density of cache-equipped routers. Namely, for a given parameter $k\,(0 \leq k < M)$, every router whose identifier i satisfies $i \equiv k \pmod{M}$ has the cache size C_1, and all other routers have the cache size C_2. For instance, all routers have the same cache size for $M = 1$, and the one-fourth of routers have the cache size C_1 and others have C_2 for $M = 4$. We use $C_2 = 0$ [content].

All other conditions are the same with those in experiment E1.

5.3.4 E3: Robustness against Measurement Errors in Cache Hit Ratios

The third experiment investigates how the effectiveness of the shortest-path routing and the optimal two-hop detour routing is affected by measurement errors in cache hit ratios. Effectiveness of cache-aware routings, including the optimal k-hop routing, depends on the accuracy and timeliness of cache hit ratios at routers. In this experiment, we simulate how the performance of the two-hop detour routing is degraded when it chooses paths using *dirty* cache hit ratios.

We use the same conditions with those in experiment E2 except the computation of the average content delivery delay, which will be explained below, and $C_1 = 20$.

For a given condition (e.g., network topology, cache size C_1, cache sparseness M,

and routing algorithm), we first determine routes from all entities to the repository, and compute the cache hit ratios at all routers using our ICN performance analysis [31]. We regard these cache hit ratios as *dirty* cache hit ratios; i.e., contaminated cache hit ratios used by the optimal two-hop detour routing. As *genuine* cache hit ratios, we randomly generate 100 sets of cache hit ratios using a parameter ϵ. Specifically, every cache hit ratio is multiplied by a random number following the uniform distribution $[1-\epsilon : 1+\epsilon]$. By definition, the cache hit ratio is larger than 1 is truncated to 1.

5.4 Results and Discussion

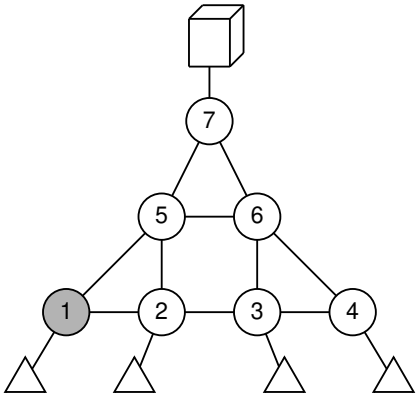
5.4.1 E1: Effect of Giant Cache

Average content delivery delays under shortest-path routing and optimal two-hop detour routing when changing the cache size of router 1 are shown in Figs. 5.2(b), 5.2(c), 5.3(b), and 5.3(c). In these figures, "proportional" indicates the proportional request pattern whereas "Zipf" indicates the Zipf-distributed request pattern.

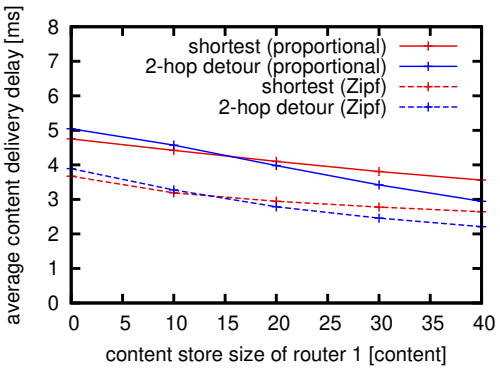
One can find from these figures that the optimality of the shortest-path routing depends on cache sizes of routers on the path. It can also be found that the shortest-path routing is optimal regardless of the content request patterns when cache sizes of all routers are identical. Since the triangular network is equally-balanced, it is intuitive that the shortest-path routing is optimal when cache sizes are the same. But the shortest-path routing is also optimal under the seven-node network which is an unbalanced network.

The larger the cache size of router 1 becomes, the less optimal the shortest-path routing becomes. From Figs. 5.2(b), 5.2(c), 5.3(b), and 5.3(c), it is found that the average content delivery delay under the optimal two-hop detour routing becomes smaller when the cache size ratio (i.e., the ratio of the cache size of router 1 to that of other routers) exceeds approximately 2–3.

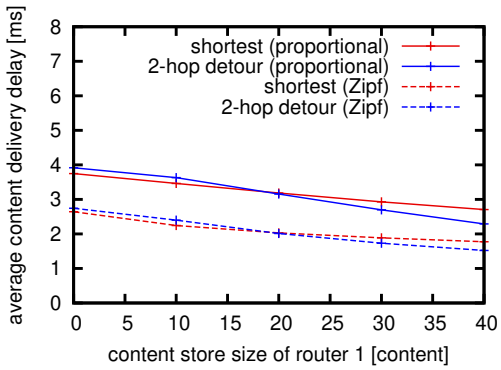
These results show that the shortest-path routing is optimal under a balanced network with comparable cache sizes at routers, and that the optimal two-hop de-



(a) network topology



(b) Average content delivery delays for $C = 5$ [content]



(c) Average content delivery delays for $C = 10$ [content]

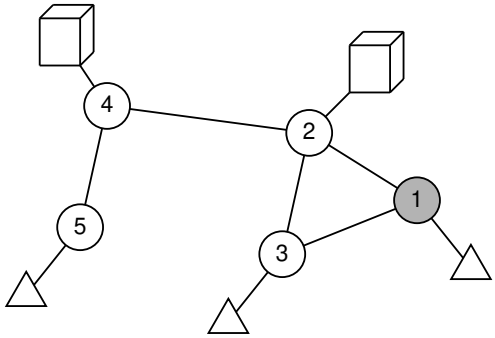
Figure 5.2: Effect of the cache size at the specific router on average content delivery delays (triangular network)

tour routing archives better application-level performance when the cache size ratio is large.

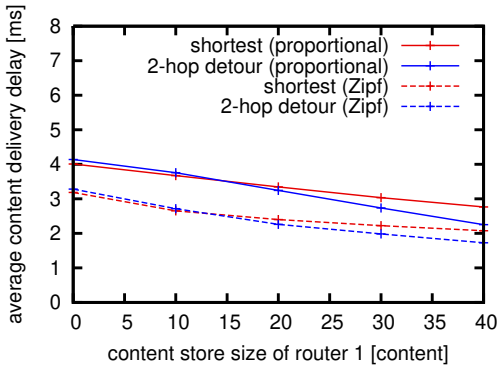
5.4.2 E2: Effect of Cache Sparseness

Average content delivery delays under shortest-path routing and optimal two-hop detour routing when changing cache sparseness M are shown in Figs. 5.4(b),5.5(b) and 5.6(b). To clearly reveal differences in average content delivery delays with the shortest-path routing and the optimal two-hop detour routing, detouring inefficiency (the average content delivery delay with the optimal two-hop routing / that with the shortest-path routing) are plotted in Figs. 5.4(c), 5.5(c) and 5.6(c).

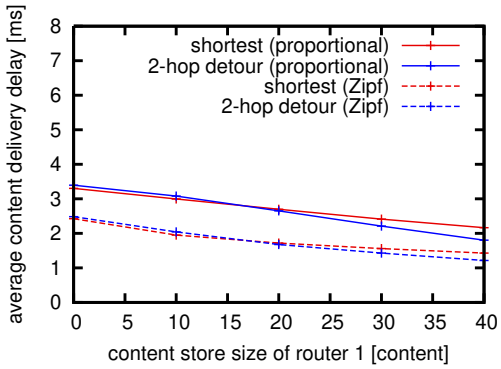
One can find from these figures that the shortest-path routing achieves the best performance when the cache sparseness is very low (i.e., $M = 1$). This observation



(a) network topology



(b) Average content delivery delays for $C = 5$ [content]



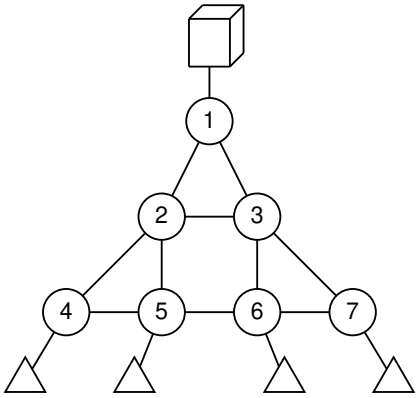
(c) Average content delivery delays for $C = 10$ [content]

Figure 5.3: Effect of the cache size at the specific router on average content delivery delays (seven-node network)

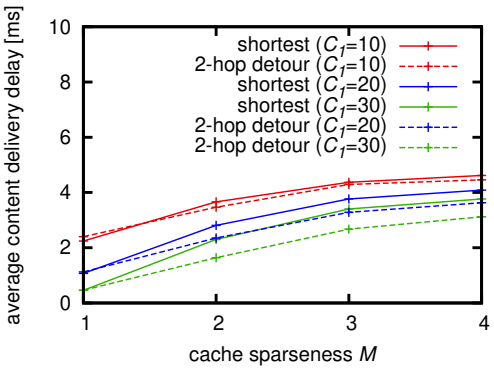
agrees with our finding in the previous section; i.e., all networks used in experiment E2 are equally-balanced so that the shortest-path routing is the optimal.

