Popularity-driven Caching Strategy for Dynamic Adaptive Streaming over Information-Centric Networks

by

Wenjie Li

A thesis submitted to the

School of Computing

in conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

July 2015

Abstract

The growing demand for video streaming is straining the current Internet, and mandating a novel approach to future Internet paradigms. The advent of Information-Centric Networks (ICN) promises a novel architecture for addressing this exponential growth in data-intensive services, of which video streaming is projected to dominate (in traffic size).

In this thesis, I present a novel strategy in ICN for adaptive caching of variable video contents tailored to different sizes and bit rates. My objective is to achieve optimal video caching to reduce access time for the maximal requested bit rate for every user. At its core, my approach capitalizes on a rigorous delay analysis and potentiates maximal serviceability for each user. I incorporate predictors for requested video objects based on a popularity index (Zipf distribution). In my proposed model, named *DASCache*, I present queuing analysis for Round-Trip Time (RTT) of cached objects, providing a cap on expected delay in accessing video content. In *DASCache*, I present a Binary Integer Programming (BIP) formulation for the cache assignment problem, which operates in rounds based on changes in content requests and popularity scores. *DASCache* reacts to changes in network dynamics that impact bit rate choices by heterogeneous users and enables users to stream videos, maximizing Quality of Experience (QoE). To evaluate the performance of *DASCache*, in contrast to current benchmarks in video caching, I present an elaborate performance evaluation carried out on ndnSIM, over NS-3.

Acknowledgments

I am using this opportunity to express my gratitude to everyone who provided invaluable suggestions and support throughout my research.

I would like to express my warm thanks to my two supervisors, Dr. Hossam S. Hassanein and Dr. Sharief M.A. Oteafy, for their constructive criticism and aspiring guidance through my research.

I would also like to thank Abdulmonem Rashwan, Ramy Atawia, Anas Mahmoud, Hesham Farahat, Lantian Guo and all my friends in Telecommunications Research Lab (TRL) for their friendship, support and encouragement. Special thanks go to Dr. Xianrong Zheng for sharing his illuminating views on a number of issues and to Basia Palmer for the help of proofreading.

Financial support from the Collaborative Research and Training Experience (CREATE) program offered by the Natural Sciences and Engineering Research Council of Canada (NSERC) is greatly appreciated.

Last but not least, thank my parents and family for their endless love and encouragement through my entire life.

Contents

List of Tables

List of Figures

List of Acronyms

CCN	Content-Centric Network.
CDN	Content Distribution Network.
CR	Content Router.
CS	Content Store.
DASH	Dynamic Adaptive Streaming over HTTP.
DNS	Domain Name System.
FIB	Forwarding Information Base.
ICN	Information-Centric Network.
LFU	Least Frequently Used.
LRU	Least Recently Used.
MPD	Media Preparation Description.
NS-3	Network Simulator 3.
OSI	Open Systems Interconnection.
P2P	Peer-to-Peer Network.

PDU	Protocol Data Unit.
PIT	Pending Interest Table.
QoE	Quality of Experience.
QoS	Quality of Service.
RTT	Round-Trip Time.
SDU	Service Data Unit.
URI	Uniform Resource Identifier.

Chapter 1

Introduction

The dynamics of the Internet, which have scaled adequately over the past twenty years, are currently faltering under the projected growth of data demands. With the rise of mobile computing, especially the advent of smart handheld devices, people have more access to the Internet than ever. The dream of having ‘Information at my fingertips any time, any place’ is no longer a dream at all. Thus, the generated data traffic has increased at an inconceivable speed and is exhausting network resources, such as available bandwidth, IP address, etc. In 2013, statistics showed the global IP traffic per month was 51,168 PB; that amount is going to nearly double by 2016, reaching 91,260 PB [12]. More importantly, multimedia data is expected to increase to a considerable percentage. It is projected that, by 2016, video streaming will amount to more than 55% of overall data traffic [13] in conservative estimates.

1.1 Motivation

The current host-centric Internet architecture fails to confront the challenge under this intensive video delivery reality. Every time a user is interested in some video content, an independent communication tunnel between the user's device and the server must be established at the very beginning. This mechanism will add up all the users' data demands on the server side, resulting in heavy processing burden. With increased number of users sharing the common links to the server, once limited network resources on these links are exhausted, the consequent network congestion and huge access delay then degrades the users' experience.

To remedy the aforementioned problems, one direct way is to make video content providers put more investment in upgrading hardware and network infrastructure. However, another approach is to build an application layer overlay network, *e.g.*, Peer-to-Peer Network (P2P) and Content Distribution Network (CDN).

By having symmetry in roles where a client could also be a server, P2P overlay network [3, 30] relieves the burden of satisfying increased data demand on the content server. With the assumption that all clients in P2P network are altruistic, data can be retrieved from not only the origin server, but other clients simultaneously, which thereby improves the data delivery efficiency.

CDN [32, 31] is another typical example of an overlay network which utilizes surrogate servers to retain copies of identical contents, offloading the traffic from the origin server. Users' requests will be directed to the nearest surrogate server instead of traveling through a congested routing path to the origin server. CDN helps to promote the information availability and benefits users by reducing access latency.

In addition, handling the increased multimedia traffic can also be done via a client-driven method, called Dynamic Adaptive Streaming over HTTP (DASH) [37]. DASH could provide time-shift control on media requests according to varying network conditions and achieves the highest quality of streaming as possible. The fundamental feature of DASH is to encode a media stream with multiple alternative bit rates and chop it into a sequence of small HTTP-based file segments of the same length while each one contains a short interval of playback time. These segments are provided on ordinary servers; for each segment, DASH client will adapt automatically between versions of video and choose the best possible quality to download. As shown in Figure 1.1, such adaptation decision is made according to real-time bandwidth, without incurring stalls in the playback. Therefore, based on the individual network condition, a user now could be served with the most suitable bit rate which guarantees the best viewing experience.

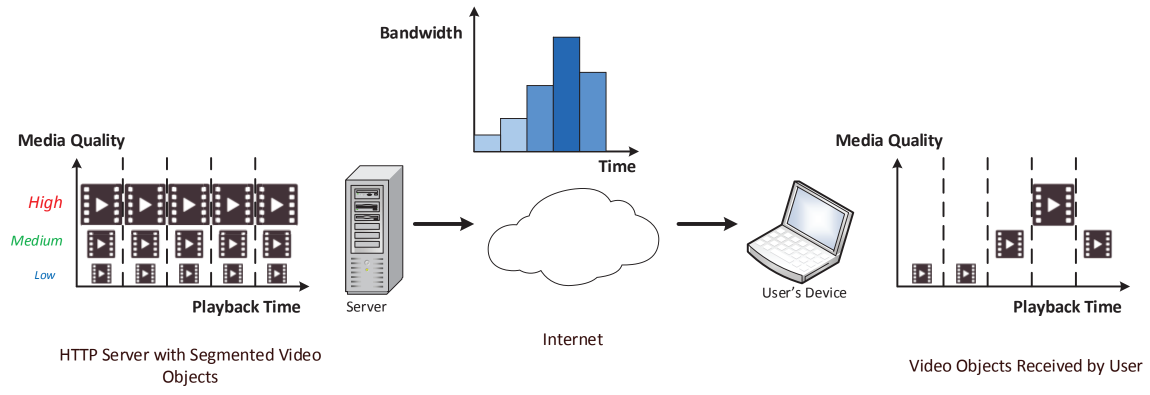


Figure 1.1: Dynamic Adaptive Streaming over HTTP

Even though CDN and P2P provide overlays over the Internet to disperse the burden on the server in order to handle large amounts of content demand, they both have their own issues which have limited progress for years. For example, there is always debate on heavy bandwidth usage, copyright infringement and security, which slow the application of P2P. As to CDN, replicas must be synchronized all the time via either a pull-based or push-based cooperation scheme between the origin and surrogate server, which makes managing CDN not only costly but constrained by the scale of topology.

DASH is a creative way to adjust users' requests according to network load so that it deduces the influence of network performance on users but intrinsically, DASH never solves the fundamental issue of the resource strain on the current Internet.

Therefore, all these approaches which are based on current host-centric architecture, do not give an ultimate solution. They basically are functionality patches to upgrade the Internet, fulfilling tasks which were not designed to be addressed before.

1.2 Objectives

In recent years, a rising concern in the paradigms governing the Internet has resulted in the emergence of a new architecture, Information-Centric Network (ICN) [43]. That is proposed as the next generation of the Internet designed to intrinsically handle large content and distribute it via network layer primitives which essentially unravel the vicious cycle of 'patching over patches' on the current Internet architecture. In ICNs, rather than adopting the client-server approach of the current host-centric Internet, the core premise is adapting to content and catering to both consumers and producers.

There are several appealing features of ICN, such as: time and space decoupling, naming over network layers, multi-path routing, all which make optimization over handling large content dissemination possible. More importantly, ICN exploits caching as a networking primitive which solves the efficiency issue of content delivery over the current Internet thoroughly.

The challenge of catering to video content, as a dominant type of traffic, is also instrumental in the development of ICNs. To this end, some researchers attempted data manipulation such as transcoding to address video caching in ICNs. Yet, the problem of addressing heterogeneous caching over variable chunks of different bit rates, without incurring in-network data processing, remains unsolved.

In this thesis, inspired by dynamic adaptive streaming, I present a novel model, *DASCache*, to handle heterogeneous video content caching in ICNs. My objective is simply to minimize the average retrieval time for requested videos, to facilitate rapid streaming over ICNs which improves Quality of Experience (QoE) for all users. My approach addresses variable bit rates and content sizes, to mimic realistic scenarios where different users would demand the best possible bit rates given the heterogeneity of their devices and link conditions. Utilizing *DASCache*, users with varying network conditions will experience higher throughput and are thus able to switch to videos with better resolution, achieving the best experience possible.

1.3 Organization of Thesis

The remainder of the thesis is organized as follows. I proceed with a complete literature overview, including recent research on caching schemes in ICNs and predecessor work on dynamic adaptive streaming in Chapter 2. Chapter 3 presents my system model of *DASCache* management strategy. My experiment setup and performance evaluation results are detailed in Chapter 4 and I conclude in Chapter 5 with my final remarks and proposals for future work in this viable direction.

Chapter 2

Background and Related Work

To explain my contributions, it is necessary to detail background from the areas of ICN and DASH. In this chapter, I reveal the fundamental features of ICN (especially ICN Caching) and the working process of DASH which are the premise of my work.

2.1 Information-Centric Networks

The problem with the current Internet arises from architecture which was previously projected to handle communication at a time when only rare resources need to be shared along a long-distance link [43]. It was impossible for the designers to predict the data traffic burst of today's Internet at that given time. To meet the requirements that the Internet was not designed for, patches to current architecture have to be applied (*e.g.*, CDN, P2P and Mobile IP). Nevertheless, more and more patches could only add complexity to the Internet, which thereby degrades the performance further.

Information-Centric Network emerges which takes into consideration both the current and future need of the Internet. Driven by the fact that users are more interested in receiving information, rather than where it is located, ICN totally abandons the host-centric model and achieves appealing features as follows.

2.1.1 Decoupling in ICN

Unlike the current Internet where the user must provide a specific location to denote from which place the information could be retrieved, ICN addresses this issue by employing publish-subscribe (*Pub-Sub*) model [2, 14]. Information sources advertise the contents through *Publish* to the network and then users subscribe to what content they need. Inside ICN, content notification service is responsible for matching the subscriptions to corresponding publications. Each terminal contains a *name resolution*, which plays the same role of Domain Name System (DNS) server in the current Internet, and a *named-based routing* responsible for data delivery, as shown in Figure 2.1. These two features will be explicated in detail in Section 2.1.2. Based on this model, the user does not need to know the exact destination (*i.e.* IP address of the server) but instead consults name resolution and then dispatches the subscription. The routing of subscription and publication are entirely left to ICN. Publishers will not be aware of how many subscribers are interested in the content, neither do subscribers, which thereby ensures space decoupling in ICN.

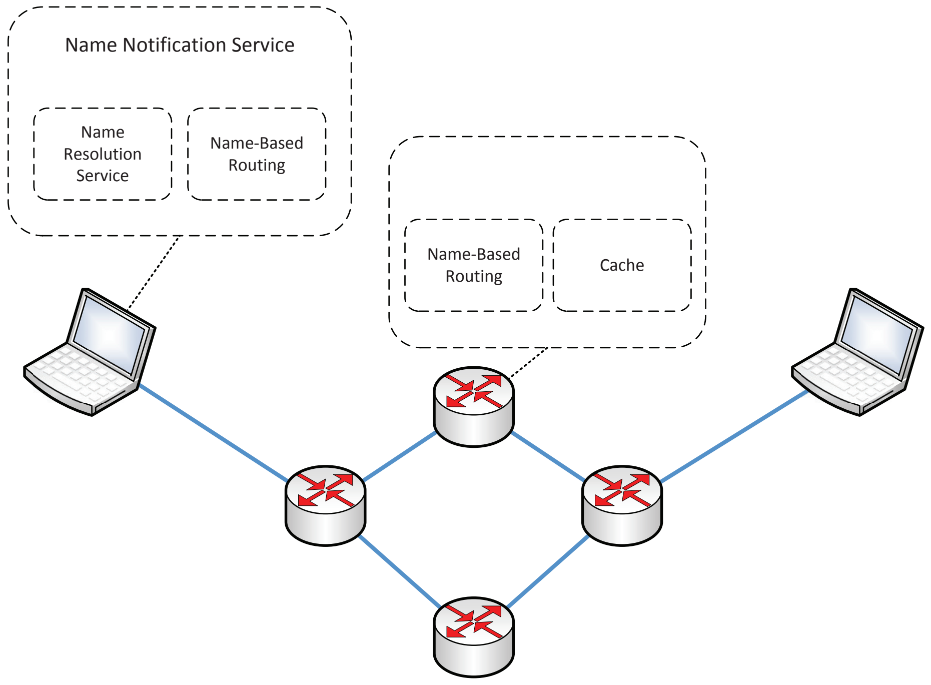


Figure 2.1: ICN Architecture

Different from the traditional *Pub-Sub* system [10, 11], ICN is pure receiver-driven architecture. The action of *Publish* in a traditional model contains actual data transmission, which means users can only subscribe and retrieve information in the future: users' *Subscribe* can only be accepted by the publisher after the content is prepared. However, *Publish* in ICN is only an announcement of information availability and registers the content in notification service. It is user's subscription that pulls the actual data from the data source. Therefore, the subscription could be registered even before the corresponding content is published. In ICN, publication and subscription do not need to be time synchronized.

Decoupling in time and space guarantees the flexibility of ICN architecture and particularly allows for native support of mobility which is not easily satisfied in current

Internet via Mobile IP. As wireless devices will switch networks and IP addresses frequently, Mobile IP is proposed to handle intermittent connectivity through a method called *triangular routing*: mobile device first needs to register in a home network where data packets will be first routed to and then delivered from that place to current location via a communication tunnel. This mechanism is inefficient since the routing path is non optimal. However, in ICN, since the host has no binding to some particular IP address, this problem is tackled in a much easier way: the mobile user just needs to re-send the subscription after moving to a new location and the contents will be delivered directly and optimally to the mobile user's current location instead.

2.1.2 Naming in ICN

Another important feature which distinguishes ICN from the current Internet is the network-layer naming, a key to *Pub-Sub* model which involves no topology information. A name specified to each content is the only identity for matching between publication and subscription. It ensures that users do not need extra attention to content address and provides more flexibility to network, choosing the best source for retrieving information. Design in naming varies in different ICN projects. It could be flat or hierarchical, readable or un-readable. A name could look like a HTTP request in a hierarchical design or merely a string of meaningless numbers.

More importantly, naming is designed at the network layer which promises ICN the capability of information-awareness which enables content delivery from an optimal source. Current Internet refers to Open Systems Interconnection (OSI) model design [46]. A packet header formulated in each communication layer attaches to a Service Data Unit (SDU) and then constitutes a Protocol Data Unit (PDU) which will be passed to the lower layer as a SDU. When the packet arrives at network layer, incremental headers from upper layers make the data packet inefficient to decode. In the current Internet, there is no demand to interpret delivered content in order to maintain the design of between-layer isolation. Such that, inefficient decode is not regarded as a problem since information is treated as a string of binary codes. However, such opaque design weakens the control and losses opportunities of network optimization. In contrast, ICNs re-design the network layer to achieve exact knowledge of what is transmitted by parsing the name of the content, which makes ICNs able to adjust routing hop-by-hop and exploit in-network storage automatically and optimally.

