Architectures and Algorithms for Content Delivery in Future Networks

Marsa Rayani

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information Systems Engineering) at

Concordia University

Montréal, Québec, Canada

June 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Marsa Rayani**

Entitled: **Architectures and Algorithms for Content Delivery in Future Networks**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Chun-Yi Su

_____ External Examiner
Dr. Ahmed Karmouch

_____ External to Program
Dr. Juergen Rilling

_____ Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Chadi Assi

_____ Thesis Supervisor
Dr. Roch Glitho

Approved by _____
Dr. Chadi Assi
Chair of Department or Graduate Program Director

June. 1, 2022
Date of Defence

_____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Architectures and Algorithms for Content Delivery in Future Networks

Marsa Rayani, Ph.D.

Concordia University, 2022

Traditional Content Delivery Networks (CDNs) built with traditional Internet technology are less and less able to cope with today's tremendous content growth. Enhancing infrastructures with storage and computation capabilities may help to remedy the situation. Information-Centric Networks (ICNs), a proposed future Internet technology, unlike the current Internet, decouples information from its sources and provides in-network storage. However, content delivery over in-network storage-enabled networks still faces significant issues, such as the stability and accuracy of estimated bitrate when using Dynamic Adaptive Streaming (DASH). Still Implementing new infrastructures with in-network storage can lead to other challenges. For instance, the extensive deployment of such networks will require a significant upgrade of the installed IP infrastructure. Furthermore, network slicing enables services and applications with very different characteristics to co-exist on the same network infrastructure.

Another challenge is that traditional architectures cannot meet future expectations for streaming in terms of latency and network load when it comes to content, such as 360° videos and immersive services. In-Network Computing (INC), also known as Computing in the Network (COIN), allows the computation tasks to be distributed across the network instead of being computed on servers to guarantee performance. INC is expected to provide lower latency, lower network traffic, and higher throughput. Implementing infrastructures with in-network computing will helps fulfill specific requirements for streaming 360° video streaming in the future. Therefore, the delivery of 360°

video and immersive services can benefit from INC.

This thesis elaborates and addresses the key architectural and algorithmic research challenges related to content delivery in future networks. To tackle the first challenge, we propose algorithms for solving the inaccuracy of rate estimation for future CDNs implementation with in-network storage (a key feature of future networks). An algorithm for implementing in-network storage in IP settings for CDNs is proposed for the second challenge. Finally, for the third challenge we propose an architecture for provisioning INC-enabled slices for 360° video streaming in next-generation networks. We considered a P4-enabled Software Defined network (SDN) as the physical infrastructure and significantly reduced latency and traffic load for video streaming.

Acknowledgments

My journey was only made possible with the help of family, friends, and colleagues along the way, who urged me to keep my eyes on the prize.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Roch Glitho. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and exciting. His dedication and enthusiasm for his research were contagious and motivational, even during the most challenging times in pursuit of my Ph.D. He has taught me how the Ph.D. journey can transform you into a stronger, more zen individual.

I gratefully acknowledge my Ph.D. committee members, Prof. Chadi Assi, Prof. Jamal Bentahar, Prof. Ferhat Khendek and Prof. Juergen Rilling, for their time, effort, and constructive comments. I would also like to extend my appreciation to the external examiner, Prof. Ahmed Karmouch, for accepting to serve on my Ph.D. thesis committee.

I am very thankful to Prof. Halima Elbiaze for all the enlightening discussions, comments and collaboration. Furthermore, I would like to thank Prof. Diala Naboulsi for her intellectual and emotional support over the years and Dr. Amin Ebrahimzadeh for all the insightful comments and contributions. I am also very grateful to Laurie McLaughlin for helping me to proofread and edit the thesis and my papers throughout these years. It was an honor to have had this opportunity to work with you all.

The members of the TSE lab have contributed immensely to my personal and professional time. This group has been a source of friendships as well as collaboration and fun. I am especially grateful to my TSE group friends who stuck together and reminded each other of the larger picture

these past few years: Vahid Maleki Raee and Sepideh Malektaji.

My special thanks to Narges Atabaki, who has always been there with me from the day my journey started at the Hilton Malaysia to this day, including through the most challenging moments. I am also thankful to my friend Farinaz Rasouli. Your friendship and support over the years have been a blessing, and I am delighted to have you by my side. I am fortunate to have you all amongst my friends and in some of my best memories.

I am very thankful to Vida Gholipour and Ahmad Jamshidi. You were the first familiar friendly faces I met in Montreal when I came here years ago, and I will never forget how welcome you made me feel from the outset. Starting this journey without your support and kindness would have been daunting. While it is impossible to name every friend who has made a positive impact on my journey, I wish to thank, in particular: Razieh Abbasi, Bahareh Khorsand, and Edris Ghalishoorani.