On the contrary, if the caches are sparse in routers (e.g., $M = 2$ or $M = 3$), the shortest-path routing shows worse performance than that with the optimal two-hop detour routing in triangular and cluster networks. In particular, the optimal two-hop detour routing is quite effective in the triangular network with large cache size (e.g., $C_1 = 30$) and modest cache sparseness (e.g., $M = 2$).

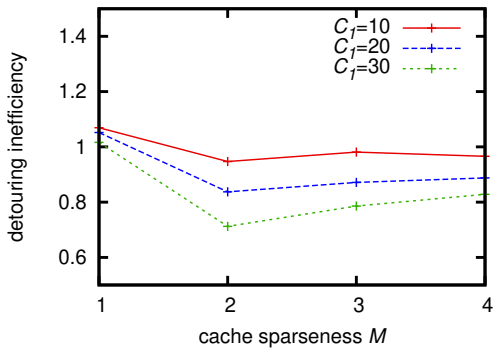
Surprisingly, regardless of the cache sparseness and the cache size, the shortest-path routing is always optimal in the grid network. This implies that the cache-aware routing, including the two-hop detour routing, should be carefully deployed since the (generally complex) cache-aware routing does not always achieve better performance than the simplest shortest-path routing.



(a) network topology



(b) average content delivery delay



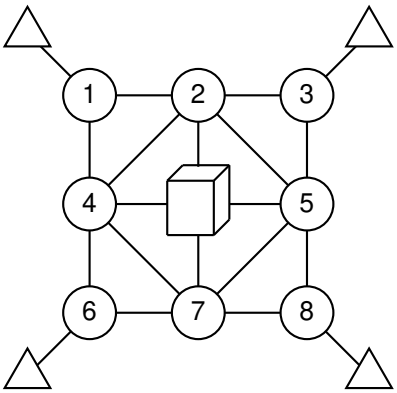
(c) detouring inefficiency

Figure 5.4: Effect of the cache sparseness on average content delivery delays (triangular network)

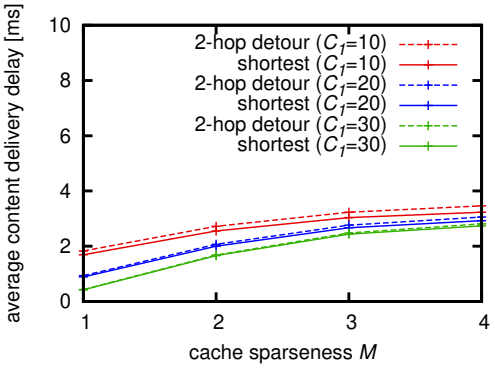
5.4.3 E3: Robustness against Measurement Errors in Cache Hit Ratios

Average content delivery delays under the shortest-path routing and the optimal two-hop detour routing are shown in Figs. 5.7(a) and 5.8(a). Also, degradation factors for a given content routing, which is defined as the ratio of the average content delivery delays calculated from dirty cache hit ratios to those calculated from genuine cache hit ratios, are shown in Figs. 5.7(b) and 5.8(b). Figures. 5.7 and 5.8 show the results for $\epsilon = 0.5$ and $\epsilon = 1.0$, respectively. In those figures, average content delivery delays sorted in descending order are plotted.

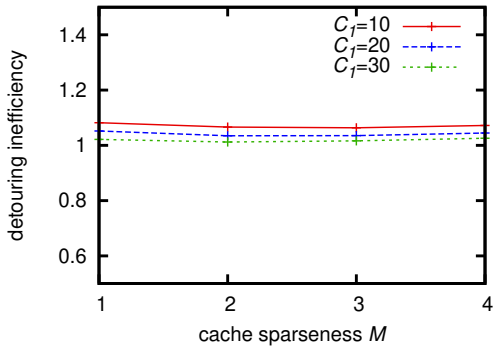
One can find from these figures that, even though the cache hit ratios include errors, the shortest-path routing is still better than the optimal-two hop detour routing in the case of $M = 1$. Hence, both of the shortest-path routing and the optimal two-hop detour routing are affected by measurement errors. In particular, it is found that



(a) network topology



(b) average content delivery delay



(c) detouring inefficiency

Figure 5.5: Effect of the cache sparseness on average content delivery delays (grid network)

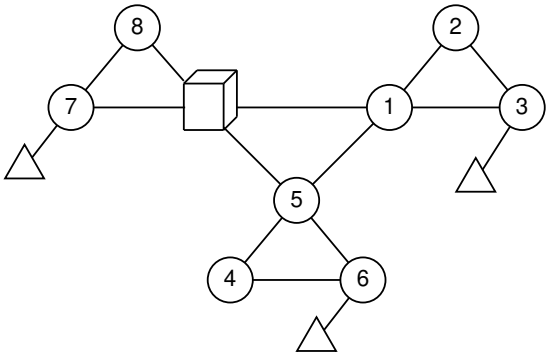
the variation in average content delivery delays with the optimal two-hop detour routing is large in the case of $M = 2, 3$. On the contrary, the shortest-path routing achieves the almost same performance regardless of measurement errors in cache hit ratios in the case of $M = 2, 3$.

5.4.4 Discussion

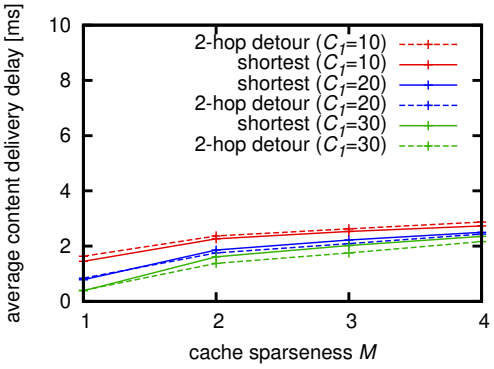
In the following, we answer research questions described in Section 5.1 from observations in experiments E1–E3, and discuss the optimality of the shortest-path routing in ICN.

Q1. Under a given condition, which of the shortest-path routing or the optimal k-hop detour routing is suitable in terms of the average content delivery delay?

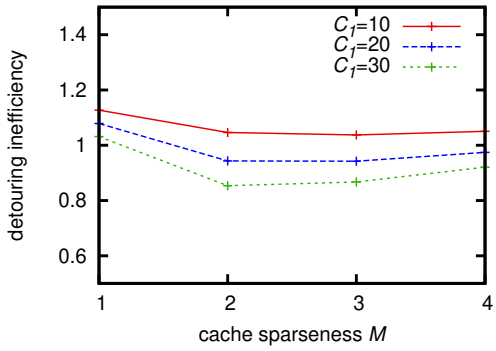
The shortest-path routing is suitable when the network is balanced and cache



(a) network topology



(b) average content delivery delay



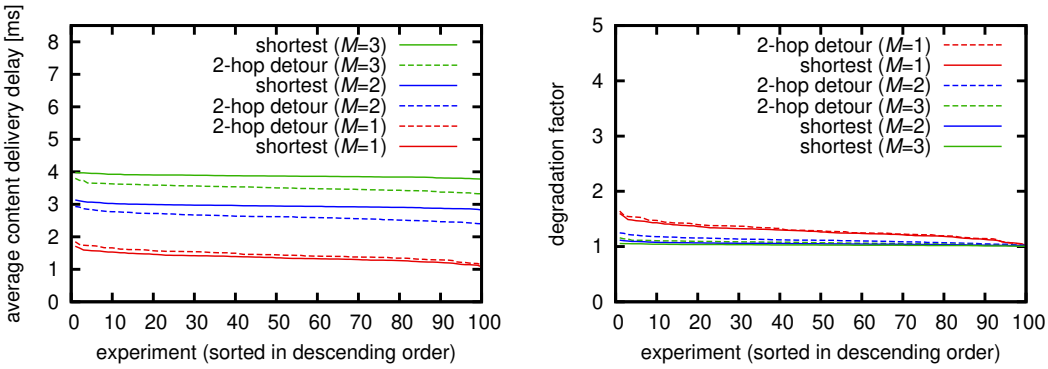
(c) detouring inefficiency

Figure 5.6: Effect of the cache sparseness on average content delivery delays (cluster network)

sizes of routers are homogeneously allocated. Namely, in practice, the shortest-path routing might be better routing in networks whose topology can be arbitrary designed (e.g., data-center network). In contrast, the optimal k-hop detour routing is suitable when the network is unbalanced and variation in cache sizes is large; i.e., a specific router has a giant cache or a part of routers have caches.