2.1.3 Caching in ICN

Caching is considered to be an essential component of ICN architecture. As shown in Figure 2.1, each router in ICN is equipped with name-based routing algorithm and cache storage. Caching in ICN could effectively reduce redundant data traffic generated by duplicated requests made from different users. Unlike the current Internet which must ensure a communication tunnel between explicit data consumer and producer, ICN supports information retrieval from anywhere so that users' requests may be satisfied by not only the data source but any intermediate nodes in the network. ICN is a feasible solution to alleviate the pressure from rapid data traffic growth (especially multimedia) without resorting to the application layer (like CDN and P2P).

Caching in ICN can be categorized into two types, *On-Path caching* and *Off-Path caching*. Off-Path caching is also referred as content replication with the aim of increasing data availability. By notifying and coordinating with name resolution service, a selection of routers on which requests are normally not reachable become alternative content providers. Off-Path caching is similar to CDN in concept but content replication happens to network layer primitives. As to On-Path caching, data could be only cached on routers along the forwarding path from information providers to consumers. If data has already been cached in ICN routers by requests from previous consumers, it will save time for delivering cached content to subsequent users once the name resolution paths of the contents overlap.

2.2 Architecture of Content-Centric Network

ICNs have substantially gained interest from both the research and industry community. There are several ongoing projects implementing versions of the ICN, *e.g.*, DONA [21], Publish-Subscribe Internet Technology (PURSUIT) [16], Scalable & Adaptive Internet solutions (SAIL) [14], COMET [17], to name a few. Pioneering among these architectures is the Content-Centric Network (CCN), also known as Named Data Network (NDN), developed by Jacobson *et al.* in [20]. CCN stands out as one of the prominent architectures that witnessed significant uptake by the research community; facilitating rapid benchmarking and insightful performance analysis across proposed protocols and schemes.

In this section, I focus on the CCN project as an example to explain basic concepts and features discussed in ICN and show how it is implemented in detail.

2.2.1 Name Structure

CCN is a chunk-based delivery system. Contents are split into chunks where each one is identified by a unique name. *Content Name* in CCN is structured hierarchically and consists of several *components*. Each *component* should be encoded in binary, convenient for transmission. However, since names in CCN, represented as variable-length binary bytes, are not human-readable, names are usually shown with format of Uniform Resource Identifier (URI), using delimiters (*e.g.*, '/') to separate components which are represented by strings or integers (meaningful characters). For example, a CCN *Content Name* can be '/cs.queensu.ca/thesis/wenjie.tex/_v1/_s0'.

In the above example, *components* are organized in a tree structure shown in Figure 2.2. In addition, naming in CCN also reflects current application-level convention, which tracks the evolution of information; **_s** is a segment marker followed by an integer (**0** in the example) which represents a block offset value; **_v** is a version identifier catering to frequent updates to information followed by a version number. In CCN, a longest prefix matching rule is applied for searching requested content. Through traversal of naming tree, subscriber's application thus can discover accessible data and control requested information with various levels of granularity for efficient delivery. For example, after the application has received the first version and first segment of data, it could either ask for the most right sibling **s1** by requesting for name of '/cs.queensu.ca/thesis/wenjie.tex/_v1/_s1' or even retrieve the whole content with any version via requesting '/cs.queensu.ca/thesis/wenjie.tex'.

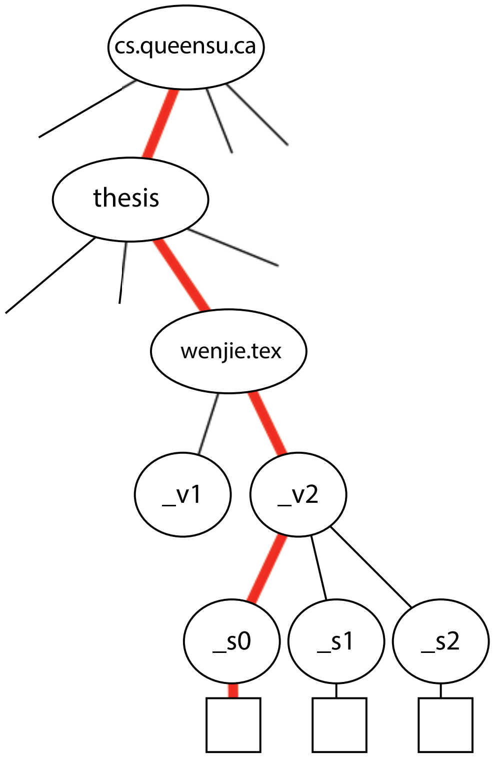


Figure 2.2: CCN Naming Tree (reproduced from [20])

2.2.2 Node Model

Inherited from ICN *Pub-Sub* model, CCN retrieves information through a receiver-driven mechanism as well. There are two types of CCN packets, *Interest* and *Data*, both of which own a common component, *ContentName*, as shown in Figure 2.3. This component is used to match between subscription and publication: users need to first broadcast queries with an appointed *ContentName* in an *Interest* packet and any data source (either the server or an intermediate router) in which exists content with the identical *ContentName* to *Interest* packet responds with *Data* packet.

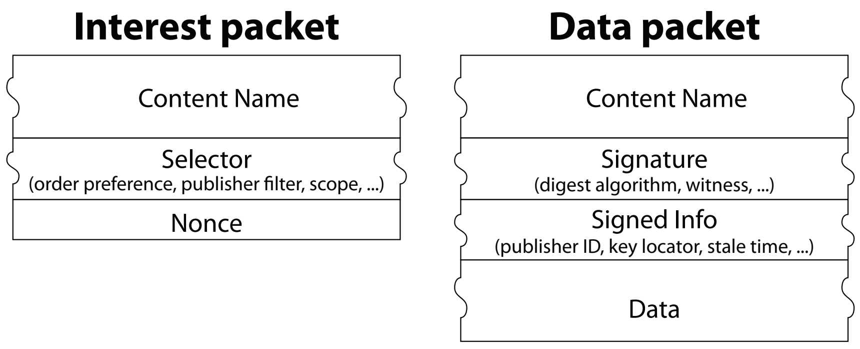

Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Figure 2.3: CCN Packet Format (reproduced from [20])

CCN router, also known as *Content Router (CR)*, has three main data structures: Content Store (CS), Forwarding Information Base (FIB) and Pending Interest Table (PIT). They are abstracted as tables in CR within which each entry is indexed by unique *Content Name*. A brief example is shown in Figure 2.4. Compared with the current Internet where the IP address is used as an index, lookup over all entries in CCN is extremely fast by applying hashing techniques with complexity *O(1)* while expensive *O(log(n))* tree search or TCAM (high-end hardware) has to be used in IP network.

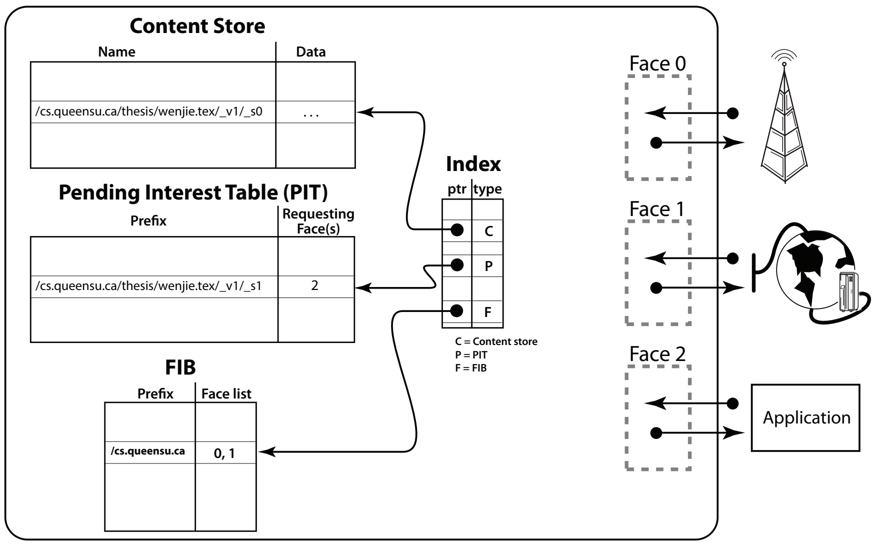


Figure 2.4: Components of CCN Router (reproduced from [20])

The FIB accomplishes a similar task to current IP network for requests routing except it could choose multiple output interfaces simultaneously for forwarding *Interest* packets towards several potential data sources guided by name-based routing strategy. Therefore, within CCN, it is possible to seek for information in parallel.

The PIT, however, is used to keep track of the *Interest* packets in order to guide *Data* packet downstream along the routing path to the subscriber. CCN uses 'bread crumb' strategy to couple name resolution and data routing. Only the *Interest* packet needs to be routed to potential source. As the *Interest* packet arrives at a certain router, it registers in PIT, leaving the 'bread crumb'. Once matched *Content Name* is found on some node, the corresponding *Data* packet simply follows this 'bread crumb' in PIT towards the initial requester without routing.

Content Store acts differently with the buffer memory in IP router because of its replacement strategy (*e.g.*, LRU or LFU). The buffer in an IP router is basically a FIFO queue to balance the uneven speed between input and output interfaces. However, since CCN packet is self-identified with a unique name, Data packets kept in the Content Store have potential benefit to other consumers whose request is for the same content.

When an Interest packet arrives at Content Router, the *Content Name* is first extracted and longest prefix lookup is applied over Content Store. If the same name can be found, the subscription is immediately satisfied with local cache and the *Interest* is discarded. Otherwise, the forwarding process is initiated. The *Content Name* is searched among entries in PIT. A matched entry means the request for the same content has ever reached this router before but has not been satisfied yet. Such that, there is no need to propagate this packet to the potential information source again. Instead, CR only generates the 'bread crumb' by adding the arrival interface of the new *Interest* packet to Requesting Face (as shown in Figure 2.4) of the entry to ensure when the corresponding *Data* packet arrives, a copy could be sent through interface that the new *Interest* packet arrived on. Otherwise, a further searching should be carried out upon FIB table. If there is a matching entry in FIB, the *Interest* packet has to be sent upstream towards data source and a new PIT entry thus is created to keep track of the routing path.

When a *Data* packet arrives, a longest-match lookup within Content Store is also preferred over PIT. A matched entry found in Content Store means the same content has been cached on this router such that this duplicated packet is discarded. Otherwise, data is first cached in Content Store and then several copies are sent out via each interface listed in the Requesting Face of corresponding PIT entry.

2.3 Caching Over Information-Centric Networks

In this section, I discuss related work on caching schemes over ICN. I first emphasize the characteristics of caching in ICN compared with other overlay networks and then provide a brief explanation of several issues which are broadly concerned in research. Within these research problems, caching decision policy is highlighted.

2.3.1 Comparison With Caching in Overlay Networks

Caching in ICN differs from current overlay networks (*e.g.*, CDN and P2P) in many ways. First of all, due to unified and consistent naming, there exists a one-to-one matching between the content and its name which enables ICN's network layer to be information-aware which makes it possible for ICN to take advantage of caching. Secondly, in ICN, caching is ubiquitous. Compared with CDN and Web caching, an important research problem in these two areas is to determine a set of nodes in the network on which cache space should be allocated. However, caching opportunities in ICN are everywhere, which reflects the high dynamics of the system. Thirdly, in CDN, data is pushed and updated manually on surrogate servers based on prior knowledge (such as access frequency) and could only be visited through connection redirection made by DNS resolution. While in Web caching, the cache space only on those nodes which lie on the request route are accessible. ICN, however, synthesizes these two designs, providing both Off-Path and On-Path caching.

Nevertheless, there is still one point common between caching in ICN and overlay networks. Same as overlay networks, ICN also faces the problem of where and what content should be cached. The goal of content placement in ICN is to optimize objectives by targeting different metrics under particular application scenario, such as energy efficient delivery [28] or video streaming, catering to different traffic types, such as file, multimedia, web page, etc. As decoupling architecture makes optimization hop-by-hop possible, caching in ICN achieves more flexibility in control and thus surpasses overlay networks by providing cross-application performance improvement instead of just aiming at saving data traffic and alleviates the load on the original server.

2.3.2 Research Problems

Recently, there are three main areas which receive sustained attention from the research community: *cache capacity allocation*, *cache space sharing* and *cache decision policy* respectively [45].

The research of cache capacity allocation is: given fixed total storage resources, how to arrange capacity among all nodes in ICN to make sure the whole system reaches its optimized performance. Different methods have been proposed to tackle this problem. Rossi *et al.* in [34] utilize the information of network topology and allocate cache capacity based on centrality of each node. For example, if I use the simple degree centrality $d(i)$ which is defined by the number of links on each node, the cache storage is proportional to $\frac{d(i)}{\sum_i d(i)}$ such that nodes with high connectivity in topology will be allocated more resources. A similar idea is discussed in [42] and [33], authors define *Core* and *Regular* to distinguish routers in ICN where *Core* are those nodes located on important intersections in the topology and should be allocated more cache space. Wang *et al.* in [40] proposed a model, solving the optimal allocation problem by dynamic programming, in which it takes not only the static topology but also the dynamics of users' requests into consideration.

Since ICN architecture supports cross-application caching, different traffic types must compete for limited cache resources. Research on cache space sharing is to find a way to partition cache capacity across different traffic classes on all nodes. In 2004, Lu *et al.* has already proposed an approach to adjust partition ratio in the context of web caching based on the dynamic feedback control in [29]. A native approach under the scenario of ICN is to use a fixed ratio on each node but the issue of efficiency is argued in some work [9]. Carofiglio *et al.* presented in [9] a dynamic partition method using *priority-based sharing* and *weighted fair sharing*. Lower priority class of contents will be replaced by contents belonging to higher priority classes dynamically.

Cache decision policy is one of the hottest topics in ICN research. It is an area which studies what and where objects are to be placed in caches. My work on caching strategy for dynamic adaptive streaming belongs to this area. In the next section, I will provide detailed explanations on schemes within this scope.

2.3.3 Caching Decision Policy

According to the degree of coordination in cache decision process, caching decision schemes can be classified as either *explicit* or *implicit*. Explicit caching policy requires exchanging users' access statistics (*i.e.* popularity of items), storage availability among nodes which thereby incurs overhead. Instead, implicit caching decision is made independently by each node and no statistics need to be exchanged.

2.3.3.1 Explicit Caching Policy

Explicit caching has been widely explored under CDN. As mentioned earlier, each surrogate node in CDN requires knowledge of the whole network through either offline or online communication to work out the optimal solution. This results in added maintenance costs. Considering the dynamics of ICN, it is necessary to propose approaches which decrease the overhead of exchanging statistics. Recent research [45] mainly utilizes two approaches: *path coordination* and *neighbourhood coordination.*

In path coordination, exchanging statistics only occurs on nodes located on routing path. Thus, piggyback is typically the fundamental approach to avoid excessive coordination. For example, Li *et al.* in their work [24] focused on minimizing the remaining traffic based on popularity of contents. Their distributed algorithm starts with collecting item's popularity statistics and refreshes periodically from lower tier routers. Next, routers are responsible for passing the statistics together with the caching decision along the name resolution path to higher tier routers. Such that, each router could be notified what objects have been cached along the routing paths from users, in order to avoid caching redundant objects.

In addition, neighborhood coordination is another way to maintain limited information exchange. Just as its name implies, coordination only occurs on each node's direct or two-hop away neighbours. It is usually coupled with a modified forwarding strategy (*Interest* routing) in order to utilize caching more efficiently. Recent research done by Li *et al.* [25] is representative of this approach. Figure 2.5 shows an example of cache distribution with neighbourhood coordination in [25]. Authors used a hash function to distribute different contents over adjacent routers. This method prevents duplicated caching in the neighborhood and chooses the outgoing interfaces for *Interest* packet routing decided by that hash function. For example, the request for chunk 0 received by $R1$ will not be forwarded to $R2$ according to default shortest path protocol even though $R2$ is close to the server, but to $R0$ instead because hash function on $R1$ could calculate that chunk 0 is highly possible to be cached on $R0$.