I owe a special thanks to my brothers and my Aunt Mina who supported me and helped me to overcome my sadness at not being able to be physically present for my mother and father in the toughest times. To my mother Zhila and my father Majid, I owe you an immeasurable debt of gratitude for the long, hard hours you worked throughout our youth to allow me and my brothers to follow our dreams. Words cannot express how grateful I am for all of the tremendous love you've shown us along the way. I dedicate this thesis to you, mom and dad.

And most of all, my loving, supportive, encouraging, and patient husband, Ali Etemad, whose faithful support during this Ph.D. is so appreciated. Thank you for always being my best friend by my side. Without you, none of this would have been possible. I also want to thank my dog, Toffee, whose unconditional love helped me to stay positive and get through the pandemic years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Overview

The tremendous growth of video traffic is bringing new challenges (e.g., scalability, efficiency in content distribution) that current CDNs are less and less able to meet. CDNs are designed to deliver requested content at a reasonable cost and at the required QoS; however, the current CDNs face performance, reliability, and scalability challenges [5]. The ICN has emerged as an alternative to the current IP-based Internet architecture. It offers features such as in-network storage that may help to meet the challenges faced by the current CDNs by enhancing the infrastructure [6]. In addition, requests for content in ICN-based CDNs can be served by several routers that are closer to end-users than surrogate servers, thereby enhancing performance and scalability.

Unfortunately, ICN-based CDNs still face many challenges. For instance, the rate-based DASH framework currently in use in CDNs does not function well in storage-enabled infrastructure [7]. In DASH, video files are encoded with different qualities or bitrates. DASH enables clients to adapt to different network conditions by requesting appropriate video segments. A DASH client computes the Round-Trip Time (RTT) of each segment it requests and receives and then uses it to estimate the RTT of the next segment. The quality requested for the next segment is decided accordingly. In IP networks, the assumption is that all segments are retrieved from the same server. However,

this assumption does not hold in an ICN, because segments might be retrieved from intermediate routers due to the presence of in-network storage in ICN routers. Furthermore, the ICN in-network storage feature can introduce oscillations in rate estimation thus lead to unstable streaming. The European Telecommunication Standards Institute's (ETSI) Multi-Access Edge Computing (MEC) paradigm can tackle this issue. MEC provides cloud computing capabilities at the edge of mobile networks [8]. Network-assisted rate estimation on the MEC server can help solve inaccuracy and instability issues for rate-based adaptive streaming over core networks with in-network storage.

Yet another challenge is that the extensive deployment of ICN-based CDNs is very costly for infrastructure providers, as it requires a significant changes, including replacing the installed IP routers with ICN routers that have in-network storage. Network Slicing can address the aforementioned challenge by enabling the co-existence of virtual networks with different characteristics on top of the same physical network. A slice is a set of either virtual (e.g., VMs) or physical resources that an infrastructure provider can assign to a tenant based on the description of what the tenant requires [9]. Business roles have been proposed for slicing in the 5G ecosystem. CDN providers can rely on slicing to build a virtual core network with in-network storage on top of a physical infrastructure that supports traditional IP routers but does not support routers with in-network storage. The key benefit is that content delivery can be performed over a network with traditional IP routers or over a network with routers with in-network capability, both on top of the same infrastructure in a flexible and cost-efficient manner, and simultaneously. Network slicing thus ensures extra profit for CDN operators and infrastructure providers by facilitating multi-tenancy and by improving network coverage to meet adaptability and flexibility requirements. Network Function Virtualization (NFV) [10] is an emerging paradigm that makes slicing possible. NFV uses virtualization technology to decouple a physical network's resources from the functions running on top of it to provide agility, flexibility, and dynamicity.

Users' interest in improving their video streaming experience has increased considerably in the last few years. As a result, immersive Virtual Reality (VR) and 360° video technologies are advancing tremendously. Devices such as Head-Mounted Displays (HMDs), mobile devices, PCs,

etc., are equipped with sensors, processing, and display features that enable them to receive, process, and stream immersive 360° videos. However, handling these steps in streaming 360° videos is quite challenging, as it involves large amounts of data (requires higher bandwidth) and is a very low latency-sensitive application. Leveraging computing in the network can help to improve network performance and overcome the above-mentioned challenges. Software Designed Networks (SDNs) [11] enables a technology that separates the control and data planes and allows networks to be programmed using open interfaces. Integrating in-network computing with SDNs can enable programmability and flexibility to improve future CDNs.