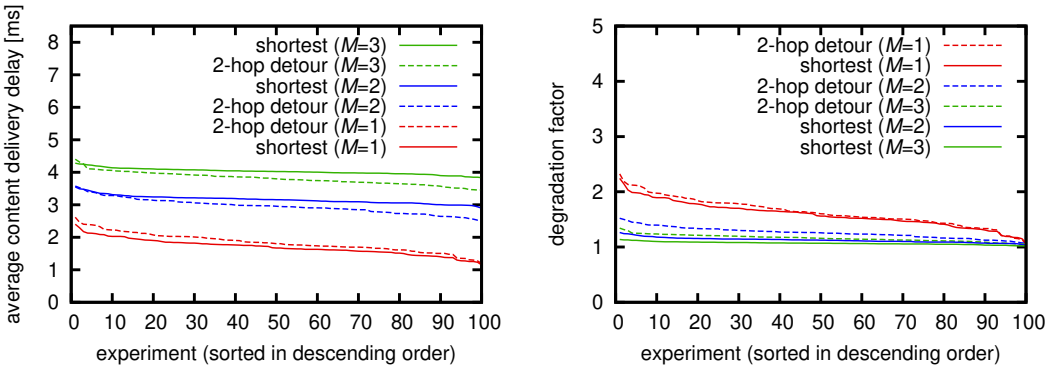
Q2. How robust are the shortest-path routing and the optimal k-hop detour routing against measurement errors in cache hit ratios at routers?

When cache sizes of routers are homogeneously allocated, average content delivery delays with both of the shortest-path routing and the optimal two-hop detour routings increase due to measurement errors in cache hit ratios. In contrast, when cache sizes of routers are heterogeneously allocated, the average content delivery delay with only the two-hop optimal routing is degraded.



(a) average content delivery delay (b) degradation factor

Figure 5.7: Effect of errors in cache hit ratios (triangular network, $\epsilon = 0.5$)



(a) average content delivery delay (b) degradation factor

Figure 5.8: Effect of errors in cache hit ratios (triangular network, $\epsilon = 1.0$)

5.5 Summary

In this chapter, we have investigated the optimality of the shortest-path routing in terms of application-level performance metrics. Specifically, we have compared the average content delivery with the shortest-path routing and that with the optimal two-hop detour routings through a number of experiments. Our findings include that the shortest-path routing is suitable when the network is balanced and cache sizes of routers are homogeneously allocated, and that the optimal k-hop detour routing is suitable when the network is unbalanced and the variation in cache sizes is large. Furthermore, we have investigated the robustness of the shortest-path routing and the optimal k-hop detour routing against measurement errors in cache hit ratios. Consequently, we have shown that the shortest-path routing achieves the almost same

performance regardless of measurement errors in cache hit ratios when cache sizes of routers are heterogeneously allocated.

Chapter 6

Proposal and Evaluation of Lossy Link Detection Mechanism for Content-Centric Networking

6.1 Introduction

In recent years, many studies have investigated networks that mainly focus on transmitting and receiving content (i.e., information-centric networking) rather than on hosts that transmit and receive contents (host-centric networking), such as a conventional TCP/IP network [1, 2]. Content-Centric Networking (CCN) [1] is one of the promising architectures for realizing information-centric networking.

In CCN, a unique identifier is assigned to every content, and content delivery is realized through a request-and-response communication model. When a user (i.e., entity) requests a content, it injects a content request packet called *Interest packet* into a network. The Interest packet is routed among routers based on those routing tables, and delivered to a server (i.e., repository). When the repository receives the Interest

This chapter is a minor version of [52].

packet, it returns a response packet called *Data packet* which corresponds the received Interest packet.

In CCN, routers on a network can cache forwarded contents and reuse data. Specifically, a CCN router has a buffer memory called *Content Store* that caches forwarded Data packets. Subsequently, when this CCN router receives another Interest packet for the same content, it returns the cached content from as a Data packet to the entity instead of forwarding the Interest packet. Consequently, CCN can help reduce the traffic volume transferred through the network as well as the content delivery delay for an entity.

Because of content caching at routers and existence of multiple repositories storing content replicas, content delivery in CCN is classified as an *anycast (any-to-one)* communications rather than a *unicast (one-to-one)* communication [1].

For this reason, in CCN, using the transport protocol developed for IP is difficult [53]. Different from IP, fundamental futures of CCN are, for instance, that routers can cache forwarded Data packets and that routers can aggregate multiple Interest packets requesting an identical content to a single Interest packet.

Challenges in transport protocol design for CCN are *quick and accurate packet loss detection* in the network and *congestion control* for efficient and fair bandwidth sharing among competing entities. A quick and accurate packet loss detection is essential for realizing timely and efficient error recovery from Interest and/or Data packet losses as well as for designing a loss-based congestion mechanism.

In [54], we proposed a packet loss detection mechanism, Interest ACK (ACKnowledgment) and investigated the effectiveness of Interest ACK through several simulations. Interest ACK enables an entity to quickly detect the ocuurence of Interest/Data packet losses using Interest ACKs embedded in a returned Data packet.

The objective of Interst ACK is to rapidly and accurately detect *whether* a content request (i.e., a pair of the Interest packet and its corresponding Data packet) is lost in the network [54]. We further extend the idea of Interest ACK to enable to detect *which* of Interest and Data packets is lost as well as *at which link* the packet was discarded.

More specifically, in this chapter, we propose a lossy link detection mechanism

called LLD-IA (Lossy Link Detection with Interest ACKs), which enables an entity to estimate the link where the packet was discarded in a network. Different from Interest ACK, in LLD-IA, routers along a path as well as the repository update Interest ACKs on successful Interest and Data packets transmissions. Using extended Interest ACKs, an entity can estimate not only the occurrence of packet loss but also the location where the packet was discarded.

Furthermore, through simulation results, we demonstrate that LLD-IA can effectively diagnose faulty links under moderate packet loss ratios.

The reminder of this chapter is organized as follows. First, Section 6.2 summarizes previous works related to transport protocols for CCN. Section 6.3 proposes LLD-IA (Lossy Link Detection with Interest ACKs) and investigates the accuracy of LLD-IA through simulations. Section 6.4 concludes this chapter.

6.2 Related Work

Multiple transport protocols have been proposed for CCN [19-25], and each of them takes a different approach to solve CCN-specific problems.

For example, Interest Control Protocol (ICP) [20] uses Additive Increase and Multiplicative Decrease (AIMD)-based window flow control. The entity sending an Interest packet dynamically adjusts the number of packets (i.e., *window*) that can be sent during its round-trip time. ICP adopts the same loss-based window flow control as the conventional Transmission Control Protocol (TCP). In ICP, losses of an Interest packet or a Data packet are detected by timeout, and AIMD-based window flow control is performed according to the presence and absence of packet loss in the network. In ICP, each entity measures the round-trip time (i.e., the time elapsed between the injection of an Interest packet and the reception of the corresponding Data packet), and it adjusts the timeout timer based on the measured round-trip time. Timeout timer adjustment like that in ICP works effectively in networks with small round-trip time variation. On the other hand, in networks with large round-trip time variation, packet losses can not be detected promptly.

Content-Centric TCP (CCTCP) [23] uses a different timeout mechanism to solve problems caused by the characteristic that content delivery in CCN is performed from multiple sources to a single sink. When contents are returned from multiple repositories, the round-trip time differs for every entity-repository pair. In this case, if the same timeout timer is used for multiple repositories, the round-trip time will vary greatly, and therefore, the timeout timer will take a very large value; this will lead to failure of packet loss detection. In CCTCP, an appropriate timeout timer is determined by using multiple timeout timers for different repositories. Moreover, as pointed out in [55], packet loss detection solely based on the timeout timer might not be sufficient because of difficulties in appropriate configuration of the initial timeout (RTO) value and in accurate measurement of round-trip times.

Another countermeasure for accelerating packet loss detection is an *ECN (Explicit Congestion Notification)* from congesting routers [24, 25]. For instance, the authors of [24] proposed a transport protocol called CHoPCoP, in which an entity adjusts its own window size by quickly detecting the occurence of congestion from ECN notifications. Specifically, in CHoPCoP, when a router in a network forwards a Data packet, it randomly marks the packet according to the current occupancy of its output buffer. The entity that receives the marked Data packet can quickly detect the congestion occurrence in the network and appropriately adjust the window size. Through several simulations, the authors showed that, in a multi-source and multi-path environment, CHoPCoP can achieve high throughput.