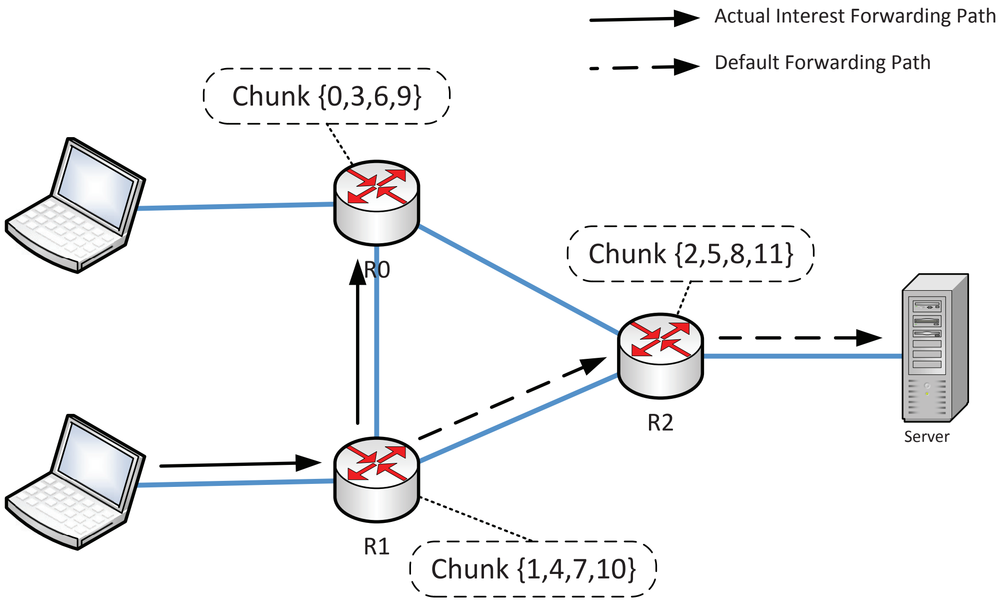


Figure 2.5: An Example of Cache Distribution With Neighbour Coordination

2.3.3.2 Implicit Caching Policy

Implicit caching policy works independently on each node. The advantage of this approach stems from its simple implementation while still maintaining reasonably good performance. However, since cached contents are not arranged from the whole system's point of view, implicit caching policy may not be the optimal solution.

- **LCE**: The simplest but widely applied policy is called leave copy everywhere

(LCE) [20] which is also the default method in ICN protocol [14]. Contents will be cached on each router along the delivery path. Since contents will always be served from cache in downstream routers if there are any requests from users, duplicated caching on upstream nodes thus incurs huge redundancy and inefficient usage of cache storage. If the total capacity is limited, new incoming objects in cache have to be switched out frequently following the Least Recently Used (LRU) replacement scheme, even before it could serve any requests which degrade the performance of this method further.

- **LCD**: Leave Copy Down (LCD) [22] is an improved version of LCE by restricting which nodes could cache contents. When there is a cache hit on a particular router, only its direct downstream node could cache this object again. It helps to move objects which are frequently hit close to users such that LCD achieves better performance than LCE. Another approach, Move Copy Down (MCD), is regarded as the upgraded version of LCD by eliminating the redundancy. Whenever the cached contents are moved downstream, the replica will be removed from cache on any upstream nodes.

- **Prob**: Copy with Probability (Prob) is a general approach. Caching decision is made according to a probability p. This value p, could be fixed, standardized [4] for each router, *e.g.*, $p = 1/4$. In addition, p could also depend on the distance of routing path. For example, if the hop count of the data packet delivery path is 3, then p is set 1/3 for all nodes located on that route. This method thus expects the object could be cached and only cached once.

- **ProbCache**: Probability p can be not only fixed but also dynamic. Psaras *et.al.* presented *ProbCache* model [33] which takes network topology into consideration. Each router will be assigned a varied probability which is decided inversely proportionally to the hop counts from users to this router. If the router is close to requesters, undoubtedly, a higher probability should be assigned since the further contents are cached, the less benefit the network can earn through caching.

2.4 Dynamic Adaptive Streaming Over HTTP

Suffering from a continuously changing network condition, Internet users may not possess a stable bandwidth during the whole process of video demand. If the number of users under the service increases, the limited network resources then have to be shared among more people, thus available bandwidth dedicated to each user will decrease.

Therefore, in order to face the challenge of varying network conditions while still able to satisfy increased data demand, especially on multimedia data, DASH is proposed to recognize and adjust to network capacity dynamically for each user's request with the purpose of no interruption on playback no matter which section of streaming is being watched and what the network condition is. If the current request exceeds the bandwidth the network can provide, DASH changes stream on-the-fly to lower quality without negotiation with streaming server. If not, DASH suggests streams switching to higher quality automatically to reach maximum QoE for users.

In this section, the working process of DASH and its components will be explained in detail. I will also elaborate that the ICN can seamlessly integrate with dynamic adaptive streaming to serve various bit rates of streams. In the end, I present a brief review on recent research on applying adaptive streaming over ICN.

2.4.1 QoE and QoS

Given similar pricing schemes from different network providers, the primary factor which may influence the user's choice among these network services is the expected and experienced quality. Consequently, how users perceive reliability, usability and quality becomes the primary interest of network providers who compete to improve their services which brings, named Quality of Experience (QoE).

QoE combines users' experience with non-technical parameters (*e.g.*, glitches, artifacts, excessive waiting times, etc). However, another important concept, named Quality of Service (QoS) is represented by technical parameters, such as throughput, packet drop rate, latency, etc. Obviously, QoS problems **imply** QoE problems. Their quantitative relationship could be found in [15].

A better QoE in my work could be defined as, with varying network conditions, users who request for streaming data could receive better video quality with no interruption or freezing in the playback.

2.4.2 DASH Principal

HTTP is regarded as a popular approach which is initially designed for file transfer. However, applying HTTP over real-time multimedia streaming has its unique advantage. Unlike other streaming methods such as Real-Time Streaming Protocol (RTSP) [36] where it transmits data through a continuous stream over TCP or UDP transport and never tracks the state of client once the session is established, HTTP is stateless and handles the request as a standalone transaction, thus promotes flexible, and client-driven control.

Before user initiates DASH requests, there is a preparation stage on HTTP server where streaming files are encoded with multiple bit rates and chopped into small HTTP-based file segments. These various file formats are defined in Media Preparation Description (MPD).

MPD is organized hierarchically. As shown in Figure 2.6, it consists of one or more non-overlapping *Period*s where each *Period* is to present a logical content segmentation (*e.g.*, new contents, advertisements, etc.). Each Period consists of one or more *Representation*s, which describe available encoding options. In other words, Representations define quality of stream differed by bit rates, resolution, codecs etc. A Representation includes a number of small-piece, sequential *Media Segments*. These segments are chopped with a fixed interval (*e.g.*, 10s in Figure 2.6) pre-defined in Representation and connected end to end with each other. Each Media Segment is assigned a start time for fast seeking and a unique URL for individual request.

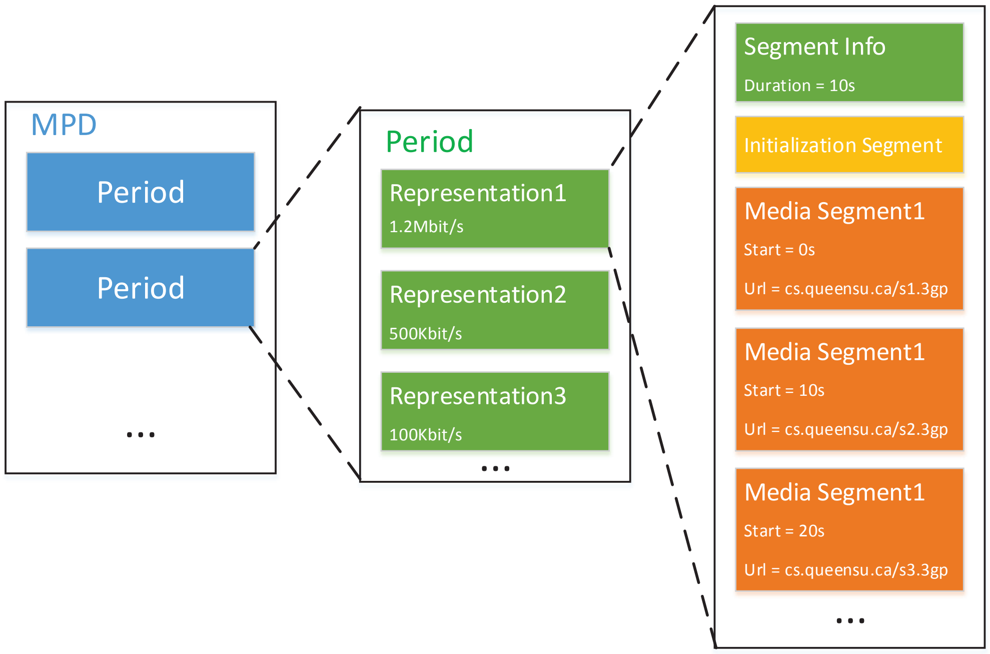


Figure 2.6: Media Preparation Description (MPD) Organization

After all versions of multimedia files have been prepared, MPD is spread over streaming servers. When clients intend to watch streams, they have to request for MPD at the very beginning through HTTP GET. Then, by parsing MPD, clients get to know which bit rates are accessible on servers and based on the available bandwidth, streaming control will suggest requesting media contents segment by segment via assigned URL. This process is shown in Figure 2.7.

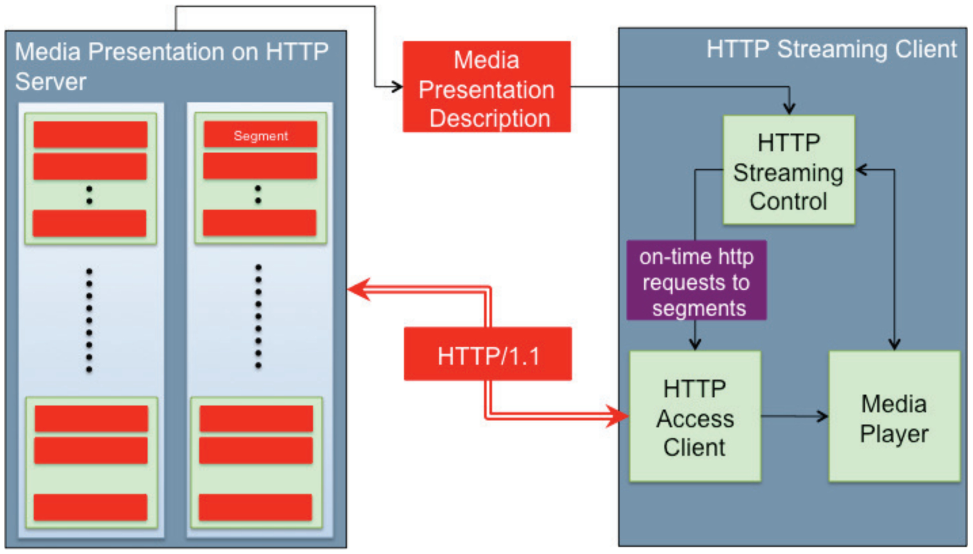

Media Presentation on HTTP Server
Segment
Media Presentation Description
HTTP Streaming Client
HTTP Streaming Control
on-time http requests to segments
HTTP/1.1
HTTP Access Client
Media Player

Figure 2.7: Interaction between Client and Server in DASH (reproduced from [37])

The core component in HTTP streaming client is adaptation strategy in streaming control. It basically is a feedback mechanism where DASH client updates decision repeatedly in response to instant network condition. Such a decision usually is made by measuring the **average throughput** of previous requested video contents and is used to guide which bit rate should be chosen in the request for next segment. Considerable research has been devoted to that adaptation strategy such as [41], [38] and [39] considering playback smoothness, average quality, playback interruption, etc.

2.4.3 Design Over Information-Centric Networks

It is also instrumental in the development of ICN to cater to increased multimedia demand. Inspired by the concept of dynamic adaptive streaming over HTTP, serving multiple bit rates according to network condition is also appealing to research within the scope of ICN.

Since both DASH and ICN are client driven, dynamic adaptive streaming thus has opportunity to apply over ICN without much modification on existing working process. The only change needs to be made is to substitute URL with a unique ICN name to identify a Media Segment. Instead of delivery video files over current Internet, ICN now is the underlying network. Clients still follow the same working process by starting with subscribing MPD file, then requesting for a particular named content explicitly.

Since all segmented files are transported over ICN, it is possible to accelerate delivery via ubiquitous caching. Grandl *et al.* in [18] pointed out the issue of caching competition between contents in different bit rates over CCN. Caching all versions (bit rates) of the same video content is not a feasible solution since it exhausts limited caching resources quickly. Thus they proposed *DASH-INC* which only keeps the highest rate in cache and performs transcoding once a lower rate is requested. However, such method will incur two problems: First, since transcoding is a computationally intensive task, in-node processing will become less feasible as the demand for video traffic increases. As large number of requests arrive at some router, almost at the same time, the router will be overloaded by transcoding which increases response latency and degrades overall network performance. Secondly, video segments encoded with the highest bit rates have the largest packet sizes. Caching only the highest bit rate on routers which serve users requesting low bit rates is not an efficient method to utilize cache capacity. Therefore, in my work, instead of transcoding, I argue for caching multiple bit rates according to proposed strategy.

Even though there is already plenty of research on DASH, these attempts are largely HTTP based, and do not address ICN architectures with intrinsic caching capabilities. In addition to the work of *DASH-INC*, most of the recent research shows experimental results of current CCN protocol. In [23], Lederer *et al.* conducted extensive experiments on protocol overhead when transmitting data packets upon HTTP and CCN. Without considering the caching capability, the result proves the performance of transmitting over CCN can definitely surpass over HTTP 1.0 but CCN incurs large protocol overhead. The reason lies in the fact that CCN is immature and the implementation is just a prototype. Liu *et.al.* conducted an experiment [27] studying caching behaviour over CCN when dynamic adaptive streaming is applied. Clients in the experiment generate requests in sequence and the results prove the effectiveness of caching: the last client could even be served with bit rates higher than actual bandwidth.

To the best of my knowledge, my work is the first attempt to tackle this problem in ICN by caching contents with multiple bit rates according to their popularity.

Chapter 3

DASCache System Model and Solution

In this chapter, I describe the system I built for video contents delivery with multiple bit rates, in which my caching strategy assists to satisfy requests in order to achieve least average access time per bit among all users. The problem is formulated as optimization and solved by Binary Integer Programming (BIP).

3.1 Problem Formulation

It is important to explain the design of caching strategies in light of the operation of ICN, and their intrinsic in-network caching properties. In this section, I will elaborate on the proposed *DASCache* management strategy, and detail the assumptions and axioms upon which my system is built. More importantly, the objective of minimizing access time while improving throughput is explained in terms of the facilitated improvement in users' QoE.

3.1.1 System Description and Axioms

I build *DASCache* over CCN architecture. It is important to note that my proposed strategy is designed to address the primitives of ICN paradigms in general, and is not specific to CCN. However, I choose to address CCN as a viable use case to demonstrate utility and benchmark to current state-of-the-art caching models in the literature. This is further elaborated upon in the Performance Analysis, Chapter 4.

My system primarily targets video delivery in ICN. All packets in the envisioned CCN scenario are assumed to be related to video contents and the size of network is encompassed by a single ISP [24]. However, my solution could be generalized for multiple (co-existing) ISPs by considering cache partitioning and multi-homing. To maintain a concrete description, I will opt for analyzing and presenting *DASCache* in light of a single ISP.

There are three different types of routers: the edge router, intermediate router and a single gateway. It is assumed that all chunks are kept in a repository (main content producer) which is reachable via the ISP's gateway over a high bandwidth link; mimicking a common scenario in real world deployments and envisioned ICN architectures. All clients are served by edge routers; intermediate routers never connect with clients directly. In this model, node failure is not considered. If there is any router which is not accessible, *DASCache* just needs to refresh the routing topology and runs for a new cache management solution.

Although CCN supports multi-path routing which increases the chance of cache hits, the overhead studied by [35] demonstrated performance degradation at the same time. In order to contain my model, I adhere to CCN's default setting: single-path routing and use the shortest-path algorithm without cooperative schemes that add an extra complication in building the model. My proposed strategy is indifferent to any network topology. However, with the setting of single-path routing and a single gateway, the routing topology could be abstracted as a tree (marked in red lines) shown in Figure 3.1.