1.2. Thesis Contributions

Future network content delivery can be improved by enhancing infrastructures and addressing the related challenges.

1.2.1 An Algorithm for Improving the Accuracy and Stability of Rate-based Adaptive Streaming Over Networks with In-network Storage [1].

A network with in-network storage capability allows contents to be conveyed closer to end-users. This capability has been broadly cited for its success at improving content delivery networks. The ICN architecture offers in-network storage and has several characteristics that can help to improve CDNs. However, ICN based CDNs still face many challenges. For instance, the rate-based DASH framework currently used in CDNs does not function well in ICN settings. In IP networks, the assumption is that all segments are retrieved from the same server, i.e., the surrogate server. Unfortunately, this assumption does not hold in ICNs, because segments might be retrieved from intermediate routers due to the presence of in-network storage in ICN routers. Furthermore, the ICN in-network storage feature can introduce oscillations in rate estimation. In our work, we tackle the algorithmic challenges for rate-based dynamic adaptive streaming in networks with in-network storage. We target the accuracy of the estimated rate and the stability of the streaming in

these networks. The goal is to maximize the QoE by ensuring a more accurate rate for the next segment and thus stable streaming. In addition, an overall system view that includes the MEC server is proposed.

1.2.2 An Algorithm for the Implementation of In-network Storage in Traditional IP Settings [2], [3].

The deployment of in-network storage is very costly for network providers. Furthermore, it requires substituting the traditional IP routers with routers that have storage. Network Slicing enables the co-existence of virtual networks with different characteristics over the same physical network. A slice is a set of either virtual (e.g., VMs) or physical resources that an infrastructure provider can assign to a tenant based on the description of what the tenant requires. Slicing physical infrastructure to build a virtual network with in-network storage (i.e., an ICN core network) is challenging and worth investigating. Embedding a core network slice with in-network storage on top of traditional infrastructure while ensuring the profit of the infrastructure provider can be helpful. In addition, the QoS constraints for different levels of caching are significant.

1.2.3 An Architecture for Provisioning In-Network Computing enabled Slices for 360° Video Streaming [4].

Many applications require performance improvements that cannot be provided with the current internet architecture, i.e., 360° video streaming and immersive services. Deploying computation functionality can help the overall performance enhancement to meet these applications' future needs, such as low latency and high traffic load. Furthermore, INC provides a promising solution to flexibly managing dynamic changes. INC can reduce latency and traffic load by monitoring network resources (i.e., telemetry), load balancing, congestion control, and reliability. Therefore, architectures for provisioning INC-enabled slices need to be rethought and tested for the application of 360° video streaming.

1.3. Background Information

This sub-section presents the background information that is relevant to the domain of this research. The background information covers content delivery networks, in-network storage and in-network computing. Network slicing is also covered because it is a technology used in the solutions proposed in the thesis for in-network storage and in-network computing implementation.

1.3.1 Content Delivery Networks

This sub-section covers the content delivery networks today, dynamic adaptive streaming over HTTP as a well used method of video streaming, and the descriptions related to 360° video streaming.

A) Content Delivery Networks Today

A Content Delivery Network (CDN) [12] consists of origin servers, surrogate or replica servers, and controllers. The content is stored on the origin server, and copies of that content are distributed over different surrogate/replica servers. The surrogate servers offload the content providers' origin servers by hosting the replicated content and delivering transparent and efficient content to the end-users. The role of the controller is to redirect users' requests to the most appropriate surrogate server based on content availability, network conditions, cost, etc. CDNs are designed to deliver the requested content at a reasonable cost and the required quality of experience by making the services faster and more reliable. They have emerged to improve network performance and content distribution by offering mechanisms and infrastructures to deliver content to many end-users. The analysis of current CDNs reveals that, at the minimum, a CDN focuses on the following business goals: scalability, security, reliability, responsiveness, and performance [13].

Each CDN architecture has three main components; content provider, CDN provider, and end-users. A content provider delegates the URI namespace of the web objects. The origin server of

the content provider holds those objects. A CDN provider is a proprietary organization or company that provides infrastructure facilities to content providers to deliver content in a timely and reliable manner. End-users or clients are entities who access content from the content provider's website [14]. CDN provider uses surrogate servers at different geographical locations to distribute content. Client requests are redirected to the nearby surrogate server, and a selected one delivers the requested content to the end-users. Figure 1.1. illustrates CDN architectural components.