6.3 LLD-IA (Lossy Link Detection with Interest ACKs)

Interest ACK enables a requesting entity to detect, at the time of packet loss, which of the Interest packet or the corresponding Data packet was lost in the network.

If a requesting entity could obtain more fine-grained information on packet losses such as the *locations* of the lossy links, it could make more sophisticated decision regarding congestion control and error recovery.

In what follows, by extending Interest ACK, we propose LLD-IA (Lossy Link De-

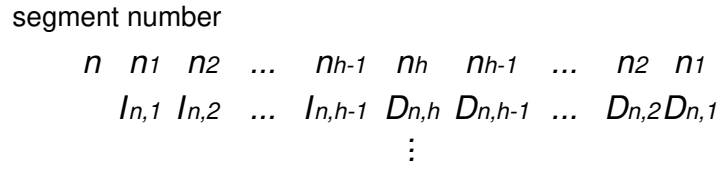


Figure 6.1: Extended Interest ACK used by LLD-IA

tection with Interest ACKs), which is a mechanism for an entity to estimate a link where the packet was discarded in a network. Please refer to [54] for the details of Interest ACK.

6.3.1 Overview

The fundamental idea of LLD-IA is that not only the repository but also all routers along the path update Interest ACKs of all packets exchanged between the entity and the repository. In LLD-IA, when routers and a repository receive Interest/Data packets, each of them overwrites history of Interest/Data packet transmissions. Thus, when the entity receives a Data packet including Interest ACK, it can detect the link where Interest packet or the corresponding Data packet was lost in the network based on Interest ACK stored in the Data packet.

Specifically, in LLD-IA, Interest ACK is extended to include information on Interest and Data packet forwarding at all routers along the path. The extended Interest ACK used by LLD-IA is shown in Fig. 6.1. In the extended Interest ACK, the field $I_{n,i}$ indicates whether i-th router along the path successfully forwarded Interest packet requesting for the n-th segment. Similarly, the field $D_{n,i}$ indicates whether i-th router successfully forwarded Data packet corresponding to the n-th segment.

An example operation of LLD-IA is illustrated in Fig. 6.2. In this figure, the entity sends a series of Interest packets requesting segments 1, ..., 6. Also, the Interest packet 3 is lost at the link from router r_2 to r_3, and the Data packet 4 is lost at the link from router r_2 to r_1. In this case, Interest ACK embedded in the Data packet 6 that the entity received is shown in Fig. 6.3. Utilizing the Interest ACK described in Data packet 6, the entity can detect the loss of the segment 4 (Interest packet 4) at the link from router r_2 to r_3 and that of the segment 6 (Data packet 6) at the link from router

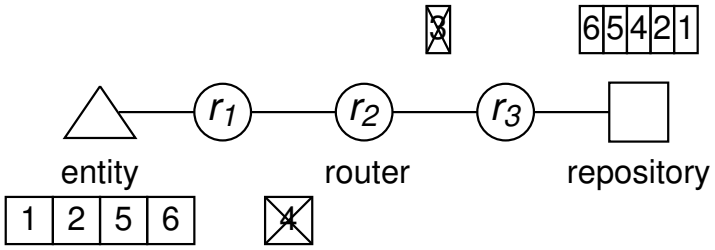


Figure 6.2: An example of packet loss detection with LLD-IA; the entity detects that Interest packet 3 was discarded at the link from router r_2 to r_3, and that Data packet 4 was discarded at the link from router r_2 to r_1.

	r_1	r_2	r_3	repository	r_3	r_2	r_1
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	1	1	0	0	0	0	0
4	1	1	1	1	1	1	0
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

Figure 6.3: An example of packet loss detection with LLD-IA: Interest ACK in the header of Data packet 6

r_2 to r_1.

6.3.2 Operation

In what follows, we explain how extended Interest ACKs are updated by the repository and routers along the path, and how the entity detects lossy links using extended Interest ACKs.

Basically, the entity and the repository operate as in the Interest ACK mechanism [54]. Routers between the entity and the repository perform the following operations when each of them receives a Data packet (i.e., segment). First, the router retrieves the list of the segment numbers from Interest ACK in the Data packet. For each of segments in the list, if the router forwarded an Interest packet requesting for the corresponding segment (i.e., if unsolved PIT entry corresponding to the segment exists), it overwrites the field I of the Interest ACK with 1. Also, if the router successfully forwarded the corresponding segment, it overwrites the field D of the Interest ACK with 1. In contrast, if the router forwarded an Interest packet requesting the seg-

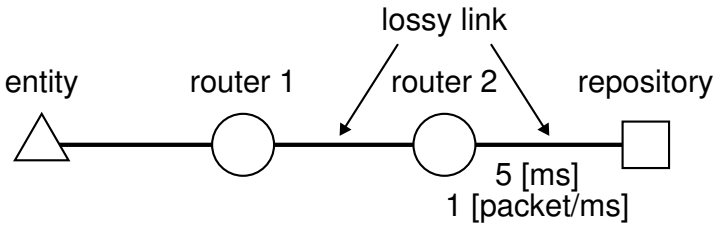


Figure 6.4: Linear network topology used in Section 6.3

ment which is *not* included in the list of segments of the Interest ACK (i.e., if unsolved PIT entry corresponding to the segment exists), it appends a new entry corresponding to the segment and writes 1 in the field I of the Interest ACK.

6.3.3 Evaluation

Through simulation, we investigate how accurately LLD-IA can detect the location of packet losses in a lossy environment.

For examining the fundamental properties of LLD-IA, we intentionally used a rather simple network topology consisting of an entity, two routers, and a repository (see Fig. 6.4). Propagation delays and bandwidths between two routers and between a router and a repository were equally set to 5 [ms] and 1 [packet/ms], respectively. The propagation delay between an entity and a router was set to 0 [ms], and the bandwidth between an entity and a router was set to infinity.

For simulating a lossy environment, packet loss ratios of all links were equally fixed at 0.01. Specifically, in links between router1-router2 and router2-repository, 1% of Interest and Data packets were randomly discarded.

To investigate the effect of content caching in a network on accuracy of LLD-IA, we artificially generated cache hit and miss events at router1 and router2. More specifically, a given fraction of Interest packets were randomly chosen from all arriving Interest packets, and those Interest packets were treated as *cache-hit* Interest packet while others were as *cache-missed* ones.

The accuracy of lossy link detection with LLD-IA was measured by the *link detection accuracy*, which is defined as the ratio of the number of packet losses with successful lossy link detection to the total number of packet losses. In our simulation,

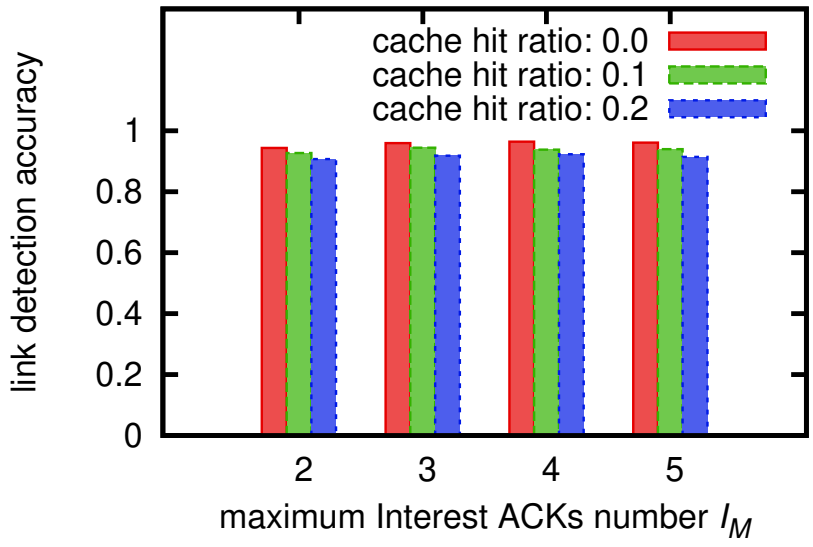


Figure 6.5: Link detection accuracy of LLD-IA

to investigate how the number of Interest ACKs embedded in a Data packet affects the accuracy of LLD-IA, the maximum Interest ACKs number I_M was varied. The number of packets pending detection I_p was set to zero.

We used a packet-level CCN simulator (pccnsim) developed by our research group and performed a simulation for 10 [s], in which the entity operating the AIMD flow control mechanism requests a series of segments consisting a content. For a single parameter setting, simulations were repeated 10 times, and the average of the accuracy was obtained.