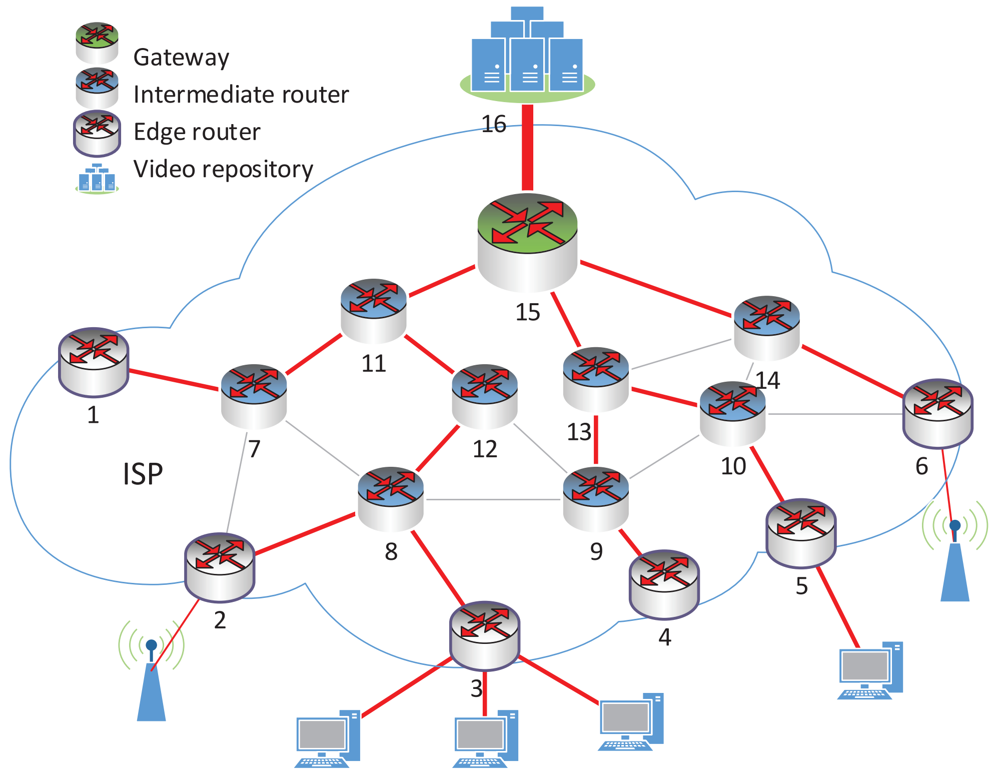


Figure 3.1: Network Topology of *DASCache*

There is no transcoding involved in my work. All video segments encoded with multiple alternative bit rates are backed up on the video repository in case they are requested by users according to varying link conditions but missed by the cache in routers. *DASCache* will work in rounds to update the cache contents in each router. The detailed working process will be explained in Section 3.1.2.

3.1.2 DASCache operation over rounds

DASCache works in rounds. In each round, the network provider is responsible to set up a monitoring window to collect data at each edge router which contain users request frequency on video contents and requested bit rates. Based on this aggregated data, It is assumed that the network provider will be able to predict the bit rate chosen by clients in the next round. *DASCache* is triggered to run for an optimal cache allocation to tune the content placement according to this predicted information at the end of each round, as shown in Figure 3.2. Once the placement is determined, the contents in cache remain unchanged during the entire round.

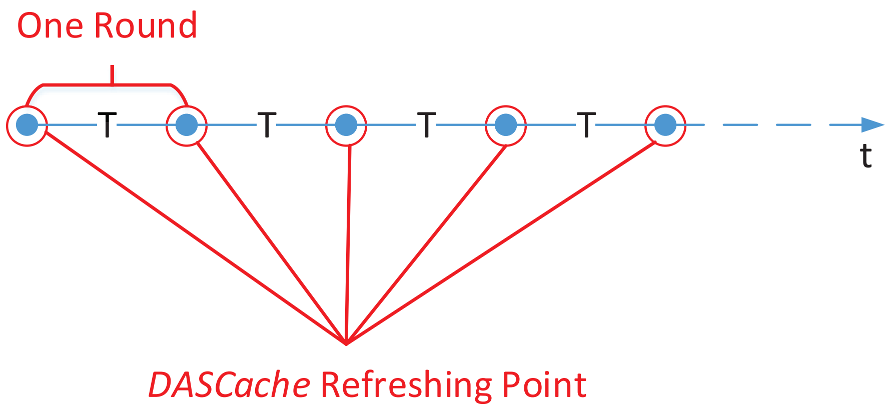


Figure 3.2: *DASCache* Working Process

3.1.3 Assumptions

I allow for variations in requested bit rates for every user, since adaptation algorithms should cater to individual requests by each user. The chosen bit rates may vary even under the service of the same edge router at any given time. However, because users connected to a given edge router share similar network conditions, typically most users will choose (end up with) the same bit rate that current network resources could support.

Previous research made assumption that the web page requests from users received by a server follows the Poisson Process [5]. As to my specific problem, the distribution of requests for multimedia data received by CCN routers is still unclear. Since users' requests on video objects influenced by the adaptive strategy and buffer status, the real traffic model still needs further study. However, in this work, it is assumed the *Interest* packets received by edge routers follow an independent Poisson Process. Since *DASCache* only refreshes the contents of cache in CCN routers at the end of each round, for the reason of simplicity, I also assume that the forwarded *Interest* packets received by intermediate routers follow the Poisson Process as well.

3.1.4 Notations

All nodes in the proposed network are modelled as a connected graph $G = (V, E)$ in which node in V are composed of a set of Edge nodes N, intermediate nodes I and a single gateway R. Node j, where $j = 1, \ldots, |V|$ is equipped with content storage with capacity C_j[1].

[1] C_j represents the class-specific capacity dedicated to video caching at node j

Video files are divided into K chunks, according to a fixed time interval, and each chunk is identified by a unique number. Chunk k is requested by clients with probability $\{q_k\}_{k=1,\ldots,K}$. There are B bit rates available for request while a vector $S_{1\times B}$ is used to denote the size of chunks encoded with different bit rates. For example, $S(b), b = 1, \ldots, B$ denotes an element in vector S representing chunk size with bit rate b. Hence, each video chunk now is identified by a two-dimensional index (k, b).

$\{(\pi_{B\times 1})_j\}_{j=1,\ldots,|N|}$ is defined as a binary indicator vector for edge router j, in which element $\pi_j(b) \in \{0, 1\}, b = 1, \ldots, B$. If $\pi_j(b) = 1$, it indicates that bit rate b is requested by clients at edge router j. As it is assumed that clients served by the same edge router will request for the same bit rate, thus I have $\sum_{b=1}^{B} \pi_j(b) = 1$. Let matrix $\{(x_{K\times B})_j\}_{j=1,\ldots,|V|}$ denote my cache deployment configuration, where $x_j(k, b) \in \{0, 1\}, k = 1, \ldots, K; b = 1, \ldots, B$. $x_j(k, b) = 1$ indicates that the video chunks indexed by (k, b) should be cached at node j during the next round.

Users' requests received by each router is assumed to be following Poisson Process. According to the superposition property, the interest arrival rate on intermediate routers is the sum of rates of its children nodes. I denote the interest arrival rate matrix at node j with $\{(\lambda_{K\times B})_j\}_{j=1,\ldots,|V|}$. The element at row k column b represents request rate for video chunk with index (k, b). As for edge router j, let λ'_j denote the request rate made by all clients and b_j the chosen bit rate. Thus, I have $\lambda_j(k, b_j) = \lambda'_j q_k$.

$\{L_j\}_{j=1,\ldots,|N|}$ is a sequence of node IDs. Elements in this sequence are those which receive interest packets for video chunks sent from edge router j. In other words, $\{L_j\}$ contains nodes located on the routing path starting from edge node j. I define

$L_j(1) = j$ and $L_j(i+1)$ always denotes the ID of the next hop node of $L_j(i)$ on the routing path of *Interest* packet. The last element in $\{L_j\}$ is the video repository accessed by the ISP. For example, following the topology depicted in Figure 3.1 and considering $j = 2$, L_2 will be $(2, 8, 12, 11, 15, 16)$.

To summarize, Table 3.1 lists all notations used in my model. Please note that those symbols that have not been explained earlier but will be presented later are included as well.

Table 3.1: Notations in System Model

V	CCN routers within the ISP
N	Edge routers
I	Intermediate routers
R	Gateway
K	Number of Video Chunks
B	Number of Available Bit Rates
C_j	Cache capacity (size) of node j
q_k	Popularity distribution of requests for chunk k
π_j	Binary indicator of requested bit rate at edge router j
S	Video chunk size
x_j	Cache deployment configuration matrix at router j
λ_j	Interest request arrival rate matrix at router j
λ'_j	Sum of users' request rate received by edge router j
L_j	Set of nodes on routing path from edge router j
$VRTT_k(j)$	Virtual round trip time delay from edge router j
$D(i, j)$	two-way delay between router i and j
P	Processing and propagation delay on each link
θ_{ij}	Downlink bandwidth between router i and j
Q_{ij}	Queueing delay between router i and j
μ_j	Interest miss rate matrix at router j
ρ_{ij}	Traffic load on link between router i and j
H_j	Set of children nodes of intermediate router j
$\varphi_{ij}(b)$	Data packet service time delivered from router j to i
RS_{ij}	Residual service time on j when packets are delivered to i
$N_Q(b)$	Number of packets (encoded with b) in the queue

3.1.5 Research Statement

As users adjust requested bit rates with varying network bandwidth, *DASCache* would improve network performance and video delivery in terms of average throughput by optimizing video cache placement over routers.

3.1.6 Optimization Objective

The problem is formulated as optimization. As in adaptive streaming, throughput is a fundamental metric in rate adaptation algorithms to estimate maximal supported bit rates by measuring the Round-Trip Time (RTT) delay of video chunks. I argue for approximating the maximal average throughput each user could achieve by minimizing the average access time per bit. Therefore, the optimization objective could be represented as:

$$\min \quad \sum_{j=1}^{|N|} \frac{\mathbb{E}[AccessTimePerBit(j)]}{|N|} \tag{3.1}$$

The optimization only targets on the *average* throughput that the network can provide without considering the fairness across all users. To calculate $\mathbb{E}[AccessTimePerBit(j)]$, I first define the Virtual Round-Trip Time (VRTT) delay in Equation 3.2 where $D(i, j)$ denotes the value of two-way delay among router i and j. Then, the VRTT of video chunk k requested by clients under service of edge router j is:

$$\mathbb{E}[VRTT_k(j)] = \sum_{i=1}^{|L_j|-1} \bigg(\mathbb{E}[D(L_j(i), L_j(i+1))] \Big(1 - \max_{m=1,\ldots,i} x_{L_j(m)}(k, b) \Big) \bigg) \tag{3.2}$$

The summation in Equation 3.2 is to add up the average delay on each link of routing path starting from edge router until the node in which the interest is satisfied.

To elaborate through Figure 3.1, I take $j = 2$ as an example and b_j as requested bit rate. If there is no cache in the network, $\mathbb{E}[VRTT_k(2)]$ is the sum of two-way delay on links between routers $(2, 8), (8, 12), (12, 11), (11, 15)$ and $(15, 16)$. However, whether data packets actually travel through the link depends on the cache on the routing path. For example, delay between router $(12, 11)$ should not be add into $\mathbb{E}[VRTT_k(2)]$ when the video chunk k has already cached in router 2, 8 or 12. To explain the max operator in 3.2, if the maximal value among binary indicator $x_2(k, b_j), x_8(k, b_j), x_{12}(k, b_j)$ is 1, $\mathbb{E}[D(L_2(12), L_2(11))]$ will not be added. In general, Equation 3.2 sums up the delay on links when the content has not appeared in cache along the routing path.

The expected access time per bit for chunk k requested from edge router j is denoted as,

$$\mathbb{E}[AccessTimePerBit_k(j)] = \frac{\mathbb{E}[VRTT_k(j)]}{S \cdot \pi_j} \tag{3.3}$$

Such that,

$$\mathbb{E}[AccessTimePerBit(j)] = \sum_{k=1}^{K} q_k \times \frac{\mathbb{E}[VRTT_k(j)]}{S \cdot \pi_j} \tag{3.4}$$

Hence, the optimization objective in 3.1 is formally represented as:

$$\min \quad \sum_{j=1}^{|N|} \sum_{k=1}^{K} q_k \times \frac{\mathbb{E}[VRTT_k(j)]}{|N| S(b_j)} \tag{3.5}$$

where b_j is subject to $\pi_j(b_j) = 1$.

3.1.7 Queueing model and Derivations

In order to compute the $VRTT_j(k)$ in Equation 3.2, I make a quantitative analysis for each link to calculate the $D(i,j)$. Since processing and propagation delay are relatively small, I assign a fixed value P to denote them on each link. Let θ_{ij} denote the downlink bandwidth between node i and j (data packet delivered from j to i). When video chunk with bit rate b' is transmitting, the average two-way delay between node i and j, $\mathbb{E}[D(i,j)]$, consisting of propagation, processing, transmission and queueing delay, is calculated as

$$\mathbb{E}[D(i,j)] = P + \frac{S(b')}{\theta_{ij}} + \mathbb{E}[Q_{ij}] \tag{3.6}$$

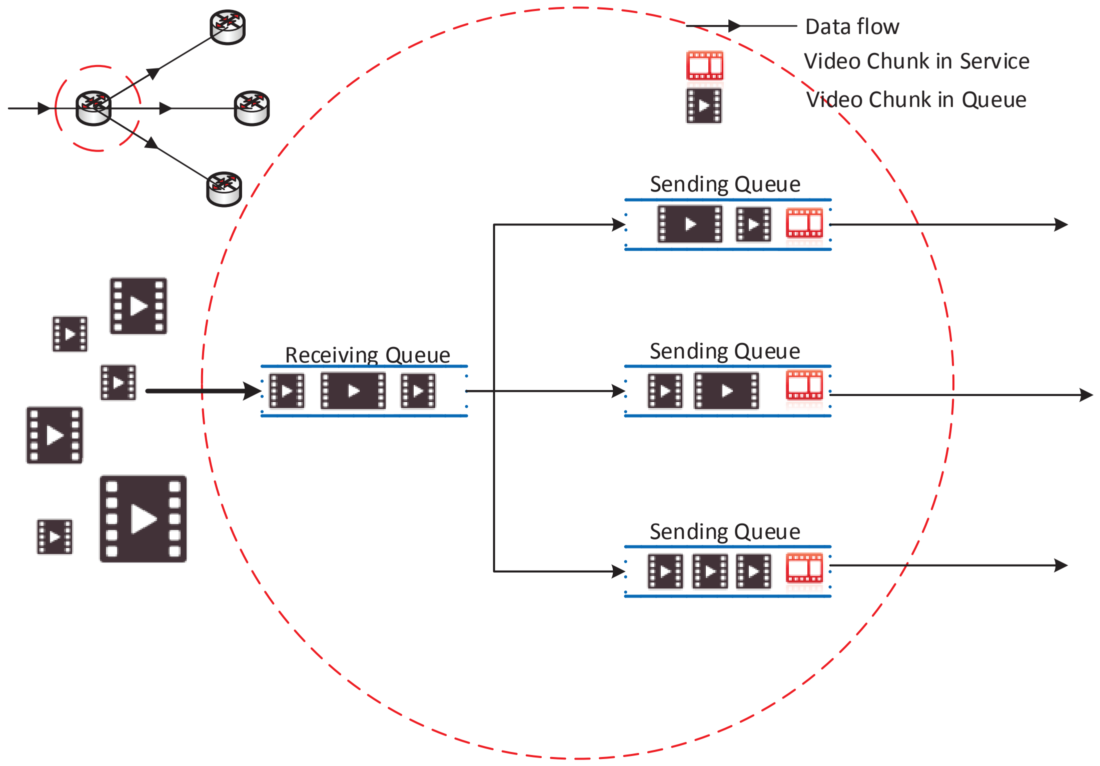


Figure 3.3: Queueing Model for Adaptive Streaming

I employ a multiclass M/G/1 queueing model [6] to calculate the queueing delay, $\mathbb{E}[Q_{ij}]$. M/G/1 is used to denote the type of queueing system where 'M' denotes the memoryless arrival process. Since the *Data* packet is delivered according to *Interest* request which is modelled as a Poisson Process, then data arrival on each router also follows a Poisson Process which owns the memoryless property. 'G' is the general probability distribution of service times. As to my specific problem, data packets containing video contents with multiple bit rates are mixed up. The size of packets encoded with a given bit rate is equivalent to each other because each packet contains the same length of playback time. Such that, the job service time of a packet in the queueing system is always chosen from a set of fixed values determined by available bit rates. Therefore, the output process owns neither memoryless property nor constant service time. However, the service time for packets encoded with a particular bit rate is deterministic. Based on the arrival rate of packets of different bit rates, I can still calculate the mean service time for this queueing system. '1' represents the number of service node in the system where only one node is considered in my queueing model to receive data packets.

The reason why I only consider multimedia data in the queueing analysis is to simplify the system model. As mentioned in Chapter 1, streaming traffic is dominating the total data traffic over the Internet. Other traffic types, such as web pages, emails, do not lead in percentage of data traffic. According to the analysis (Equation 3.3 and 3.10) made in the following paragraphs, I can observe the smaller content size will generate less influence on average queueing delay. Compared with several hundreds or thousands of KB per segmented video file, a regular web page is usually several dozens of KB at most. Therefore, other traffic types, like web pages have limited impact on result of queueing analysis in my particular scenario and thus are omitted in the model.