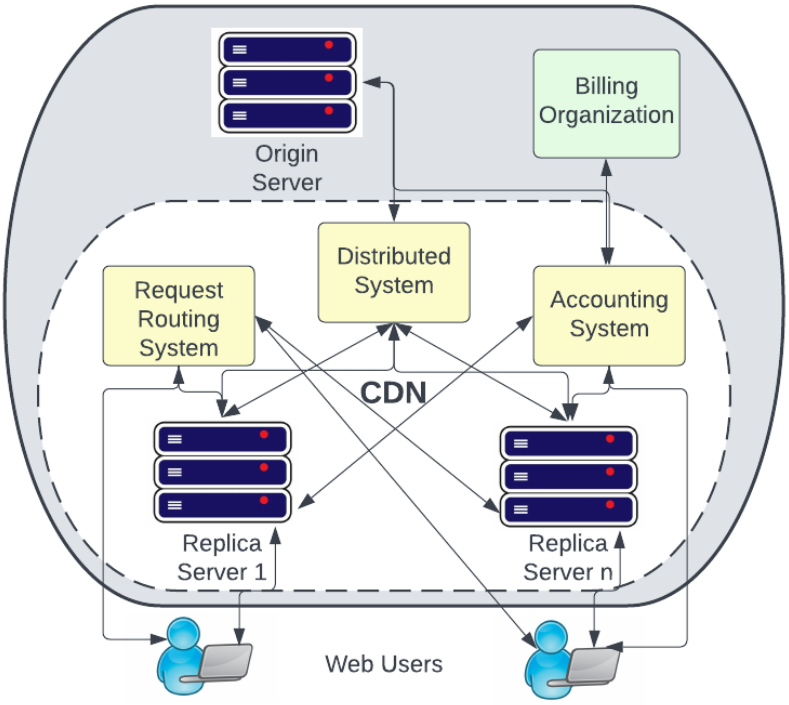


Figure 1.1: CDN architectural components.

B) Dynamic Adaptive Streaming

In the early 2000s, Hypertext Transfer Protocol (HTTP) over Transmission Control Protocol (TCP) became the primary protocol for content delivery over the Internet [15]. Using the application layer buffers to pay for the rate fluctuations of TCP and leveraging HTTP on top of TCP proved beneficial for video delivery. The preliminary implementation of video delivery over HTTP

is called progressive download. The client downloads the whole video file as the TCP connection supports. The drawback to progressive downloading is that clients with different network connections receive the same video quality. This issue led to the appearance of Dynamic Adaptive Streaming over HTTP (DASH) in the mid-2000s. The critical differences between DASH and previous protocols for streaming are: DASH is on top of TCP, while earlier protocols were UDP-based. DASH is a client-driven approach, and the client drives the algorithm in a pull-based manner. DASH deals with content in chunks instead of continuous video packets [16].

In the principles of DASH, video files are encoded into different versions with different qualities or representation bitrates. Each video is fragmented into segments of given lengths between 2 to 10s, and the client can smoothly shift between bitrates if necessary. All the video files' information is described in an XML file called Media Presentation Description (MPD), including video metadata, codec, server IP address, download URL for each segment, etc. [15]. A client starts requesting the MPD and video segments by HTTP GET, and the requests are served using HTTP servers. In DASH, the main goal is to prevent unwanted stalls and increase QoE for the users. A DASH client can adapt to the different available bandwidths of the network by requesting different bitrates. HTTP client sends an HTTP request for the first video segment using the MPD, then based on the ABR algorithm, it selects a proper bitrate for the next segment. The client can dynamically adapt to the fluctuating bandwidth to provide a better QoE. DASH strategies classifies into two main categories : Rate-based (RB) or Buffer-based (BB) [17]. The main idea of RB adaptation is to use the throughput of the last downloaded segment and estimate the throughput of the next segment to ideally request the highest affordable quality. There are some drawbacks to the pure RB algorithms [18]. These drawbacks include the need for rebuffing, bandwidth under-utilization, overestimation, instability, and unfairness. Instead, the main idea of BB adaptation is to request the segment with a proper quality based on the current buffer occupancy. The buffer is divided into different ranges and different actions occur based on the buffer levels. For example, when the buffer level is low (the buffer is empty or below a certain threshold), the algorithm requests the lowest quality to avoid re-buffering. The highest quality can be requested down the road when the buffer level is above a

certain threshold. The accumulated segments in the buffer have the effect of a cushion to absorb small bandwidth fluctuations. On the other hand, unnecessary quality switches can happen if the number of available qualities is high compared to buffer ranges [19]. Since ”DASH” is a standard abbreviation, we will use it for both IP and ICN networks in this thesis. Figure 1.2 shows the DASH principle.