Link detection accuracy for different control parameters are shown in Fig. 6.5. In this figure, the cache hit ratio at routers are changed between 0 and 0.2. From these results, one can find that LLD-IA almost detects the location where packet losses are occurred regardless of the cache hit ratio. One can also find that the control parameter I_M of LLD-IA does not highly affect the accuracy of LLD-IA.

In what follows, we discuss the overhead of Interest ACKs, in particular, the size of Interest ACK occupying a Data packet. In LLD-IA, the size of Interest ACK table is given by the product of the number of Interest ACKs that can be stored in a Data packet I_M and the number of routers along a path plus a repository. In the case of

simulation in this section with *naive* representation, the size of Interest ACK table is at most 270 byte, which occupies 18% of the size of a Data packet. However, by adopting compact data representation of segment numbers with encoding, the size of Interest ACK table can be significantly reduced.

Based on the above observations, we can conclude that LLD-IA can accurately detect the location of the lossy links under moderate packet loss ratios.

6.4 Summary

In this chapter, we have proposed a lossy link detection mechanism called LLD-IA by extending Interest ACK. The fundamental idea of LLD-IA is that not only the repository but also all routers along the path update Interest ACKs of all packets exchanged between the entity and the repository. LLD-IA enables the requesting entity to detect the link where Interest packet or the corresponding Data packet was lost in the network. Through simulations, we have showed that LLD-IA can accurately detect the location of the lossy links under moderate packet loss ratios regardless of its control parameters.

Chapter 7

Conclusion

In this chapter, we summarize the research in this thesis, and address future directions.

In Chapter 2, we have investigated the scalability of CCNx, which is an open-source software implementation of CCN , in terms the number of nodes. Specifically, multiple CCNx daemons connected forming CCN networks were executed on a single physical computer, and CCN request generators were executed on the other one. We have measured the total throughput of content deliveries, the packet loss ratio in the network, and the average content delivery time while changing the network size (i.e., the number of CCN routers). Also, we have revealed that the performance bottleneck of CCNx is the Data-chunk digest computation at CCN routers, which consumes approximately 20% of the total CPU time in our experiments. Furthermore, we found that the hardware offloading of the Data-chunk digest computation is effective of improving the total throughput and reducing the content delivery time.

In Chapter 3, we have analyzed the performance of CCN on an arbitrary network topology by utilizing the MCA algorithm, which is an approximation algorithm which analytically calculates cache hit probability in a multi-cache network. We have analytically derived content delivery delay, throughput, and availability in a network comprising multiple routers and multiple repositories. Through several numerical examples, we have shown that the benefits of performance improvement by content caching (i.e., reduction in content delivery delay and improvement in availability)

were more pronounced when the router was closer to the repository in CCN. By comparing analytic results and simulations results, we have also shown the validity of our analysis.

In Chapter 4, we have focused on a large-scale ICN and derived the cache hit probability at each router, the average content delivery delay for each entity, and the average content delivery delay of all entities over a content distribution tree comprised of a single repository, multiple routers, and multiple entities. Furthermore, through several numerical examples, we have investigated the effect of the topology and the size of the content distribution tree and the cache size at routers on the average content delivery delay of all entities. Our findings include that the average content delivery delay of ICNs converges to a constant value if the cache size of routers are not small, which implies high scalability of ICNs in terms of the network size, and that even when the network size would grow indefinitely, the average content delivery delay is upper-bounded by a constant value if routers in the network are provided with a fair amount of content caches.

In Chapter 5, we have investigated the optimality of the shortest-path routing in terms of application-level performance metrics. Specifically, using our mathematical analysis of ICN in Chapter 3, we have compared the average content delivery with the shortest-path routing and that with the optimal two-hop detour routings through a number of experiments. Our findings include that the shortest-path routing is suitable when the network is balanced and cache sizes of routers are homogeneously allocated, and that the optimal k-hop detour routing is suitable when the network is unbalanced and the variation in cache sizes is large.

In Chapter 6, we have proposed a lossy link detection mechanism called LLD-IA by extending Interest ACK. The fundamental idea of LLD-IA is that not only the repository but also all routers along the path update Interest ACKs of all packets exchanged between the entity and the repository. LLD-IA enables the requesting entity to detect the link where Interest packet or the corresponding Data packet was lost in the network. Through simulations, we have showed that LLD-IA can accurately detect the location of the lossy links under moderate packet loss ratios regardless of its

control parameters.

In this thesis, we have revealed the scalability of ICN (in particular, CCN) in terms of the network size using experiments and mathematical analyses. Also, we have tackled to improve the efficiency of ICN by investigating the optimality of the shortest-path routing in ICN and proposing the lossy link detection mechanism for CCN.

This thesis focused on pure ICN networks, however, fundamental ideas of ICNs (e.g., name-based communication and contents caching) can be applied to other network architectures. In particular, it is expected that ICN-based communication paradigm can be introduced to wireless sensor network and DTN (Delay/Disruption Tolerant Networking), which leads to performance improvements in those networks.

As future directions, by extending our mathematical analysis in this thesis, we are planning to analyze the dynamics of contents diffusion of ICNs as a type of cache networks. In this thesis, we have focused on analyzing the performance of ICNs in a steady state, however, we believe that understanding the dynamics of content diffusion in ICNs helps to design more efficient ICN-based network architectures.

Bibliography

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the Fifth International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2009)*, pp. 1–12, Dec. 2009.

[2] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 66–73, July 2014.

[3] G. Xylomenos *et al.*, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, pp. 1024–1049, 2014.

[4] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in *Proceedings of the 21st International Conference on Computer Communications and Networks (ICCCN 2012)*, pp. 1–9, July 2012.

[5] T. Tang, "High performance content centric networking on virtual infrastructure," Master's thesis, University of Toronto, 2013.

[6] P. H. V. Guimarães *et al.*, "Experimenting content-centric networks in the future internet testbed environment," in *Proceedings of the 2013 International Workshop on Cloud Convergence: challenges for future infrastructures and services (WCC 2013)*, pp. 1383–1387, June 2013.

[7] G. Rossini and D. Rossi, "A dive into the caching performance of content centric networking," in *Proceedings of the IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD 2012)*, pp. 105–109, Sept. 2012.

[8] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnSim: an highly scalable CCN simulator," in *Proceedings of the 2013 International Conference on Communications (ICC 2013)*, pp. 2309–2314, June 2013.

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

[10] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proceedings of the 1st IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2012)*, pp. 310–315, Mar. 2012.

[11] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of CCN-caching trees," in *Proceedings of the 10th IFIP-TC6 Networking Conference (NETWORKING 2011)*, pp. 78–91, May 2011.

[12] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking (ICN 2015)*, pp. 69–78, Sept. 2015.

[13] A. Udugama, S. Palipana, and C. Goerg, "Analytical characterisation of multipath content delivery in content centric networks," in *Proceedings of the 2013 Conference on Future Internet Communications (CFIC)*, pp. 1–7, May 2013.

[14] W. Guoqing, H. Tao, L. Jiang, C. Jianya, and L. Yunjie, "Approximate models for CCN data transfer in general topology," *China Communications*, vol. 11, pp. 40–47, July 2014.

[15] V. Ramaswamy, "Modeling and performance analysis of information-centric networks," in *Proceedings of the 25th International Conference on Computer Communication and Networks (ICCCN 2016)*, pp. 1–7, Aug. 2016.

[16] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 2847–2886, May 2016.

[17] V. Sourlas, P. Flegkas, and L. Tassiulas, "Cache-aware routing in information-centric networks," in *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pp. 582–588, May 2013.

[18] V. G. Vassilakis, M. F. AL-Naday, M. J. Reed, and B. A. Alzahrani, "A cache-aware routing scheme for information-centric networks," in *Proceedings of the 9th IEEE International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP 2014)*, pp. 721–726, 2014.

[19] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer issues in information centric networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012)*, pp. 19–24, Aug. 2012.

[20] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *Proceedings of the 1st IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2012)*, pp. 304–309, Mar. 2012.

[21] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012)*, pp. 37–42, Aug. 2012.

[22] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Proceedings of the 2nd IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2013)*, pp. 363–368, Nov. 2013.

[23] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A scalable receiver-driven congestion control protocol for content centric networking," in *Proceedings of the 12th*

IEEE Internatinal Conference on Communications (ICC 2013), pp. 3775–3780, Nov. 2013.

[24] F. Zhang, Y. Zhang, A. Reznik, H. Liu, Q. Qian, and C. Xu, "A transport protocol for content-centric networking with explicit congestion control," in *Proceedings of the 23rd IEEE Internatinal Conference on Computer Communication and Networks (ICCCN 2014)*, pp. 1–8, Aug. 2014.