To use this model, I must ensure there is no overload on the link. Denote with ρ_{ij} the traffic load between node i and j, it means $\rho_{ij} < 1$ must be satisfied. If not, the buffer on node j is easily overflowed such that there will be a large number of data packets dropped. If it is caused by an increase in the number of requests, a rate adaptation strategy may arrange users affected by this congestion, by requesting lower bit rates to relieve the load. Such a trend (increasing number of requests) could also be monitored by ISP and make proper prediction. Otherwise, the network is not in a stable status and will return to original state later. In either case, $\rho_{ij} < 1$ will be satisfied in the end. Denote with router H_j a set of children nodes of router j in tree topology. I have the following proposition to derive $\mathbb{E}[Q_{ij}]$.

Proposition 3.1.1. *If $\rho_{ij} < 1$, $\forall j \in V, \forall i \in H_j$, the average queueing delay for data packets from router j to i is*

$$\mathbb{E}[Q_{ij}] = \frac{1}{2}\frac{\sum_{b=1}^{B}\mu_i(\cdot,b)\mathbb{E}[\varphi_{ij}(b)]^2}{1-\sum_{b=1}^{B}\mu_i(\cdot,b)\mathbb{E}[\varphi_{ij}(b)]} \tag{3.7}$$

Proof. The queueing in my video delivery scenario is illustrated in Figure 3.3. According to the superposition property, the rate of interest packet which request for bit rate b missed by caches in router i, $\mu_i(\cdot,b)$ is given by,

$$\mu_i(\cdot,b) = \sum_{k=1}^{K}\lambda_i(k,b)(1-x_i(k,b)) \tag{3.8}$$

$$\lambda_i(k,b) = \sum_{m=1}^{\|H_i\|}\mu_{H_i(m)}(k,b) \tag{3.9}$$

Equation 3.8 and 3.9 show the relation between interest arrival and interest miss rate. Take Node 11 in Figure 3.1 as an example, Equation 3.9 means that the interest packet arrival rate on Node 11 is the sum of rates missed by caches in Node 7 and 12. Equation 3.8 calculates the interest miss rate on Node 11 based on the arrival rate and whether requested contents are cached on Node 11 or not. If it is cached (which means $x_{11}(k,b) = 1$), the corresponding request could be immediately satisfied and there will be no need to forward this interest packet. As node j is responsible for forwarding data packet along the link to the node i according to the 'bread crumb' left by interest packet, $\mu_i(\cdot,b)$ is also used as the data (encoded with bit rate b) input rate to the buffer in router j. The expected job service time of data packets encoded by bit rate b, $\mathbb{E}[\varphi_{ij}(b)]$ is given by,

$$\mathbb{E}[\varphi_{ij}(b)] = \varphi_{ij}(b) = \frac{S(b)}{\theta_{ij}} \tag{3.10}$$

and then, the traffic load ρ_{ij} is,

$$\rho_{ij} = \sum_{b=1}^{B} \rho_{ij}(b) = \sum_{b=1}^{B} \mu_i(\cdot, b)\mathbb{E}[\varphi_{ij}(b)] \tag{3.11}$$

when $\rho_j < 1$, it means that the input rate to the queue is less than the output rate. Hence, the model is in stable status. In other words, the requested resource will never exceed the maximum that the network can provide.

To calculate $\mathbb{E}[Q_{ij}]$, I apply Little's Theorem[2] and extend *Pollaczek-Khinchin (P-K) formula* listed in [6]. In P-K formula, the expected waiting time in queue for the ith packet is the sum of service time of any previous packets which arrive before ith packet and the Residual Service time (RS_{ij}). RS_{ij} represents the remaining time seen by the new packet when it arrives until current in-service packet is complete. Let us denote $N_Q(b)$ the number of data packets waiting in the queue which are encoded

[2]Little's Theorem [26] denotes the average number of packets in the queue is the product of packet arrival rate and average time a packet kept in the queue when the network is in a steady state

with bit rate b. Such that, the expected queueing delay $\mathbb{E}[Q_{ij}]$ is,

$$\begin{aligned}\mathbb{E}[Q_{ij}] &= \mathbb{E}[RS_{ij}] + \sum_{b=1}^{B} \mathbb{E}[N_Q(b)]\mathbb{E}[\varphi_{ij}(b)] \\ &= \mathbb{E}[RS_{ij}] + \sum_{b=1}^{B} \mu_i(\cdot, b)\mathbb{E}[Q_{ij}]\mathbb{E}[\varphi_{ij}(b)] \\ &= \mathbb{E}[RS_{ij}] + \sum_{b=1}^{B} \rho_{ij}(b)\mathbb{E}[Q_{ij}] \\ &= \frac{\mathbb{E}[RS_{ij}]}{1-\rho_{ij}}\end{aligned} \tag{3.12}$$

To apply in multiclass scenario, $\mathbb{E}[RS_{ij}]$ [6] is,

$$\mathbb{E}[RS_{ij}] = \sum_{b=1}^{B} \frac{\mu_i(\cdot, b)\mathbb{E}[\varphi_{ij}(b)^2]}{2} \tag{3.13}$$

Because service time of data packets, $\varphi_{ij}(b)$ shown in Equation 3.10, is a deterministic value, I will have $\mathbb{E}[\varphi_{ij}(b)^2] = \mathbb{E}[\varphi_{ij}(b)]^2$. Therefore, synthesis Equation 3.11 3.12 3.13, queueing delay in 3.7 is proven. $\square$

3.2 Problem Solution

My objective in *DASCache* is to find the optimal video content placement, in every round, which minimizes the average access time per bit to all users and thereby increases their respective throughput. A solver is called at the end of every round, as highlighted in Section 3.1.1, based on aggregated popularity predictors and changes in traffic, to find an optimal cache assignment. In order to formulate a feasible optimization problem, I explain a method in this section which transforms the objective given in Equation 3.5 to a binary integer programming (BIP) problem.

Synthesizing Equation 3.2 and 3.5, I apply the '**big-M**' method (' **M**' denotes a very large positive number) to obtain a standard form of binary integer programming by substituting $\max_{m=1,\ldots,i} x_{L_j(m)}(k,b)$ in Equation 3.2 with an artificial binary variable $m_j(i)$ and add another two constraints in Equation 3.18. My optimization problem is finally formulated as follows,

$$\min \quad \sum_{j=1}^{|N|} \sum_{k=1}^{K} \sum_{i=1}^{|L_j|-1} \frac{\mathbb{E}[D(L_j(i), L_j(i+1))](1-m_j(i))}{|N|S(b_j)} \tag{3.14}$$

$$s.t. \quad \forall j \in [1..|N|], \quad \forall l \in [1..|V|], \quad \forall i \in [1..|L_j|-1] \tag{3.15}$$

$$b_j = arg(\pi_j(b) = 1) \tag{3.16}$$

$$m_j(i) = \max_{m=1,\ldots,i} x_{L_j(m)}(k, b_j) \tag{3.17}$$

$$m_j(i) \leq \begin{cases} x_{L_j(i)}(k, b_j) + \mathbf{M} n_j(i) \\ m_j(i-1) + \mathbf{M}\left(1 - n_j\left(i\right)\right) \end{cases} , \forall i > 1 \tag{3.18}$$

$$\sum_{k \in K} \sum_{b \in B} W_l(b) x_l(k,b) \leqslant C_l, \tag{3.19}$$

$$W_l(b) = \begin{cases} S(b_j) & if \quad l \in L_j \\ +\infty & if \ not \end{cases} \tag{3.20}$$

$$m_j(i), n_j(i), x_l(k,b) \in \{0,1\} \tag{3.21}$$

In the optimization constraints above, Equation 3.16 corresponds to my assumption that users under the service of a common edge router request for the same bit rate. Equation 3.19 constitutes the cache capacity constraint in optimization which ensures the size of cached video objects cannot exceed the upper limit (C_l). $W_l(b)$ represents required storage space to assign a video chunk encoded with bit rate b to node l in L_j. However, $W_l(b)$ is only equal to video chunk size on those routers which interest requests for bit rate b ever reach. If not, $W_l(b)$ is set infinity to avoid unnecessary placement, as shown in Equation 3.20.

Note that in this problem, cache configuration matrix (x_j) and artificial variables ($m_j(i)$) are both need to be solved. However, in Equation 3.14, we can see only the artificial variables have non-zero coefficients. To ensure $\mathbb{E}[D(L_j(i-1), L_j(i))]$ in 3.14 is a constant value, the optimization must run iteratively such that the entire problem could be solved as a linear programming. The reason for this iteration lies in that: $\mathbb{E}[D(L_j(i-1), L_j(i))]$ is given by Equation 3.6 and 3.7. In Equation 3.7, in order to calculate interest miss rate of each router ($\mu_i(\cdot, b)$), I rely on caching configuration ($x_i(k, b)$) in Equation 3.8. Therefore, I need to know the cache configuration to calculate queueing delay which again influences the cache configuration.

My solution is to run iteratively, applying the result of $x_i(k, b)$ in the previous run to queueing model and achieving a deterministic value of $\mathbb{E}[D(L_j(i-1), L_j(i))]$. Subsequent runs constantly tune the cache configuration and this algorithm will stop if two consecutive runs generate the same/close results.

My algorithm provides a new caching strategy, which takes network communication and storage resources into consideration. Accordingly, each client's demand is satisfied with the minimal access time per bit to maximize their QoE. My solution presents a benchmark since it gives the optimal average access time.

Chapter 4

Performance Analysis

In this chapter, I evaluate the performance of my *DASCache* strategy using simulation. I build a simulation model in order to mimic the scenario that users request for streams encoded with multiple bit rates. I also develop testing cases in order to observe the behaviour of caching strategy influenced by different factors.

4.1 Simulation Tool

The tools used in the experiment are Gurobi optimization solver [19] and Network Simulator 3 (NS-3) with extended support on CCN architecture, ndnSIM [1].

4.1.1 NS-3 and ndnSIM

NS-3 is a discrete time, event driven network simulator. It builds a solid simulation core which supports sufficient real-world network protocols from different layers, such as UDP/TCP, IP and from different network types, such as, Wi-Fi, WiMAX, LTE, etc. The entire simulator is written in C++ with object-oriented design and each protocol is encapsulated as a class or module. NS-3 is well documented and its tracing function caters to the need of easy-debugging, problem diagnosis and result analysis. NS-3 also separates simulation configuration from protocols implementation which makes it convenient for users to adjust parameters, testing different scenarios.

The NS-3 simulation core supports research on not only IP based network, but also non-IP network which leads to ndnSIM. ndnSIM is a a new network-layer module in NS-3 which implements Content-Centric Network (CCN) architecture and can seamlessly work on top of other underlying protocols (such as point-to-point media).

To setup and run simulation over ndnSIM (or NS-3 in general), users must initiate NS-3, including the modules which want to be used to build the network topology and install protocol stacks on nodes. Through configuring the event scheduler, users can control when to start and stop transmitting packets.

4.1.2 Gurobi

Gurobi [19] is the state-of-the-art programming solver, particularly optimized for solving linear programing (LP), integer programming (ILP), etc. Compared with other solvers, Gurobi is designed to run in parallel which fully utilizes the computation capability of processors thus accelerates searching the optimal results. It provides various versions of API from Java, C++ to Python. To setup a solver, users need to explicitly designate coefficients of variables and choose parameters to control the searching for optimal value. For example, users may provide their own heuristics in order to more efficiently cut the searching area, or control the threshold to stop searching when a suboptimal result is satisfactory in a bounded time.

4.2 Simulation Model

This section describes the simulation model I use to test the performance over Content-Centric Network.

4.2.1 Simulation Process

I conduct the experiment in two phases. In the first phase, I implement an ILP solver in which calls Gurobi engine to solve the problem of caching video chunks with multiple bit rates on the optimal routers. The second phase is to conduct simulation through ndnSIM. By customization on caching policy of ndnSIM, I create my own policy that, as to any router which locates on the optimal position for video chunks designated in the first phase, it keeps the chunks in the Content Store for the whole simulation period.

4.2.2 Simulation Topology

Due to the complexity of my *DASCache* caching policy, I limit the size of ISP to 60 routers and assume there is only one gateway which connects directly with video repository. As mentioned in the system model, after applying the default forwarding strategy, tree-like topology could be abstracted. Therefore, I generate a tree topology randomly following three constraints: tree height, maximum out degree and the number of leaf nodes. There could be multiple users connecting to a edge router simultaneously. However, since I have assumed that each user under the service of the same edge router will request for the same bit rate, I create one *superuser* connecting directly with edge router to represent all users by integrating their requests together.

4.2.3 Experiment Setting

I configure my ndnSIM simulation environment using all default settings of CCN project except my own caching policy.

As mentioned in the system model, each video chunk is requested with a certain frequency which distinguishes between popular and unpopular contents. This frequency should be collected by ISP provider in order to make a caching decision. In the simulation, each video is requested with a probability according to a Zipf distribution [7]. Such that, a *superuser* will subscribe for streams following that probability value to reflect the request frequency of different chunks. The Zipf distribution is to assign the video segment that ranks at i with $p_i = (\beta i^\alpha)^{-1}$, where $\beta = \sum_{i=1}^{|N|} \frac{1}{i^\alpha}$. Here, α represents the skewness factor. A large α means less video chunks have similar popularity value while small α means that a large number of contents share similar popularity value. The distribution of Zipf with different α is shown in Figure 4.1.

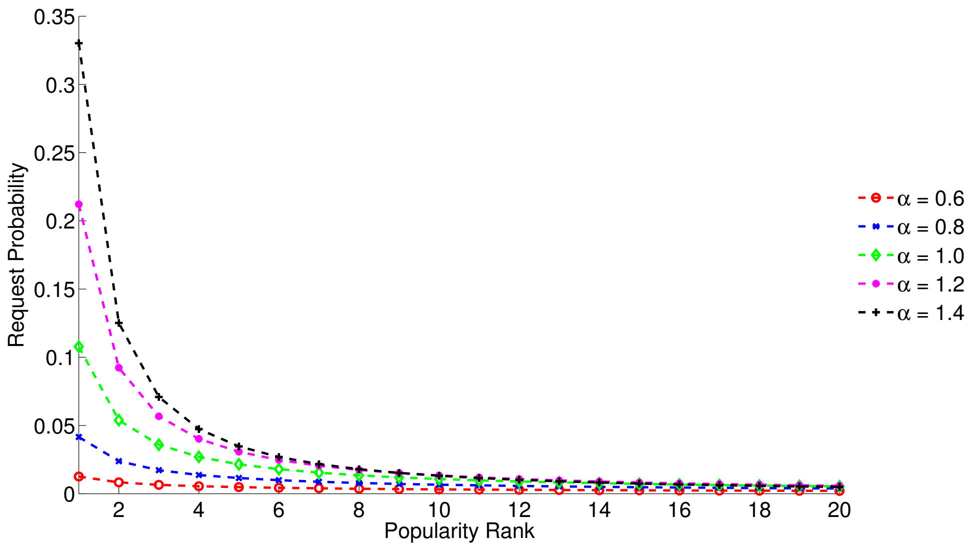


Figure 4.1: Zipf Distribution

Five available bit rates are considered in my simulation. The size of video chunks for these bit rates is determined according to sample videos [44] from Youtube. I also set the lifetime of interest packet same as the video chunk period (10 seconds) to mimic the scenario that if the data delivered overtime, the dynamic adaptive strategy would switch to a lower bit rate and previously requested data would be discarded. The parameters that are constant throughout all simulations are listed in Table 4.1. Parameters that vary will be discussed in Section 4.3.

Table 4.1: Simulation Environment Parameters

Parameters	Values
Number of video chunks (K)	6000
Video period	10 seconds
Number of routers in ISP ($\|V\|$)	60
Max out degree	5
Propagation delay (P)	5 milliseconds
Available bit rates (B)	5
Size of bit rate chunks (S)	{332, 390, 454, 1364, 2465}KB
Bandwidth between gateway and video repo	1 Gbps
Bandwidth between routers within ISP	20 Mbps
Simulation period	1500 seconds

4.2.4 Request Generation

As what I are proposing is a strategy for caching in ICN, not for dynamic adaptive streaming, in order to simplify the simulation, I choose not to use a real DASH client but design an application, generating requests for streams with an appointed bit rate which is regarded as what the dynamic adaptive streaming recommends.

To decide this bit rate and assign it to the *superuser* who will request, the network bottlenecks must be considered and the traffic cannot exceed the upper limit of available bandwidth. Generally, the network bottleneck could be any link within the ISP or the 'last mile'. The 'last mile' refers to the link between edge router and user's device. In my simulation, I only present results where the bottleneck is within ISP. The reason why I do not consider the 'last mile' is elaborated in Section 4.3.5.