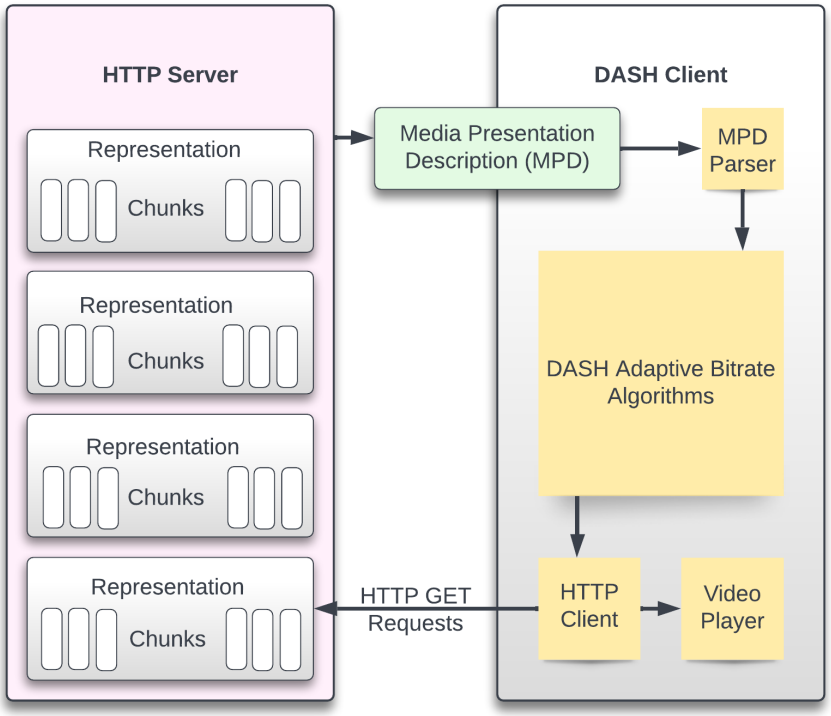


Figure 1.2: DASH principle.

C) 360° Video Streaming

Due to the recent improvements in Virtual Reality (VR) technology, 360° video streaming is increasingly remarked and used. The omnidirectional 360° videos deliver panoramic views and allow users to control their observing direction throughout the video streaming [20]. 360° video streaming has different actors. It involves a user who wears a Head-Mounted Display (HMD) device to interact with the streaming video. Another actor is the content providers who produce/capture, pre-process and stream the 360° videos. Finally, the network provider and infrastructure provider

transmit and distribute the videos to the users.

Adopting real scenarios of 360° video streaming to current networks is highly challenging. Some of these challenges are the higher bandwidth, lower latency requirements, and dealing with vast amounts of data. Mobility adds extra challenges related to radio resources and bandwidth limitations. The excessive amount of data affects the device's energy consumption, and it is not feasible for all devices to playback the video. Handling the streaming is highly time-sensitive compared with traditional streaming.

1.3.2 In-Network Storage

This subsection describes Information-Centric Networks (ICNs) as a well-known architecture with in-network storage capability. The general principles of ICNs are discussed, followed by their routing and in-network storage characteristics. Finally, ICN architecture and video streaming in networks with in-network storage such as ICN are presented.

A) General Principles of Information-Centric Networks

The Information-Centric Network (ICN) [21] has emerged as a future Internet architecture. ICN introduced a new networking scheme around a decade ago, focusing on content delivery instead of host-centric. In ICN, the required data is more significant than who is communicating or where the data is located. All the network processes, such as transport, forwarding, caching, etc., deal with named data instead of the locations of the data. So it can be said that ICN addresses information by location-independent identifiers. ICN transport has been enhanced so that instead of using the sender-receiver TCP/IP scheme, it uses pull-based transport with a connectionless and multipath network. ICN has in-network storage capabilities to reuse data with dynamic forwarding.

There are principal elements to any architecture design in ICN. One of the elements of ICN architecture is the structure of naming data objects which is as crucial as naming hosts for the current Internet. Each data object should have a unique name independent of its location. Some naming scheme has been primarily presented, such as hierarchical (very similar to the structure of

URLs) and flat namespace [21]. Routing is about constructing a path and satisfying the request by routing the object back to the client who requested it. Matching the name information to a source to supply the information is defined as name resolution. The routing and name resolution components are either integrated (coupled) or independent (decoupled). In the coupled approach, data will be routed to the information provider, who sends back the requested content in the reverse path. While in the decoupled one, the name resolution is not restricted to the path that data comes from.