[25] J. Zhou, Q. Wu, Z. Li, M. A. Kaafar, and G. Xie, "A proactive transport mechanism with explicit congestion notification for NDN," in *Proceedings of IEEE International Conference on Communications (ICC 2015)*, pp. 5242–5247, June 2015.

[26] R. Nakamura and H. Ohsaki, "Performance evaluation and improvement of large-scale content-centric networking," in *Proceedings of the 31st IEEE International Conference on Information Networking (ICOIN 2017)*, pp. 103–108, Jan. 2017.

[27] "Project CCNx." `https://github.com/ProjectCCNx`. [accessed December 7, 2019].

[28] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the First ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011)*, pp. 44–49, Aug. 2011.

[29] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.

[30] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 2–16, Feb. 2004.

[31] R. Nakamura and H. Ohsaki, "Performance analysis of content-centric networking on an arbitrary network topology," *IEICE Transactions on Communications, Special Section on Internet Technologies to Accelerate Smart Society*, pp. 24–34, Jan. 2018.

[32] M. R. Butt, O. Delgado, and M. Coates, "An energy-efficiency assessment of content centric networking (CCN)," in *Proceedings of the 25th IEEE Canadian Conference on Electrical & Computer Engineering (CCECE 2012)*, pp. 1–4, Apr. 2012.

[33] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate models for general cache networks," in *Proceedings of the 29th Conference on Information Communications (INFOCOM 2010)*, pp. 1–9, Mar. 2010.

[34] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proceedings of ACM/SIGMETRICS 1990*, pp. 143–152, May 1990.

[35] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal of Selected Areas in Comuunications*, vol. 29, pp. 1765–1775, Oct. 2011.

[36] R. Nakamura and H. Ohsaki, "On scaling property of information-centric networking," *IEICE Transactions on Communications, Special Section on Enhancing Information Centric Networking Technologies Towards Real-world Infrastructure*, pp. 1804–1812, Sept. 2019.

[37] J. Wang, "A survey of web caching schemes for the internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, pp. 36–46, Oct. 1999.

[38] C. Williamson, "On filter effects in web caching hierarchies," *ACM Transactions on Internet Technology*, vol. 2, pp. 47–77, Feb. 2002.

[39] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1305–1314, Sept. 2002.

[40] E. L. Valentina Martina, Michele Garetto, "A unified approach to the performance analysis of caching systems," in *Proceedings of the 33rd Conference on Information Communications (INFOCOM 2014)*, pp. 2040–2048, Apr. 2014.

[41] X. Xu, C. Feng, T. Zhang, J. Loo, and G. Y. Li, "Caching performance of information centric networking with content request aggregation," in *Proceedings of IEEE International Conference on Communications Workshops (ICC 2018)*, pp. 1–6, May 2018.

[42] Y. Mordjana, M. R. Senouci, and A. Mellouk, "Performance analysis of caching and forwarding strategies in content centric networking," in *Proceedings of IEEE GLOBECOM 2017*, pp. 1–6, Dec. 2017.

[43] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, pp. 315–327, Dec. 2002.

[44] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," in *Proceedings of the 24th International Conference on International Teletraffic Congress (ITC 2012)*, pp. 57–64, Sept. 2012.

[45] Y. Chen, T. Farley, and N. Ye, "QoS requirements of network applications on Internet," *Information Knowledge Systems Management*, vol. 4, pp. 55–76, Jan. 2004.

[46] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012)*, pp. 55–60, Aug. 2012.

[47] K. Yokota, K. Sugiyama, J. Kurihara, and A. Tagami, "RTT-based caching policies to improve user-centric performance in CCN," in *Proceedings of the 30th IEEE International Conference on Advanced Information Networking and Applications (AINA 2016)*, pp. 124–131, May 2016.

[48] T. Mori, K. Hirata, and M. Yamamoto, "Content-oriented probabilistic routing with measured RTT," in *Proceedings of IEEE Communications Society Communications Quality and Reliability Workshop (CQR 2016)*, May 2016.

[49] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," in *Proceedings of the 1st ACM Conference on Information-Centric Networking (ICN 2014)*, pp. 127–136, Sept. 2014.

[50] S. K. Fayazbakhsh *et al.*, "Less pain, most of the gain: Incrementally deployable ICN," in *Proceedings of ACM SIGCOMM 2013*, pp. 147–158, Aug. 2013.

[51] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM SIGCOMM Computer Communication Review*, vol. 1, pp. 1–28, May 2016.

[52] R. Nakamura and H. Ohsaki, "A fast packet loss detection mechanism for content-centric networking," *IEICE Transactions on Communications, Special Section on Enhancing Information Centric Networking Technologies Towards Real-world Infrastructure*, pp. 1842–1852, Sept. 2019.

[53] R. Yongmao, L. Jun, S. Shanshan, L. Lingling, W. Guodong, and Z. Beichuan, "Congestion control in named data networking – a survey," *Computer Communications*, vol. 86, pp. 1–11, July 2016.

[54] T. Yabuuchi, R. Nakamura, and H. Ohsaki, "Interest ACK: A fast packet loss detection mechanism for content-centric networking," in *Proceedings of the 5th International Workshop on Architecture, Design, Deployment and Management of Networks and Applications (ADMNET 2017)*, pp. 13–18, July 2017.

[55] L. Zhang, "Why TCP timers don't work well," *ACM SIGCOMM Computer Communication Review*, vol. 16, pp. 397–405, Aug. 1986.

List of Publications

Journal Papers

1. R. Nakamura and H. Ohsaki, "Performance Analysis of Content-Centric Networking on an Arbitrary Network Topology", *IEICE Transactions on Communications, Special Section on Internet Technologies to Accelerate Smart Society*, vol. E101-B, no. 1, pp. 24-34, Jan. 2018.

2. R. Nakamura and H. Ohsaki, "On Scaling Property of Information-Centric Networking", *IEICE Transactions on Communications, Special Section on Enhancing Information Centric Networking Technologies Towards Real-world Infrastructure*, vol. E102-B, no. 9, pp. 1804-1812, Sep. 2019.

3. R. Nakamura and H. Ohsaki, "A Fast Packet Loss Detection Mechanism for Content-Centric Networking", *IEICE Transactions on Communications, Special Section on Enhancing Information Centric Networking Technologies Towards Real-world Infrastructure*, vol. E102-B, no. 9, pp. 1842-1852, Sep. 2019.

4. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "Revisiting the Robustness of Complex Networks against Random Node Removal", *Journal of Information Processing*, pp. 643-649, Sep. 2019.

Conference Papers

1. R. Nakamura and H. Ohsaki, "Performance Evaluation of an Open-Source Implementation of Content-Centric Networking", *Proceedings of the IEEE Signature*

Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2015), pp. 621-624, Jul. 2015.

2. R. Nakamura and H. Ohsaki, "Performance Analysis of CCN on Arbitrary Network Topology", *Proceedings of IEEE Communications Society Communications Quality and Reliability Workshop (CQR 2016)*, pp. 1-6, May. 2016.

3. R. Nakamura and H. Ohsaki, "Performance Comparison of Shortest-Path Routing and Optimal Detour Routing in Content-Centric Networking", *Proceedings of the 40th IEEE Signature Conference on Computers, Software, and Applications (Doctoral Symposium) (COMPSAC 2016)*, pp. 494-495, Jun. 2016.

4. R. Nakamura and H. Ohsaki, "Performance Evaluation and Improvement of Large-Scale Content-Centric Networking", *Proceedings of the 31st IEEE International Conference on Information Networking (ICOIN 2017)*, pp. 103-108, Jan. 2017.

5. T. Yabuuchi, R. Nakamura and H. Ohsaki, "Interest ACK: A Fast Packet Loss Detection Mechanism for Content-Centric Networking", *Proceedings of the 5th International Workshop on Architecture, Design, Deployment and Management of Networks and Applications (ADMNET 2017)*, pp. 13-18, Jul. 2017.

6. R. Nakamura and H. Ohsaki, "On the Effect of Scale-Free Structure of Network Topology on Performance of Content-Centric Networking", *Proceedings of the 41st IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2017)*, pp. 686-689, Jul. 2017.

7. T. Yabuuchi, R. Nakamura and H. Ohsaki, "A Fluid-Based Model of Transport Protocol in Content-Centric Networking", *Proceedings of the 41st IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2017)*, pp. 690-693, Jul. 2017.

8. R. Matsuo, R. Nakamura and H. Ohsaki, "A Solution for Minimum Link Flow Problem with Sparse Modeling", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (COMPSAC 2018)*, pp. 920-925, Jul. 2018.

9. Y. Yasuda, R. Nakamura and H. Ohsaki, "A Probabilistic Interest Packet Aggregation for Content-Centric Networking", *Proceedings of the 2nd IEEE International Workshop on Future Internet Technologies (IWFIT 2018)*, pp. 783-788, Jul. 2018.