Thus, I make the following request generation rule of three steps:

- I Random bit rate is chosen and assigned to each *superuser* and take it as the predicted one.

- II Choose initial average request rate (λ) for *superuser* to ensure that the traffic load on the link is close to full load between edge router and its direct parent.
- III Calculate traffic load (φ) according to Equation 3.11 over all the links within the topology. If $\varphi < 1$ is not satisfied on any link, average request rate (λ) of clients whose interest/data packets pass through that link must be declined proportionally.

Repeat step III. until there is no violation on all links. Applying this rule, the traffic load on some links in the network will be close to full and become the bottlenecks. These bottlenecks will prevent users from switching to streams with better quality which makes the chosen bit rate reasonable in the simulation. If users switch to higher bit rate, the bottlenecks will be overloaded which results in a large amount of packet drop. Thus, the throughput could be extremely low which forces adaptive strategy to switch the video quality back.

4.2.5 Performance Metric

My *DASCache* caching strategy targets on optimizing *access time per bit*. However, the ultimate goal is to improve the throughput measured by users which increases the chance of switching to streams with better quality. Therefore, *Average Throughput* is used as the performance metric in my analysis.

To get this value, first, I calculate the average throughput per *superuser* by having video chunk size divided by average data packet retrieval time ($\frac{S}{\bar{T}}$) where $\bar{T}$ is measured by two timestamps between the departure of interest packet and the arrival of corresponding data packet at user's device. Next, *Average Throughput* is achieved by taking average over all *superuser*s.

Since I need to measure the RTT of users' requests to calculate throughput, the delay on 'last mile' must be appointed in addition to the experiment settings above. As the dynamic adaptive streaming strategy will ensure that users only request for what the network could support, I set the RTT on 'last mile' 0.5 second. This is an important value which will influence the performance dramatically. I will show the effect of this variable and further analysis in Section 4.3.5.

4.2.6 Performance Comparison

As my *DASCache* method relies on collected statistics and only refreshes cache at the beginning of a new round. In order to make fair comparison, I use periodic LRU (P-LRU) and periodic LFU (P-LFU). P-LRU is modified version of classic Least Recently Used (LRU) scheme but the replacement only occurs at the beginning of new round based on the request sequence it records in the previous round. Similar change is made to P-LFU which is based on Least Frequently Used (LFU) scheme.

4.3 Simulation Results

This section shows multiple scenarios to test different factors which may influence the throughput. I study these factors, comparing the performance among *DASCache*, P-LRU and P-LFU.

I adjust the following parameters:

Skewness of Zipf distribution (α) represents the popularity of streams by denoting the probability of requests on each video chunk (p_i). As stated earlier, I have,

$$\beta = \sum_{i=1}^{|N|} \frac{1}{i^\alpha} \tag{4.1}$$

$$p_i = \frac{1/(i^\alpha)}{\beta} \tag{4.2}$$

Cache Capacity Percentage$(\boldsymbol{\omega})$ is to define the total cache capacity percentage which is dedicated to keep multimedia data over all routers within the entire ISP. Given the fixed number of video objects and corresponding size of each bit rate, the cache space for all routers is $\omega \cdot K \cdot \sum_{i=1}^{B} S(i)$. As to each router, the cache capacity is determined by cache allocation ratio (ϵ).

Cache allocation ratio$(\boldsymbol{\epsilon})$ denotes how many times the cache capacity on edge routers is larger than the capacity on intermediate routers. Suppose I already determined ω, the cache space for edge routers C_j and for intermediate routers C_i are calculated as follows,

$$C_i = \frac{\omega \cdot K \cdot \sum_{i=1}^{B} S(i)}{|V| + (\epsilon - 1)|N|} \tag{4.3}$$

$$C_j = \epsilon \cdot C_i \tag{4.4}$$

Tree Height$(\boldsymbol{\eta})$ The height of the tree topology in the simulation.

Leaf Number$(\boldsymbol{\tau})$ The number of edge routers of tree topology in the simulation.

I also conduct one extended experiment that demonstrates the influence of ‘last mile’ delay on the performance, using υ to denote the delay on that link.

In the experiments: testing cache capacity percentage (ω), skewness factor (α) and cache allocation (ϵ), I fix the network topology and test multiple sets of bit rates requested by *superuser*s. When I test the effect of topology (tree height), I only use one set of bit rates; yet examine multiple topologies. All collected data are averaged and shown in 10% confidence interval in the following sections.

4.3.1 The Effect of Cache Capacity Percentage

How much capacity the routers could use to cache multimedia data is controlled by Cache Capacity Percentage (ω). This section describes the effect of this factor on average throughput.

4.3.1.1 Base Experiment

In this experiment, I run tests over five percentage values and choose homogeneous cache allocation ($\epsilon = 1$) which means that the storage for keeping streaming data is equally split over routers within the ISP. This is a base experiment where I evaluate performance under different size of cache dedicated for multimedia data. Table 4.2 lists all the parameters used in the experiment.

Table 4.2: Parameters for Base Cache Capacity Experiment

Parameters	Values	Description
Skewness Factor (α)	1.2	
Allocation Ratio (ϵ)	1.0	
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	0.5%, 1%, 2%, 5%, 10%	

Figure 4.2(a) shows the average throughput improvement as the the cache capacity allocated for multimedia data increases among all three caching strategies. My *DASCache* approach outperforms two other caching schemes. For example, when $\omega = 2\%$, *DASCache* has a 9.2%, 37.1% improvement over P-LFU, P-LRU respectively. Among these three strategies, P-LRU yields the worst performance in the experiment. Since P-LRU records the most recent subscriptions from users during the last round, it cannot distinguish between subscriptions which are mixed with popular and unpopular requests thus degrades the performance. The outcome of P-LFU and *DASCache* are close, because LFU is also a popularity-based scheme. Through caching the most popular contents at any router, LFU caters to most of the requests which improves the average throughput significantly. However, the reason for the performance difference between P-LFU and *DASCache* lies in the fact that *DASCache* optimizes the data delivery considering the delay and storage utilization in a synthesized manner. For example, it is possible for *DASCache* not to keep the most popular content, saving the cache storage for multiple less popular contents with lower bit rates (smaller size) while caching the most popular content at the parent node of the current router to satisfy more requests from sibling nodes, which thereby, achieves better cache utilization.

Observed from the simulation result in the Figure 4.2(a), with larger cache capacity, the average throughout increases at a lower speed. For example, when ω doubles, increasing from 0.5% to 1%, from 1% to 2% and from 5% to 10%, the performance of *DASCache* improves 24.8%, 15.7% and 8.5% respectively. This results from the fact that larger cache capacity could satisfy more interest requests for popular contents but the new contents kept in increased cache space are requested less often than previous ones. In other words, enlarging cache space could generate obvious performance improvement only when previous space is small. Otherwise, since contents in previous large cache space could already answer most of popular requests, the improvement will be limited.

All three caching schemes provide much improvement compared to the no cache situation (around 2,000 Kbps, which I do not show in the figure). It proves the importance of caching in terms of QoE.

4.3.1.2 Experiment with Larger Cache Allocation Ratio

This experiment tests the scenario where more cache size is allocated on edge routers than intermediate routers over different total cache capacity. Table 4.3 lists all the parameters chosen for this experiment.

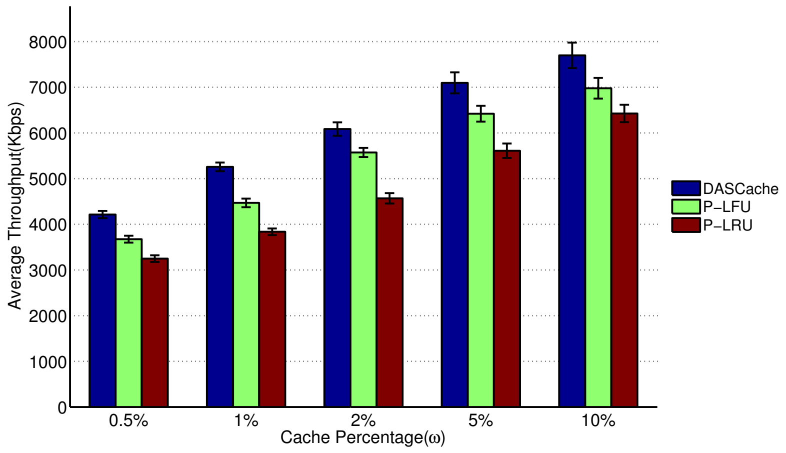
Table 4.3: Parameters for Cache Capacity Experiment With Larger Size on Edge Routers

Parameters	Values	Description
Skewness Factor (α)	1.2	
Allocation Ratio (ϵ)	**5.0**	Changed from 1.0 to 5.0
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	0.5%, 1%, 2%, 5%, 10%	

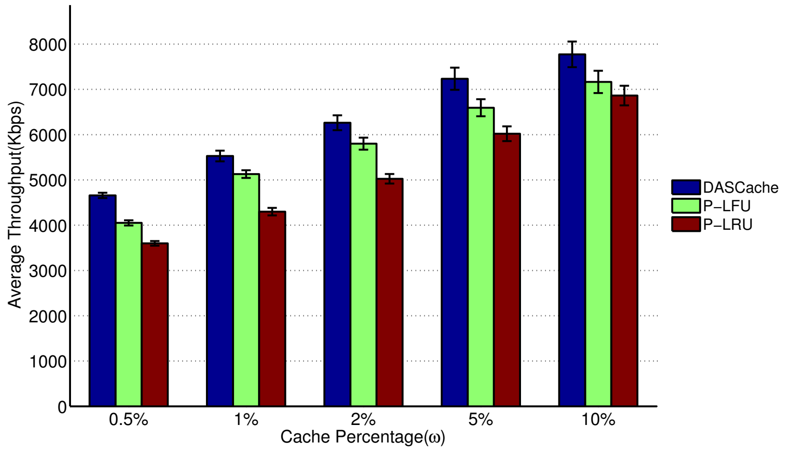
From Figure 4.2(b), I can see that all three caching strategies behave in a similar manner to the experiment conducted with $\epsilon = 1$. At each cache percentage (ω) point, my *DASCache* approach still outperforms the other two strategies. By comparing scenarios between $\epsilon = 1$ and $\epsilon = 5$ as shown in Figure 4.3, I observe that the performance at $\epsilon = 5$ is slightly better than $\epsilon = 1$ in terms of average throughput. This is because more requests could be satisfied closer to users with larger capacity on edge nodes. However, such improvement is negligible. The reason is that, under current setting of skewness factor ($\alpha = 1.2$), even though there are more cache hits occurred on edge routers at $\epsilon = 5$ compared with $\epsilon = 1$, the total number of cache hits at $\epsilon = 5$ is even less than $\epsilon = 1$ as seen in Table 4.4. Although some requests could be responded to faster, being closer to users, which produces a positive effect on the performance with larger cache space on edge routers, content redundancy degrades the cache utilization which means some previously cached contents, now must be retrieved from the server thus offsetting the positive effect.

Table 4.4: Average Number of Cache Hits Over Different Total Cache Capacity

ω	0.5%	1%	2%	5%	10%
$\epsilon = 5.0$	13,644	16,224	18,349	20,461	21,374
$\epsilon = 1.0$	13,836	16,453	18,656	20,761	21,669



(a) Base Experiment



(b) $\epsilon = 5.0$

Figure 4.2: Effect of Cache Capacity Percentage

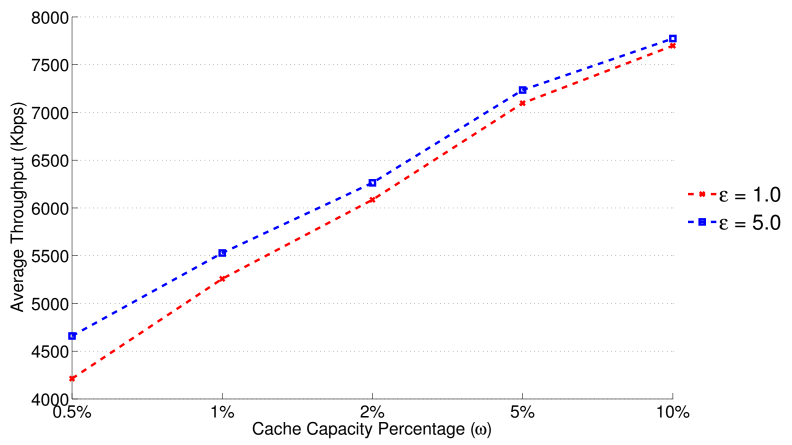


Figure 4.3: Cache Allocation Over Different Total Cache Capacity

4.3.2 The Effect of Skewness Factor

There is no doubt that users' requests will influence the behavior of caching in ICN. I choose Zipf distribution to represent the popularity of video objects. The request probability of each video object is affected by skewness factor (α). This section shows the effect of this value.

4.3.2.1 Base Experiment

In this experiment, I fix the cache capacity percentage ω to be 2% and test over α among 0.6, 0.8, 1.0, 1.2, 1.4, 2.0. This is a base experiment where I test multiple access patterns made by users with various α values under simple cache allocation and fixed total cache size. Table 4.5 lists all parameters used in the simulation.

Table 4.5: Parameters for Base Popularity Experiment

Parameters	Values	Description
Skewness Factor (α)	0.6, 0.8, 1.0, 1.2, 1.4, 2.0	
Allocation Ratio (ϵ)	1.0	
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	2%	

Figure 4.4(a) indicates that my *DASCache* strategy achieves the best average throughput compared with P-LRU and P-LFU schemes among all tested values of α. For example, when $\alpha = 0.8$, my strategy provides improvement of 15.0% and 35.9% compared with P-LRU and P-LFU respectively. When popularity skewness increases, I observe that all caching schemes achieve significant performance improvement. The reason for this phenomenon is that the larger skewness means fewer contents will be requested frequently. For example, the total number of video objects which may be requested with the probability of first 50% is 1,126 when $\alpha = 0.6$. However, as $\alpha = 2.0$, the most popular video (ranked first) already will be requested with probability more than 50%. Therefore, even if the total cache capacity remains unchanged, the amount of popular data is smaller as α is larger, thus more of popular video objects can be kept in the cache and respond to requests at a faster speed.

4.3.2.2 Experiment with Larger Cache Allocation Ratio

This experiment tests the scenario where more cache size is allocated on edge routers than intermediate routers over various skewness values of Zipf distribution. Table 4.6 lists all parameters used in the simulation.

Table 4.6: Parameters for Popularity Experiment With Larger Size on Edge Routers

Parameters	Values	Description
Skewness Factor (α)	0.6, 0.8, 1.0, 1.2, 1.4, 2.0	
Allocation Ratio (ϵ)	**5.0**	Changed from 1.0 to 5.0
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	2%	

From Figure 4.4(b), I see that all three caching strategies generate very close results compared with the experiments conducted with $\epsilon = 1$ on very α point. Figure 4.5 compares the the performance of *DASCache* with base experiment, the result shows that the difference enlarges as α is larger and achieves a slightly better average throughput with $\epsilon = 5.0$ since more requests could be satisfied closer to users.

Table 4.7 and 4.8 reveals the insight of this observation by listing statistics of cache hits. The ratio in the table is the number of cache hits which occurred on edge routers versus on all routers. When $\alpha = 2.0$, the ratio is 98.2% and 99.6% respectively as $\epsilon = 1.0$ and $\epsilon = 5.0$. Since most of requests have already been satisfied on one-hop away edge routers as $\epsilon = 1.0$, the limited improvement when $\epsilon = 5.0$ is expected. However, when α is a small value (such as 0.8), the ratio of cache hit increases from 73.3% to 91.6% which is regarded as a big improvement but the average throughput still remains unchanged. From these results, I observe that the total number of cache hits with $\epsilon = 5.0$ is lower than $\epsilon = 1.0$. Such result is due to cache redundancy. Since in the simulation, all *superuser*s will request following the same Zipf distribution and the total cache space is fixed, as each edge router is allocated more capacity, contents which are previously cached on intermediate nodes could be kept closer to subscribers but these objects have to be duplicated among edge routers. In contrast, smaller cache capacity on edge routers forces some popular contents to be stored on the common parent of edge routers which saves cache capacity for perhaps less popular contents by sharing among all its children nodes. Particularly when α is small, as the difference of request frequency between streams is also small, more cached contents on edge routers ($\epsilon = 5.0$) not only cannot satisfy as many requests as α is high but loses space to cache maybe-not-the-most-popular contents.