B) ICN Routing

The vast number of works in the ICN research keeps the routing simple and focused on designing other parts such as caching techniques. One of the simple routing algorithms is the shortest path algorithm used in ICN proposals and for testing very widely. We summarize some of the routing algorithms for ICN from the literature [22]. One of the routing protocols is the Breadcrumbs algorithm. In the Breadcrumbs, there is a table of tracked routing information in each cache. Each entry is named a breadcrumb which tracks the time of download, the id of the cache that provides the downloaded content, and the receiver cache where the content is forwarded to it. Requests are forwarded based on the breadcrumb entries at each cache. In the Hash-Routing scheme, there is a need for both edge and cache routers to implement a hash function. When an Interest arrives at the edge router, the router calculates a hash function to find where the requested content is available. In Characteristic Time Routing (CRT), each user has a time-to-live (TTL) based table and forward Interests based on the entries of that table. TTL information is populated based on the Characteristic Time for a content which is the amount of time that content remains in that specific cache. Another algorithm is Scooped Flooding, which is beneficial for congestion control and a case of bottleneck. When different copies of the same content are available in the neighborhood, the congestion-aware local search finds a less congested link and better bandwidth to download that content. In Cache Aware Routing (CAR), the algorithm calculates the transportation cost metric for each route based on popularity and caching strategy. The requests are optimally satisfied such that the cost and overall traffic of the network are minimized. Finally, in Potential Based Routing (PBR), when a node

receives an Interest, it compares available paths to the content and finds the one with maximized facilitated routing to forward the Interest packets.

C) ICN In-Network Storage

All nodes in ICN potentially have storage capability, and requests can be satisfied from the nearest cache of any node that holds a copy of it. Different approaches are proposed for ICN cache management, including homogenous, heterogeneous, off-path, on-path, cooperative, and non-cooperative caching [23]. All the routers on the path that content passes will store the content in homogenous caching. Meanwhile, the routers on a path do not cache the same content in a heterogeneous strategy. In an off-path approach, the Name Resolution System (NRS) is updated about the new content arrival to the cache. In on-path caching, a request may be fulfilled locally without informing the NRS. In cooperative caching, the routers share information about the content after storing it with the whole network. In contrast, in non-cooperative caching, each cache decides to cache individually without sharing its information with the network. We describe this more in the following.

The various cashing strategies are designed based on the best possible location of the cache or the importance of the content to be stored [22]. The first set of strategies works based on location-based caching: caching contents on specific nodes based on specific functions. For instance, the hash-routing scheme designed for off-path caching is one of the location-based caching strategies. The main objective of hash-routing is to reduce cache redundancy by using the hash function, which indicates where to store the content. Another off-path caching strategy is Cooperative in-network Caching (CIC). In CIC, content is divided into different segments and will be cached in different routers. To satisfy a request, there is no need to send the request to the server, and it can be satisfied locally and cooperatively by the routers that have the related segment of the request. The CIC is an off-path strategy because the segments are cached at the routers, which are not available on the publisher-subscriber path. Probabilistic Caching (Probe Cache) is another strategy that is considered an in-network caching resource management approach. The Probe Cache aims

at maximizing the number of different contents stored in a communication route. Also, it attempts to reduce redundancy between caches to increase the probability of finding the content. Finally, the Breadcrumbs is a straightforward, transparent, best-effort search content caching strategy. Each breadcrumb would be a 5-tuple entry with a global ID, which holds information such as when content was requested or forwarded, the node's ID, content type, and finally if the content was pushed down or received. After the content is downloaded, every router creates a trail that maintains the routing history. Next time, the request for the same content will be routed toward the router's directory.

C) ICN Architectures

There have been various ICN-oriented projects from 1999 to now [23]. Named Data Networking [24] is one of the ICN projects funded by the U. S. National Science Foundation. NDN has its root and similarities to one of the earlier projects, so-called Content-Centric Networking (CCN) [25]. The main components of NDN architecture are as follows. (i) Interest packet, (ii) Data packet, (iii) Pending Information Table (PIT), and (iv) Forwarding Information Base (FIB). A consumer sends an Interest packet for the desired content by its name. Once the Interest packet reaches the node which has the data, it will be satisfied and will traverse the reverse-path in the form of a Data packet. Each NDN router uses a forwarding strategy module to determine if, when, and where each Interest packet must be forwarded. All the forwarded Interests that are not satisfied yet are stored in PIT based on their Interest names and incoming and outgoing interface(s). When an NDN router receives an Interest, it first checks the Contest Store. The router satisfies the Interest if the content exists by sending the data packet. Otherwise, the name of the Interest will be searched in the PIT. If the Interest is available, the interface from which it comes will be recorded to enable symmetric routing. Suppose the matching data entry is absence in the PIT. In that case, the request will be forwarded to the data producer based on the available information in the FIB and also the router's forwarding strategy.