10. R. Maegawa, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on Emulating Automotive IP Networks using Network Virtualization", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 943-946, Jul. 2018.

11. R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on Sparse-Modeling based Approach for Betweenness Centrality Estimation", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 973-976, Jul. 2018.

12. C. Minamiguchi, N. Kawabata, R. Nakamura and H. Ohsaki, "A Study on Comparative Analysis of End-to-End Routing and Opportunistic Routing", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 955-958, Jul. 2018.

13. Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on the Impact of Delayed Packet Forwarding in Content-Centric Networking", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 970-972, Jul. 2018.

14. K. Yamashita, R. Nakamura and H. Ohsaki, "A Study on Robustness of Complex Networks against Random Node Removals", *Proceedings of the 42nd IEEE Signature Conference on Computers, Software, and Applications (Student Research Symposium) (COMPSAC 2018)*, pp. 966-969, Jul. 2018.

15. R. Matsuo, R. Nakamura and H. Ohsaki, "Sparse Representation of Network Topology with K-SVD Algorithm", *Proceedings of the 43rd IEEE Signature Conference on Computers, Software, and Applications (COMPSAC 2019)*, pp. 291-298, Jul. 2019.

16. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "On the Predictability of Network Robustness from Spectral Measures", *Proceedings of the 7th IEEE International Workshop on Architecture, Design, Deployment and Management of Networks and Applications (ADMNET 2019)*, pp. 24-29, Jul. 2019.

17. Y. Yasuda, R. Nakamura and H. Ohsaki, "Delayed Packet Forwarding for Information-Centric Networking", *Proceedings of the 14th International Conference on Future Internet Technologies (CFI 2019)*, pp. 8:1-8:6, Aug. 2019.

18. R. Sakaguchi, D. Matsui, R. Nakamura and H. Ohsaki, "Analysis of Constrained Random WayPoint Mobility Model on Graph", *Proceedings of the 34th IEEE International Conference on Information Networking (ICOIN 2020) (to appear)*, Jan. 2020.

19. T. Suzuki, Y. Yasuda, R. Nakamura and H. Ohsaki, "On Estimating Communication Delays Using Graph Convolutional Networks with Semi-Supervised Learning", *Proceedings of the 34th IEEE International Conference on Information Networking (ICOIN 2020) (to appear)*, Jan. 2020.

20. S. Yamamoto, R. Nakamura and H. Ohsaki, "Fluid-Based Modeling of Transport Protocol for Information-Centric Networking", *Proceedings of the 34th IEEE International Conference on Information Networking (ICOIN 2020) (to appear)*, Jan. 2020.

21. C. Minamiguchi, R. Nakamura and H. Ohsaki, "Comparative Analysis of Content Routing Strategies in Information-Centric Delay-Tolerant Networking", *Proceedings of the 34th IEEE International Conference on Information Networking (ICOIN 2020) (to appear)*, Jan. 2020.

Non-Refereed Technical Papers

1. R. Nakamura and H. Ohsaki, "A Study on the Scalability of Open-Source Implementation of Content-Centric Networking", *Proceedings of the IEICE Society Conference (B-16-6)*, pp. 356, Sep. 2014 (*in Japanese*).

2. Y. Noro, R. Nakamura and H. Ohsaki, "Performance Analysis of CCN on an Arbitrary Network Topology", *Technical Report of IEICE (IA2014-99)*, pp. 91-96, Mar. 2015 (*in Japanese*).

3. R. Nakamura and H. Ohsaki, "Scalability Analysis of Large-scale CCN Network with Network Virtualization", *Technical Report of IEICE (IA2015-24)*, pp. 29-34, Aug. 2015 (*in Japanese*).

4. K. Tanimoto, R. Nakamura and H. Ohsaki, "A Study on the Optimality of Shortest-Path Routing in CCN", *Proceedings of the IEICE Society Conference (B-16-5)*, pp. 330, Sep. 2015 (*in Japanese*).

5. T. Yabuuchi, R. Nakamura and H. Ohsaki, "A Proposal of an Interest-ACK-based Transport Protocol for CCN", *Proceedings of the IEICE Society Conference (B-16-6)*, pp. 331, Sep. 2015 (*in Japanese*).

6. R. Nakamura and H. Ohsaki, "A Study on the Scalability of Large-Scale CCN Networks", *Proceedings of the IEICE General Conference (BS-6-6)*, pp. S-126, Mar. 2016 (*in Japanese*).

7. S. Wada, R. Nakamura and H. Ohsaki, "A Study on the Effect of Scale-Free Structure of Network Topology on Delivery Delay of Content-Centric Networking", *Proceedings of the IEICE Society Conference (B-16-7)*, pp. 358, Sep. 2016 (*in Japanese*).

8. Y. Yagi, R. Nakamura and H. Ohsaki, "A Study on Modeling the Dynamics of Transport Protocol in Content-Centric Networking", *Proceedings of the IEICE Society Conference (B-16-8)*, pp. 359, Sep. 2016 (*in Japanese*).

9. T. Yabuuchi, R. Nakamura and H. Ohsaki, "Proposal and Evaluation of Packet Loss Detection Mechanism for CCN", *Technical Report of IEICE (IA2016-61)*, pp. 29-34, Nov. 2016 (*in Japanese*).

10. R. Nakamura and H. Ohsaki, "Performance Comparison of Shortest-Path Routing and Detour Routing for Content-Centric Networking", *Technical Report of*

IEICE (IA2016-28), pp. 13-18, Nov. 2016.

11. R. Nakamura and H. Ohsaki, "A Study on Approximate Analysis of Probabilistic Caching Algorithm", *Proceedings of the IEICE General Conference (BS-5-9)*, pp. S-124, Mar. 2017 (*in Japanese*).

12. Y. Yagi, T. Yabuuchi, R. Nakamura and H. Ohsaki, "On Validity of Fluid-based Model of Transport Protocol in Content-Centric Networking", *Proceedings of the IEICE General Conference (BS-5-11)*, Mar. 2017 (*in Japanese*).

13. S. Wada, R. Nakamura and H. Ohsaki, "A Study on the Effect of Scale-Free Property of Network Topology on CCN Throughput", *Proceedings of the IEICE General Conference (BS-5-13)*, Mar. 2017 (*in Japanese*).

14. S. Wada, R. Nakamura and H. Ohsaki, "On the Effect of Scale-Free Structure of Network Topology on Performance of Content-Centric Networking", *Technical Report of IEICE (IA2016-103)*, pp. 85-89, Mar. 2017 (*in Japanese*).

15. Y. Yagi, T. Yabuuchi, R. Nakamura and H. Ohsaki, "A Fluid-based Model of Transport Protocol in Content-Centric Networking", *Technical Report of IEICE (IA2016-104)*, pp. 79-84, Mar. 2017 (*in Japanese*).

16. R. Matsuo, R. Nakamura and H. Ohsaki, "A Solution for Minimum Link Flow Problem with Sparse Modeling", *Technical Report of IEICE (IA2017-10)*, pp. 53-58, Jun. 2017 (*in Japanese*).

17. R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on a Solution for Minimum Link Flow Problem with Sparse Modeling", *Proceedings of the IEICE Society Conference (BS-3-4)*, pp. S-5, Sep. 2017 (*in Japanese*).

18. Y. Yasuda, T. Yabuuchi, R. Nakamura and H. Ohsaki, "A Proposal of Probabilistic Interest Packet Aggregation Method for Content-Centric Networking", *Proceedings of the IEICE Society Conference (BS-4-7)*, pp. 19, Sep. 2017 (*in Japanese*).

19. M. Kawakami, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on Emulating Automotive Ethernet Environment with Network Virtualization", *Proceedings of the IEICE Society Conference (B-16-4)*, pp. 258, Sep. 2017 (*in Japanese*).

20. N. Niimi, R. Nakamura and H. Ohsaki, "A Study on Modeling Frame Aggregation in Large-Scale IEEE 802.15.4 Wireless Sensor Networks", *Proceedings of the IEICE Society Conference (B-16-5)*, pp. 259, Sep. 2017 (*in Japanese*).

21. R. Nakamura and H. Ohsaki, "Approximate Analysis of Average Content Delivery Delay in Large-Scale CCN Networks", *Technical Report of IEICE (IA2017-14)*, pp. 13-18, Aug. 2017 (*in Japanese*).

22. Y. Yasuda, T. Yabuuchi, R. Nakamura and H. Ohsaki, "Proposal and Evaluation of Probabilistic Interest Packet Aggregation Method for Content-Centric Networking", *Technical Report of IEICE (IA2017-19)*, pp. 1-6, Oct. 2017 (*in Japanese*).