Table 4.7: Average Number of Cache Hits Over Different Popularity Skewness ($\epsilon = 5.0$)

α	0.6	0.8	1.0	1.2	1.4	2.0
Hit on Edge Nodes	2,637	6,232	11,759	17,708	22,103	26,610
Hit on Any Nodes	3,016	6,804	12,457	18,349	22,525	26,702
Ratio	87.4%	91.6%	94.4%	96.5%	98.1%	99.7%

Table 4.8: Average Number of Cache Hits Over Different Popularity Skewness ($\epsilon = 1.0$)

α	0.6	0.8	1.0	1.2	1.4	2.0
Hit on Edge Nodes	2,155	5,349	10,528	16,366	21,009	26,283
Hit on Any Nodes	3,414	7,296	12,920	18,656	22,707	26,757
Ratio	63.1%	73.3%	81.5%	87.7%	92.5%	98.2%

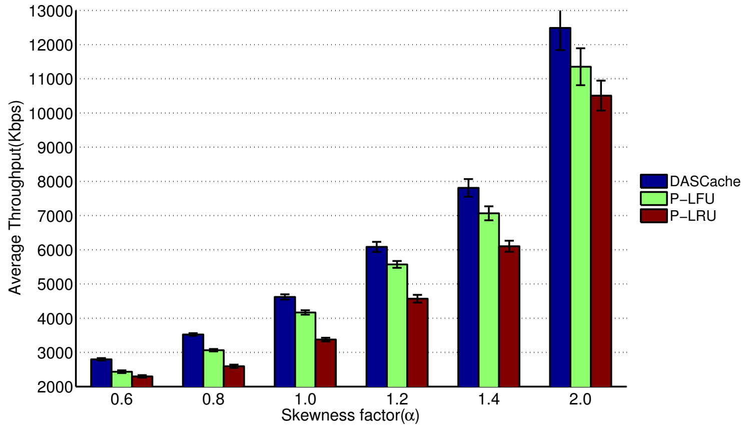
4.3.2.3 Experiment with Smaller Cache Capacity Percentage

This experiment tests the scenario where smaller capacity is dedicated for multimedia caching in ICN over various skewness values of Zipf distribution. Table 4.9 lists all parameters used in the simulation.

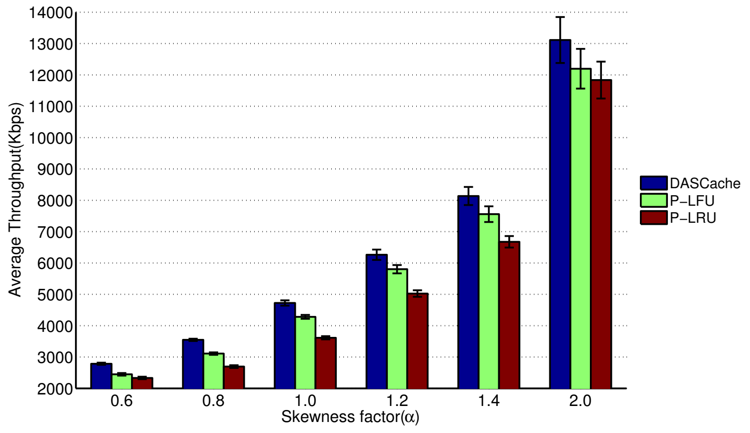
Table 4.9: Parameters for Popularity Experiment With Smaller Total Cache Capacity

Parameters	Values	Description
Skewness Factor (α)	0.6, 0.8, 1.0, 1.2, 1.4, 2.0	
Allocation Ratio (ϵ)	1.0	
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	**0.5%**	Changed from 2% to 0.5%

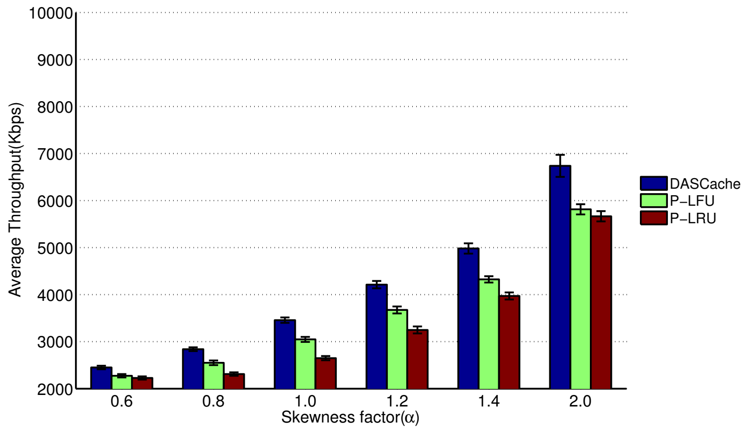
From Figure 4.4(c), I can tell that less total cache capacity will obviously decrease the average throughput over all skewness factor points. Compared with base experiment as shown in Figure 4.6, the performance difference is more obvious with a higher popularity skewness value. For example, as $\alpha = 0.6$ and ω changes from 0.5% to 2%, my *DASCache* strategy achieves a 14.1% improvement but when $\alpha = 2.0$, the improvement augments to 85.3%. This is because the cached content in augmented space could answer more requests (more cache hits) with a larger α.



(a) Basic Experiment



(b) $\epsilon = 5.0$



(c) $\omega = 0.5\%$

Figure 4.4: Effect of Skewness Factor

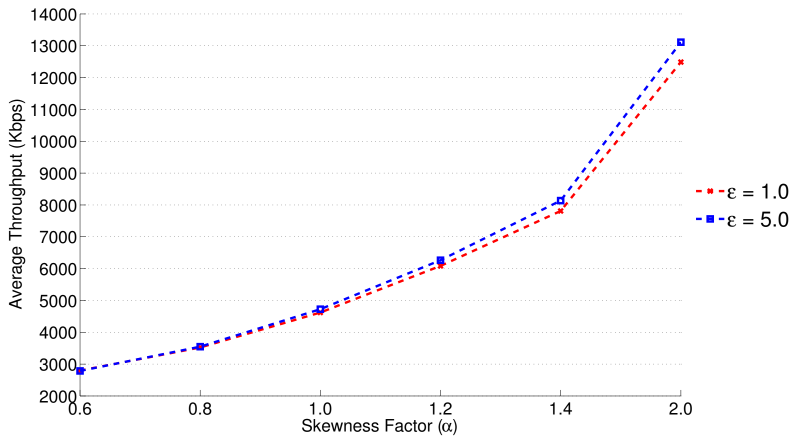


Figure 4.5: Cache Allocation Over Different Popularity Skewness

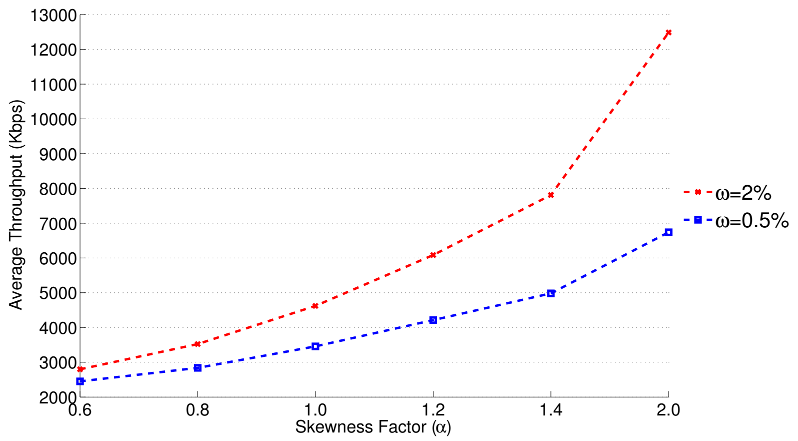


Figure 4.6: Total Cache Capacity Over Different Popularity Skewness

4.3.3 The Effect of Cache Allocation

With fixed total cache capacity dedicated for multimedia data, how to arrange this capacity over routers is still a problem and may influence the performance of caching. Section 4.3.1.2 and 4.3.2.2 have already given analysis under different ω and α. This section studies performance under various cache allocation patterns.

4.3.3.1 Base Experiment

I conduct this experiment over five ϵ values to represent different allocation patterns. Table 4.10 shows all parameters used in the simulation.

Table 4.10: Parameters for Base Cache Allocation Experiment

Parameters	Values	Description
Skewness Factor (α)	1.2	
Allocation Ratio (ϵ)	0.2, 0.5, 1.0, 2.0, 5.0	
Tree Height (η)	4	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	2%	

Shown in Figure 4.7, I observe that my *DASCache* strategy performs the best under various ϵ values. When the cache capacity on edge nodes enlarges, average throughput is improved but the performance difference is still small. The underlying reason is the same to the analysis done in section 4.3.2.2 that redundant cached contents exhaust space for caching new objects which neutralizes the benefit of moving more contents closer to subscribers. Therefore, by testing using various values of α, ω and ϵ, I can conclude that cache allocation is not a big concern under my simulation scenario. However, it is worth noting that, based on the setting I made that the bandwidth of links within ISP are all the same (which is similar to the real case), the traffic congestion is likely to occur close to the gateway. If the bandwidth of links varied and congestion appeared close to the edge router, it could be expected that large cache space of edge routers will produce much more significant improvement of average throughput than current setting.

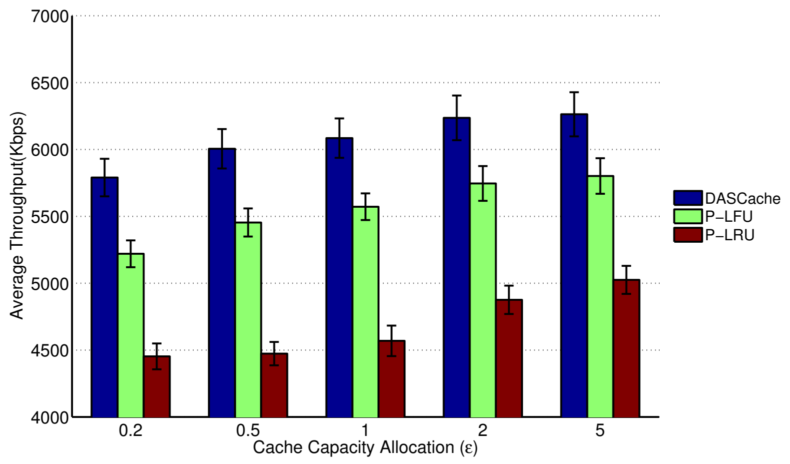


Figure 4.7: Effect of Cache Allocation Ratio

4.3.4 The Effect of Topology

Performance evaluation in above sections are under various bit rate choices which simulates different network conditions. In this section, I turn my attention to the effect of network topology on all three caching strategies.

There are two variables, tree height (η) and the number of edge routers (τ) which are considered in my experiments. To test the effect of tree height, I fix one set of bit rate choices from *superuser*s and randomly select topologies to run the simulation. To test the effect of number of edge nodes, I randomly choose one topology but test multiple sets of bit rate choices and compare the results with previous experiments.

4.3.4.1 The Effect of Tree Height

I conduct this experiment over topologies with four different tree heights but maintain the total number of caching routers. This can represent the scenario where caching for multimedia data is applied over various ISPs with similar topology size but different maximum hop counts to users. Table 4.11 lists all parameters used in the experiment.

Table 4.11: Parameters for Base Tree Height Experiment

Parameters	Values	Description
Skewness Factor (α)	1.2	
Allocation Ratio (ϵ)	1.0	
Tree Height (η)	4, 5, 6, 7	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	2%	

From Figure 4.8(a), I can see that the performance of all three caching strategies degrades as the tree height raises. The reason is straightforward: since the tree height raises, data packets which are not served from any cached nodes have to travel through a longer path to the server which increases the delay thus leads to lower average throughput. My *DASCache* strategy still leads regarding performance among all test cases.

I also carry out an experiment which tests the scenario where more cache size is allocated on edge routers than intermediate routers over different topology shapes (maximum hop counts). Table 4.12 lists all parameters used in the experiment.

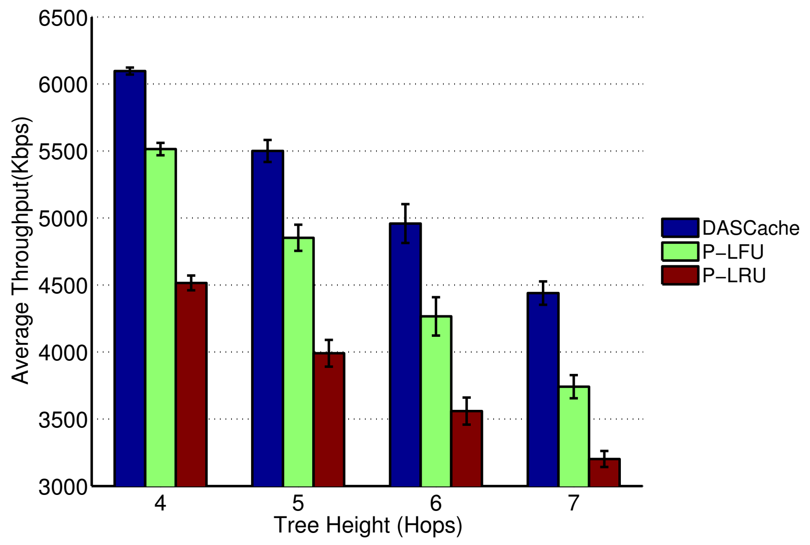
Table 4.12: Parameters for Tree Height Experiment with Larger Size on Edge Routers

Parameters	Values	Description
Skewness Factor (α)	1.2	
Allocation Ratio (ϵ)	**5.0**	Changed from 1.0 to 5.0
Tree Height (η)	4, 5, 6, 7	
Number of Edge Nodes (τ)	35	
Capacity Percentage (ω)	2%	

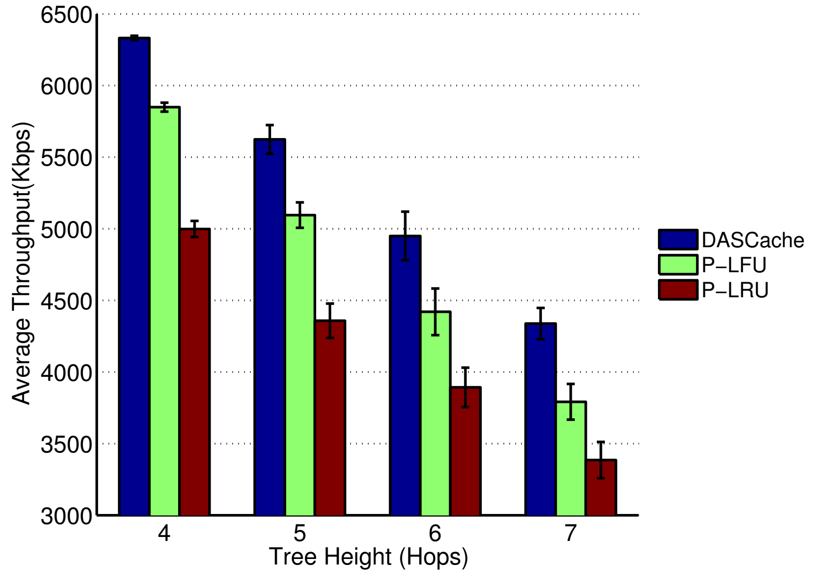
From Figure 4.8(b), I observe a similar trend on performance compared with the base experiment. Larger cache capacity on edge routers does provide better performance as I can see in Table 4.13, but this advantage shrinks as the tree height increases. When $\eta = 7$, the average performance with $\epsilon = 1.0$ even exceeds $\epsilon = 5.0$. As I fix total cache size, the worse cache utilization with large ϵ prevents caching more video objects. Such that, video objects which are missed by cache have to be retrieved from the server. When the tree height raises, the cost of traveling increases correspondingly. The benefit of larger capacity of edge nodes is neutralized further by the cost of extra data packet traveling. I can expect from the trend that, when the tree height is higher, the performance of $\epsilon = 5.0$ could be worse. It supports the conclusion I made in Section 4.3.3 by testing over different topologies.