D) Video Streaming in ICN

In traditional CDNs, all contents are stored in the origin server. With an increasing number of users, a network will face many requests. Traditional CDNs (over IP) present some weaknesses in their performance, reliability, and scalability [26]. These weaknesses are rooted in the fact that CDNs rely on traditional Internet technology. ICN will enable CDNs to overcome their weaknesses thanks to its multiple inherited characteristics [27]. The simple messaging mechanism of ICNs allows the storage medium to be highly optimized. Moreover, ICN routers can reduce the complexity of IP tunneling and minimize the amount of translation, significantly reducing traffic. These features will facilitate the performance of content delivery in ICNs.

Support for multicast is built into the ICN, and content delivery over an ICN can easily benefit from multicast distribution across its links for improved performance. In addition, an ICN's secure implementation is more reliable for delivering content and prevents many of the inherent challenges in the end-to-end traditional connection approach. Finally, every ICN router acts as a content cache, so the requests for content do not need to traverse to the origin server and can instead be served directly by the ICN routers. This would help meet the ongoing scalability challenge since replicas of the same contents are stored in multiple ICN routers. Therefore, in-network storage (an essential feature of the ICN) can play an essential role in improving content delivery. The deployment of ICN-based CDNs is attractive for greenfield CDN providers but very costly for Brownfield providers since it requires the deployment of non-traditional routers. However, as shown in the literature [28], Brownfield operators may incrementally obtain most of the gains expected from the ICN without radical changes to the current traditional Internet infrastructure.

1.3.3 In-Network Computing

In this subsection, after explaining in-network computing overview, we cover Software Defined Networks principles and P4 programming as enablers for INC.

A) In-Network Computing Overview

INC comes with the vision of enabling computation inside the network as an alternative to restricting computation to the servers outside the network [29]. There are two main categories of INC, in-fabric and end-device developments. We first briefly discuss the benefits of INC, followed by in-fabric and end-device approaches to INC realization. Last, we investigate the applications of INC.

In-network computing provides potential performance benefits such as reducing latency, increasing throughput, and reducing network load for applications with high-performance requirements [30]. To achieve the desired performance and satisfy various requirements, INC consists of two general categories: in-fabric and end-device developments. In the in-fabric category, computation occurs in network devices, such as programmable switches. This approach offloads a set of computing operations from current computing nodes, i.e., servers, to the network elements such as switches, routers, and Network Interface Cards (NICs) [31]. This approach allows the network to benefit from in-network computing by first capturing all traffic and having a vision of the whole network through a switch and NIC and second by not adding additional latency to the path that packets traverse. However, the limitations of computation and memory resources may concern this approach to support conventional routing and switching. The main benefit of the second category of INC, end-device, is that integration is possible without rearchitecting the entire data center (i.e., incremental availability. The end-device development method also enables lower latency and higher throughput than conventional approaches. However, this method does not provide the same latency benefits as in-fabric deployment.

In-network computing is widely used in applications such as packet aggregation, machine-learning acceleration, network telemetry, and stream processing. However, network devices may face limitations on computing and storage resources. Also, the application's correctness should not be affected by the computation in the network. Given the trade off that the integration of In-network computing introduces, it is crucial to find the appropriate type of computation integrated into the network. There are three criteria to identify the suitable forms of in-network computing [31], i.e.,

for different applications (1) significant decrease in network traffic, (2) minimal change required at the application level, and (3) the correctness of the computation.

B) Software Defined Networks

Software-Defined Network (SDN) [11] enables the separation of the control plane (that handles the network traffic) and data planes (that forwards the traffic based on the control plane's decisions). SDN allows networks to be programmed using open interfaces. SDN can address the limitations of the current network infrastructure. This separation is beneficial to flexibility and simplifying network management by balancing the network problem into a traceable piece. An SDN architecture enables delivering a centralized and programmable network. It consists of three main elements. A core element of SDN architecture is a controller. A controller enables management, control, automation, and policy enforcement over network (either physical or virtual) environments. Another element is Southbound APIs which transmit information between the controller and network devices (i.e., switches, routers, firewalls, and access points). The last element is Northbound APIs which transmit information between the controller and the applications. The data plane programmability in SDN is feasible with P4 programmable switches. P4 enables simplicity and speed (focusing just on the requirements), scalability (maximizing data-plane resources to the fundamental level), and data plane telemetry (monitoring real-time Inbound Network Telemetry (INT)). Figure 1.3 illustrates the SDN architecture.