23. N. Niimi, R. Nakamura and H. Ohsaki, "Modeling Frame Aggregation in Large-Scale IEEE 802.15.4 Wireless Sensor Networks", *Technical Report of IEICE (IA2017-33)*, pp. 19-22, Nov. 2017.

24. R. Matsuo, R. Nakamura and H. Ohsaki, "A Solution of Minimum Link Flow Problem with Sparse Modeling — Formulation and Preliminary Results —", *Technical Report of IEICE (IA2017-34)*, pp. 23-26, Nov. 2017.

25. T. Yabuuchi, R. Nakamura and H. Ohsaki, "Modeling Large-Scale Content-Centric Networking using Fluid Approximation", *Technical Report of IEICE (IA2017-29)*, pp. 37-42, Nov. 2017 (*in Japanese*).

26. M. Kawakami, R. Nakamura, Y. Yamasaki and H. Ohsaki, "On Emulating Automotive IP Network with Network Virtualization", *Technical Report of IEICE (IA2017-60)*, pp. 43-48, Dec. 2017 (*in Japanese*).

27. T. Yabuuchi, R. Nakamura and H. Ohsaki, "Fluid-Based Modeling of Transport Protocol in Large-Scale Content-Centric Networking", *Technical Report of IEICE (IN2017-131)*, pp. 249-254, Mar. 2018.

28. R. Nakamura and H. Ohsaki, "Evaluation and Application of Packet Loss Detection Mechanism for Content-Centric Networking", *Technical Report of IEICE (IA2018-79)*, pp. 121-126, Mar. 2018 (*in Japanese*).

29. R. Nakamura and H. Ohsaki, "Approximate Analysis of Average Content Delivery Delay in Large-Scale CCN Networks", *Proceedings of the IEICE General Conference (BS-2-15)*, pp. S-28-S-29, Mar. 2018.

30. Y. Yasuda, T. Yabuuchi, R. Nakamura and H. Ohsaki, "A Study on Probabilistic Interest Packet Aggregation Method for Content-Centric Networking", *Proceedings of the IEICE General Conference (BS-6-5)*, pp. S-118-S-119, Mar. 2018 (*in Japanese*).

31. R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on a Solution for Constrained Minimum Link Flow Problem with Sparse Modeling", *Proceedings of the IEICE General Conference (BS-6-6)*, pp. S-120-S-121, Mar. 2018 (*in Japanese*).

32. R. Matsuo, R. Nakamura and H. Ohsaki, "Proposal of Sparse-Modeling based Approach for Betweenness Centrality Estimation", *Technical Report of IEICE (IA2018-10)*, pp. 61-66, Jun. 2018.

33. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "Robustness of Complex Networks against Random Node Removals", *Technical Report of IEICE (IA2018-19)*, pp. 23-28, Sep. 2018.

34. C. Minamiguchi, N. Kawabata, R. Nakamura, Y. Yamasaki and H. Ohsaki, "Comparative Analysis of End-to-End Routing and Opportunistic Routing", *Technical Report of IEICE (IA2018-26)*, pp. 63-68, Sep. 2018.

35. R. Sakaguchi, D. Matsui, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on the Effect of Node Mobility on Hitting Time on a Graph", *Proceedings of the IEICE Society Conference (B-16-12)*, pp. 278, Sep. 2018 (*in Japanese*).

36. R. Murakami, R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on the Effectiveness of Dictionary Learning with K-SVD Algorithm for Network Topolo-

gies", *Proceedings of the IEICE Society Conference (B-16-13)*, pp. 279, Sep. 2018 (*in Japanese*).

37. R. Maegawa, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on Automotive IP Network Emulation with Network Virtualization", *Proceedings of the IEICE Society Conference (BS-7-21)*, pp. S-64-S-65, Sep. 2018.

38. R. Matsuo, R. Nakamura and H. Ohsaki, "A Proposal of Sparse-Modeling based Approach for Betweenness Centrality Estimation", *Proceedings of the IEICE Society Conference (BS-7-1)*, pp. S-28-S-29, Sep. 2018.

39. C. Minamiguchi, N. Kawabata, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on Comparative Analysis of End-to-End Routing and Opportunistic Routing", *Proceedings of the IEICE Society Conference (BS-7-19)*, pp. 60-61, Sep. 2018.

40. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on Robustness of Complex Networks against Non-Adversely Node Removals", *Proceedings of the IEICE Society Conference (BS-7-3)*, pp. S-31-S-32, Sep. 2018.

41. Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on the Performance of Delayed Packet Forwarding in Content-Centric Networking", *Proceedings of the IEICE Society Conference (BS-7-23)*, pp. 68-69, Sep. 2018.

42. Y. Yasuda, R. Nakamura and H. Ohsaki, "Characteristic Analysis of Delayed Packet Forwarding in Content-Centric Networking", *Technical Report of IEICE (IA2018-32)*, pp. 1-4, Oct. 2018.

43. R. Murakami, R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on Sparse Representation of Network Topology with K-SVD Algorithm", *Technical Report of IEICE (IA2018-79)*, pp. 351-356, Mar. 2019 (*in Japanese*).

44. Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on the Effectiveness of Delayed Packet Forwarding in Content-Centric Networking", *Proceedings of the IEICE General Conference (BS-4-8)*, pp. S-23-S-24, Mar. 2019.

45. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on Network Topology with Robustness against Random Node Removals", *Proceedings of the IEICE General Conference (BS-7-8)*, pp. S-158, Mar. 2019 (*in Japanese*).

46. R. Sakaguchi, D. Matsui, R. Nakamura, Y. Yamasaki and H. Ohsaki, "A Study on the Effect of Bias of Transition Probability in Randomwalk on Average First Hitting Time on a Graph", *Proceedings of the IEICE General Conference (BS-7-9)*, pp. S-159, Mar. 2019 (*in Japanese*).

47. C. Minamiguchi, N. Kawabata, R. Nakamura and H. Ohsaki, "A Study on the Average Content Delivery Delay in Information-Centric Delay-Tolerant Networking", *Proceedings of the IEICE General Conference (BS-7-10)*, pp. S-160-S-161, Mar. 2019 (*in Japanese*).

48. R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on Effective Solution for Minimum Link Flow Problem", *Proceedings of the IEICE General Conference (BS-7-12)*, pp. S-164-S-165, Mar. 2019.

49. R. Nakamura, R. Sakaguchi, K. Yamashita, D. Matsui and H. Ohsaki, "On the Impact of Network Topology on Information Search, Delivery, and Diffusion", *Technical Report of IEICE (PN2019-4), no. 16*, pp. 17-24, Apr. 2019 (*in Japanese*).

50. T. Suzuki, Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on Estimating Communication Delays using Graph Convolutional Networks with Semi-Supervised Learning", *Technical Report of IEICE (IA2019-10)*, pp. 1-6, Sep. 2019.

51. K. Yamashita, Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on the Predictability of Network Robustness from Spectral Measures", *Technical Report of IEICE (IA2019-15)*, pp. 21-26, Sep. 2019.

52. R. Sakaguchi, D. Matsui, R. Nakamura and H. Ohsaki, "Characteristic Analysis of Constrained Random WayPoint Mobility Model on Graph", *Technical Report of IEICE (IA2019-16)*, pp. 27-32, Sep. 2019.

53. T. Suzuki, Y. Yasuda, R. Nakamura and H. Ohsaki, "A Study on Communication Delay Estimation using Graph Convolutional Networks with Semi-Supervised Learning", *Proceedings of the IEICE Society Conference (B-16-4)*, pp. 296, Sep. 2019 (*in Japanese*).

54. J. Hagikura, R. Nakamura and H. Ohsaki, "A Study on the Optimal Cache Allocation in Information-Centric Networking", *Proceedings of the IEICE Society Conference (B-16-1)*, pp. 293, Sep. 2019 (*in Japanese*).

55. R. Nakamura, T. Yabuuchi and H. Ohsaki, "A Study on a Fast Packet Loss Detection Mechanism for Content-Centric Networking", *Proceedings of the IEICE Society Conference (BS-4-21)*, pp. S-48-S-49, Sep. 2019.

56. S. Yamamoto, R. Nakamura and H. Ohsaki, "A Study on Dynamical Modeling of Transport Protocol in Information-Centric Networking", *Proceedings of the IEICE Society Conference (BS-4-22)*, pp. S-50-S-51, Sep. 2019.

57. R. Matsuo, R. Nakamura and H. Ohsaki, "A Study on Sparse Representation of Network Topology with K-SVD Algorithm", *Proceedings of the IEICE Society Conference (BS-4-23)*, pp. S-64-S-65, Sep. 2019.