Table 4.13: Average Throughput between $\epsilon = 5.0$ and $\epsilon = 1.0$

	4	5	6	7
$\epsilon = 5.0$	6332.3	5624.1	4949.8	4339.3
$\epsilon = 1.0$	6096.7	5500.3	4958.8	4439.7
Difference	235.5	123.8	-8.9	-101.4



(a) Basic Experiment



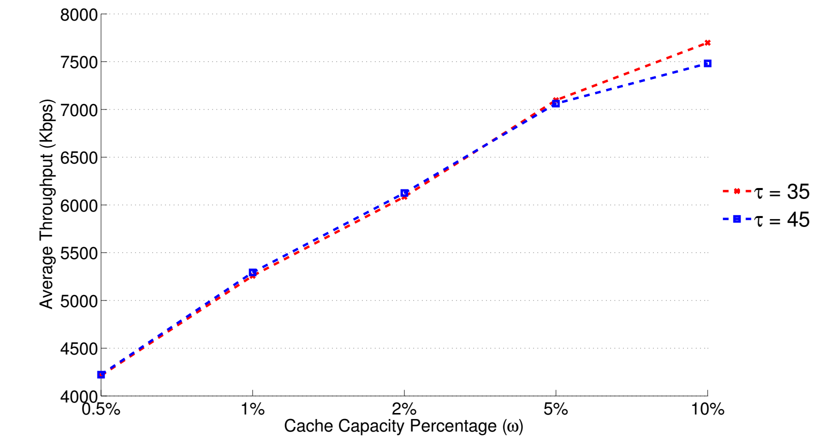
(b) $\epsilon = 5.0$

Figure 4.8: Effect of Tree Height

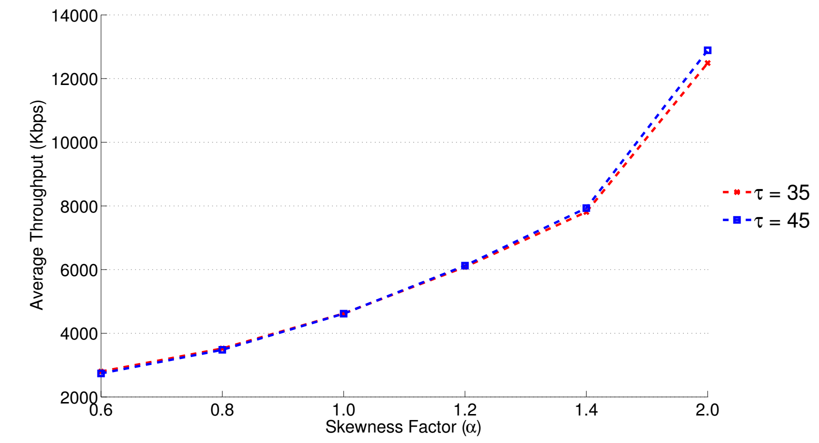
4.3.4.2 The Effect of Number of Edge Nodes

In addition to the tree height, the number of edge nodes (τ) is another factor which controls the shape of topology. The simulation in this section can represent the case where different numbers of nodes are assigned to serve users directly while other routers are only responsible for *Interest*/*Data* packet delivery. I conduct three experiments using the same parameters as listed in Table 4.2, 4.5 and 4.10 except the τ is set 45.

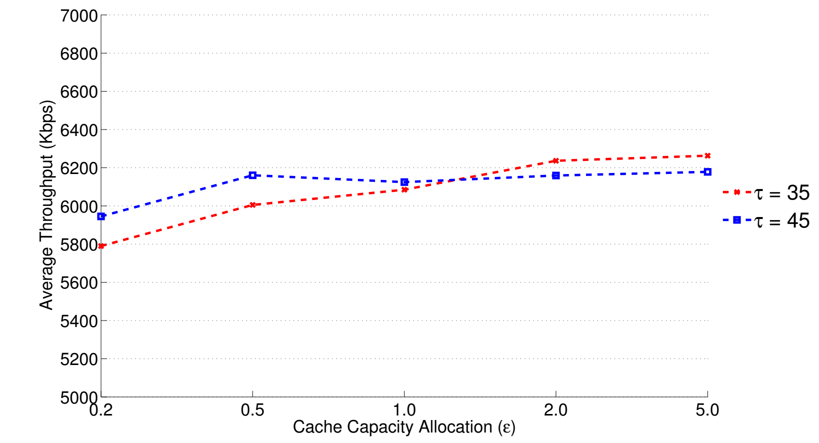
Figure 4.9(a), 4.9(b) and 4.9(c) show the comparison on performance of my *DASCache* management strategy as $\tau = 45$ with all base experiments conducted in above sections as $\tau = 35$. I observe that when I test on cache capacity percentage (ω) or skewness factor (α), the performance curves of experiments conducted with $\tau = 35$ and $\tau = 45$ almost overlap. When I test on cache allocation (ϵ), the curve of $\tau = 45$ is even flatter than $\tau = 35$. Since 75% of the routers in the topology are edge routers as $\tau = 45$, adjusting the cache allocation ratio will incur less changes on the cache space of edge routers than $\tau = 35$. Therefore, I can conclude that the number of edge routers in the topology has no influence on the performance of *DASCache* under my simulation setting.



(a) Comparison with different ω



(b) Comparison with different α



(c) Comparison with different ϵ

Figure 4.9: Effect of Number of Edge Nodes

4.3.5 Extended Experiment: Delay on 'last mile' Link

The delay on 'last mile' (υ) will have a prominent influence on the average throughput. The choice of this value is critical in the simulation. Suppose users had a very bad communication channel via the edge router (*e.g.* a user is far away from signal tower) and thus the 'last mile' delay will be huge. For example, if transmitting a video object takes 99 seconds over 'last mile' and only 1 second over links within ISP, through caching in ICN, I make the delay within ISP 0 second (almost impossible unless cache directly at edge router) which is the best performance the system can provide. The overall improvement will only be 1%. Such that, if the bottleneck happens on 'last mile' link, caching in ICN will not have much effect on performance improvement and it is beyond what optimization can do. In this extended experiment, I show results under different delay on 'last mile'. The parameters used are the same as listed in Table 4.2.

Figure 4.10 presents results with 'last mile' delay of 0.1s 0.5s and 1.0s. The result proves my earlier analysis that υ seriously impacts on the overall performance of caching in ICN. As P-LRU produces the worst performance among three caching strategies, I only generate curves of *DASCache* and P-LFU. Compared with $\upsilon = 0.5s$ and $\upsilon = 0.1s$, the performance difference between *DASCache* and P-LFU is shrinking as $\upsilon = 1.0s$ and the smaller increasing speed of the average throughput also shows the effect of 'last mile' delay: large delay offsets the benefit of caching. Considering the experiment parameters I applied, I finally chose a fixed value of $\upsilon = 0.5s$ and never considered the bottleneck on the 'last mile'.

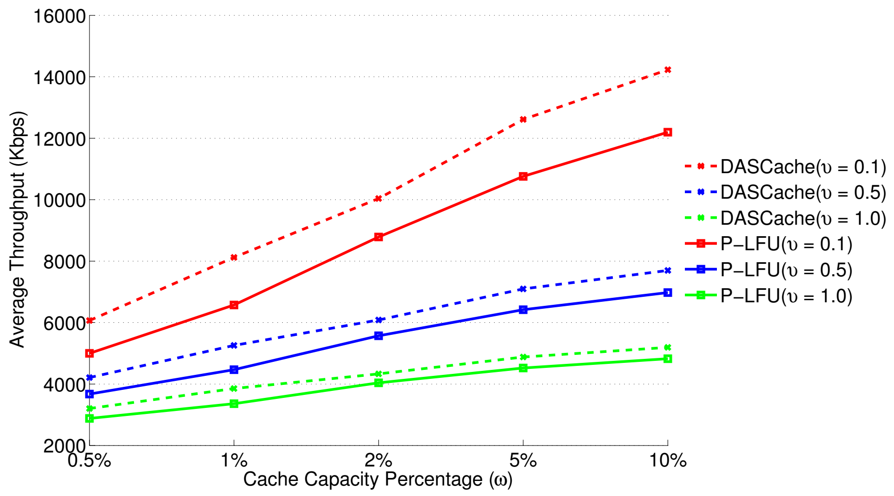


Figure 4.10: Effect of 'last mile' Delay

4.4 Summary

In this chapter, I presented tools and the simulation model used in the experiments; I also tested and analyzed multiple factors which may influence the performance of my *DASCache* management strategy.

According to the simulation results, my *DASCache* strategy outperforms classic LFU and LRU based caching schemes over all conducted experiments which proves the correctness and effectiveness of my method within multiple scenarios, in which bit rate choices and network topologies vary. Among all tested factors which may influence the outcome of my method, I conclude that:

- Allocating more cache space for multimedia data is beneficial in improving the average throughput measure by each user.

- My *DASCache* can dynamically reflect the popularity of contents requested by users. In the experiment, I use Zipf distribution as a study case and my method works very well when the skewness factor is assigned a high value. When skewness value is lower, which means harder to distinguish between popular and unpopular contents, my *DASCache* still maintains the best performance.

- Cache allocation is not regarded as an important factor to influence the performance based on my simulation setting. As the total cache space is determined, more cache space on edge routers decreases the average latency but increases the caching redundancy as well. Such redundancy neutralizes the performance improvement of moving contents closer to users which results in negligible difference among cases, testing multiple allocation ratios.
- Network topology is generated according to the tree height and the number of leaf nodes. As to the factor of tree height, if the maximum routing hop counts increases, the average throughput is expected to decrease since data has to travel through a longer path. However, having different numbers of edge nodes in topology shows almost the same simulation results, which proves it will not influence the overall level of performance.

Chapter 5

Conclusions and Future Work

In this thesis, I proposed my cache management strategy, *DASCache*, which aims to improve the QoE of users over ICN. The novel idea of this method is to achieve the optimal cache assignment considering dynamic adaptive streaming which caters to the increased demand for multimedia data. My proposed strategy, *DASCache*, outperforms two other caching schemes as simulation confirms.

In future work, instead of requiring global knowledge from network provider, I plan to extend my work and develop a distributed algorithm to make the solution more flexible to deal with dynamics of network (emphe.g., frequent bit rate switch by user), and scale with larger networks. As my formulation is based on an LP which is inherently NP-hard, I advocate for extending this work with dynamic heuristics that could operate at quasi-optimality in the large scale. Another challenge lies in existing rate adaptation algorithms not working as expected under ICN, suffering from the serious cache oscillation [18]. Proposing an ICN-aware adaptation strategy that addresses this is necessary in utilizing in-network caching.

Another open topic is related to interest aggregation. In a chunk-based delivery system, like CCN, interest request aggregation is proposed to avoid flooding. It is a mechanism for each router to keep track of unsatisfied requests and discard duplicate ones to decrease unnecessary traffic within the network. When a new interest packet arrives at a router, it will be recorded by the router and then forwarded to the next hop. However, before the corresponding data is sent back to this router, any interest packet for the same chunk will never be dispatched again.

If interest aggregation is considered, the average interval between forwarding two subsequent requests for video chunk indexed by (k, b) at router i will be $RVRTT(i) + \lambda_i(k, b)$ in which $RVRTT$ is denoted as *Residual Virtual Round-Trip Time*. It means the same *Interest* packet could be forwarded by the router again after an average period of time during which the current subscription is satisfied ($RVRTT$) and the next *Interest* has arrived ($\lambda_i(k, b)$). Such that, the interest arrival and miss process on routers within ISP is now a renewal process since the holding times take on a general distribution rather than an exponential one.

Carofiglio *et al.*[8] use *filtering probability* (p_{filt}) to revise the average rate to make the model close to the real scenario. The idea of the method is to filter those incoming interest packets of which the interval is less than $RVRTT$. As the interval between two requests satisfies exponential distribution in a Poisson Process, the filtering probability at router i in my model could be represented as,

$$\begin{aligned} p_{i,filt}(k, b) &= P(\text{interval time} > RVRTT(i)) \\ &= e^{-\lambda_i(k,b)RVRTT(i)} \end{aligned} \tag{5.1}$$

Because of decomposition property of Poisson Process, with interest request aggregation, the interest miss rate in Equation 3.8 will be updated to,

$$\mu_i(\cdot, b) = \sum_{k=1}^{K} p_{i,filt}(k, b)\lambda_i(k, b)(1 - x_i(k, b)) \tag{5.2}$$

Although the simulation result of [8] shows that such a feature has limited impact on $VRTT$, the influence on queueing model still needs further analysis. It is also necessary to experiment under video delivery scenario since each content usually has larger size.

Bibliography

[1] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN, October 2012.

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

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

[4] Somaya Arianfar, Pekka Nikander, and Jörg Ott. On content-centric router design and implications. In *Proceedings of the Re-Architecting the Internet Workshop*, page 5. ACM, 2010.

[5] Martin F Arlitt and Carey L Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking (ToN)*, 5(5):631–645, 1997.

[6] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*, volume 2. Prentice-hall Englewood Cliffs, NJ, 1987.

[7] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM*, volume 1, pages 126–134. IEEE, 1999.

[8] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. Modeling data transfer in content-centric networking. In *Proceedings of the 23rd International Teletraffic Congress*, pages 111–118. International Teletraffic Congress, 2011.

[9] Giovanna Carofiglio, Vinicius Gehlen, and Diego Perino. Experimental evaluation of memory management in content-centric networking. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[10] Antonio Carzaniga and Alexander L Wolf. Content-based networking: A new communication infrastructure. In *Developing an Infrastructure for Mobile and Wireless Systems*, pages 59–68. Springer, 2002.

[11] Antonio Carzaniga and Alexander L Wolf. Forwarding in a content-based network. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 163–174. ACM, 2003.

[12] Cisco. Cisco visual networking index: Forecast and methodology. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html`. Accessed on 2014-12-18.

[13] Cisco. Cisco visual networking index: Forecast and methodology, 2011–2016. *CISCO White paper*, pages 2011–2016, 2012.

[14] AB Ericsson and Edwall Thomas. The network of information:architecture and applications. *FP7-ICT-2009-5-257448/D-3.1*, 2011.

[15] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *Network, IEEE*, 24(2):36–41, 2010.

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

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

[18] Reinhard Grandl, Kai Su, and Cedric Westphal. On the interaction of adaptive video streaming with content-centric networking. *arXiv preprint arXiv:1307.0794*, 2013.

[19] Gurobi. Gurobi optimizer reference manual. `http://www.gurobi.com/documentation/5.6/reference-manual/`. Accessed on 2014-10-15.

[20] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In

Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 1–12. ACM, 2009.

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

[22] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634, 2006.

[23] Stefan Lederer, Christopher Mueller, Benjamin Rainer, Christian Timmerer, and Hermann Hellwagner. An experimental analysis of dynamic adaptive streaming over http in content centric networks. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

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

[25] Zhe Li and Gwendal Simon. Time-shifted tv in content centric networks: The case for cooperative in-network caching. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[26] John DC Little and Stephen C Graves. Little's law. In *Building Intuition*, pages 81–100. Springer, 2008.

[27] Yaning Liu, Joost Geurts, Jean-Charles Point, Stefan Lederer, Benjamin Rainer,

Christopher Muller, Christian Timmerer, and Hermann Hellwagner. Dynamic adaptive streaming over ccn: a caching and overhead analysis. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3629–3633. IEEE, 2013.

[28] Jaime Llorca, Antonia M Tulino, Kyle Guan, Jairo Esteban, Matteo Varvello, Nakjung Choi, and Daniel C Kilper. Dynamic in-network caching for energy efficient content delivery. In *INFOCOM, 2013 Proceedings IEEE*, pages 245–249. IEEE, 2013.

[29] Ying Lu, Tarek F Abdelzaher, and Avneesh Saxena. Design, implementation, and evaluation of differentiated caching services. *Parallel and Distributed Systems, IEEE Transactions on*, 15(5):440–452, 2004.

[30] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, Steven Lim, et al. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(1-4):72–93, 2005.

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

[32] Andrea Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32, 2012.

[33] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[34] Dario Rossi, Giuseppe Rossini, et al. On sizing ccn content stores by exploiting topological information. In *INFOCOM Workshops*, pages 280–285, 2012.

[35] Giuseppe Rossini and Dario Rossi. Evaluating ccn multi-path interest forwarding strategies. *Computer Communications*, 36(7):771–778, 2013.

[36] Henning Schulzrinne. Real time streaming protocol (rtsp). 1998.

[37] Thomas Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[38] Sunand Tullimas, Thinh Nguyen, Rich Edgecomb, and Sen-ching Cheung. Multimedia streaming using multiple tcp connections. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 4(2):12, 2008.

[39] Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. Multimedia streaming via tcp: An analytic performance study. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 4(2):16, 2008.

[40] Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie. Optimal cache allocation for content-centric networking. In *ICNP*, pages 1–10, 2013.

[41] Siyuan Xiang, Lin Cai, and Jianping Pan. Adaptive scalable video streaming in wireless networks. In *Proceedings of the 3rd multimedia systems conference*, pages 167–172. ACM, 2012.

[42] Yuemei Xu, Yang Li, Tao Lin, Guoqiang Zhang, Zihou Wang, and Song Ci. A

dominating-set-based collaborative caching with request routing in content centric networking. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3624–3628. IEEE, 2013.

[43] George Xylomenos, C Ververidis, V Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, K Katsaros, and G Polyzos. A survey of information-centric networking research. 2013.

[44] Youtube. Mpeg-dash/media source demo. `http://dash-mse-test.appspot.com/media.html`. Accessed on 2014-10-15.

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

[46] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, 1980.