C) P4 Programming

P4 stands for Programming Protocol-independent Packet Processors, and it is the standard for defining the programmable forwarding behavior and configuring switches [32]. Using P4 provides reconfigurability which means programmers can change packet processing in the already deployed switches. P4 is protocol-independent, so switches are not tied to particular network protocols. Programmers can perform packet processing functionality independent of the underlying hardware

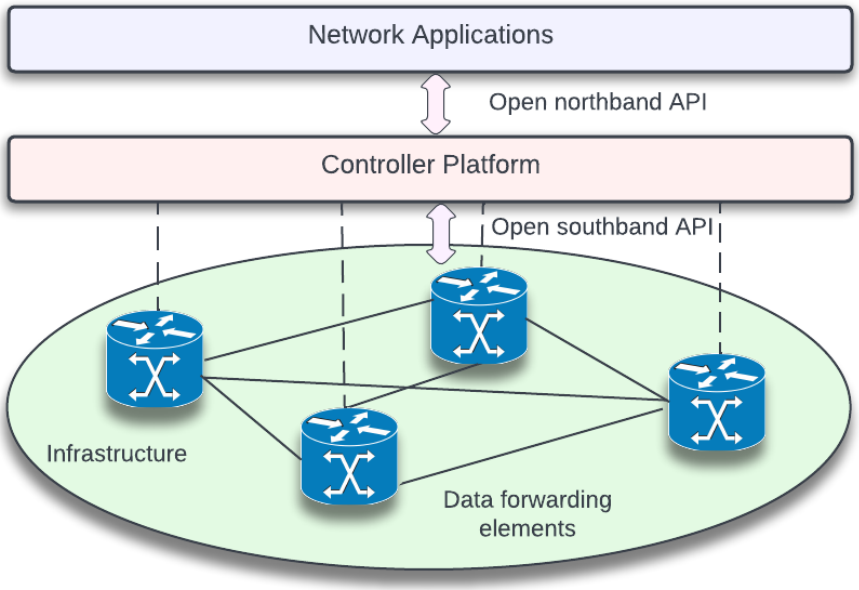


Figure 1.3: SDN architecture.

because P4 is the target independent. Moreover, P4 enabled programmable has some leading characteristics: agility, top-down design, visibility, less complexity, enhanced performance, etc. Figure 1.4 illustrates configuring a programmable switch by classic Openflow versus P4 program.

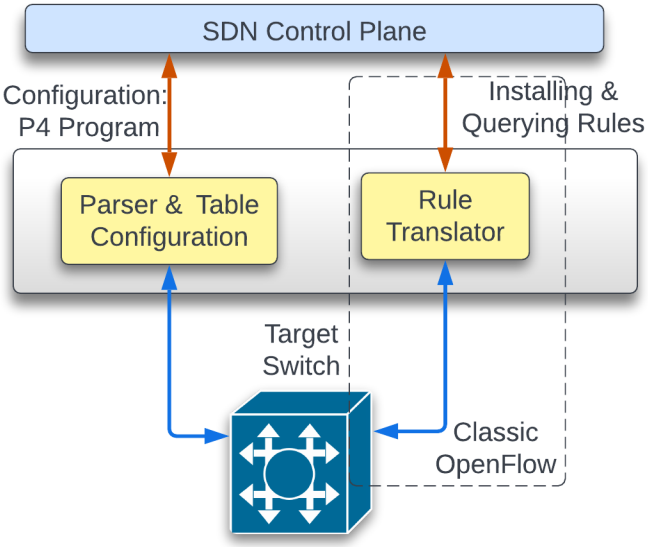


Figure 1.4: Configuring switch using P4 program.

1.3.4 Network Slicing

Network slicing enables different virtual networks' co-exist for different verticals (e.g., e-industry and e-health) over the same physical infrastructure [33]. A slice is a set of either virtual (e.g., VMs) or physical resources that an infrastructure provider can assign to a tenant based on the description of what the tenant requires. Business roles have been proposed for slicing in the 5G ecosystem. A simplified version is described next. The Infrastructure Provider owns and offers the physical network infrastructure. The slice providers are located on top of this infrastructure. In the case of CDNs, slice providers offer end-to-end slices to meet requested requirements. This layer dynamically configures and embeds the slices and does so optimally. CDN providers define the set of requirements to be mapped to a slice to ensure the QoS for their users [34]. Network slicing in 5G ensures extra profit for operators and infrastructure providers by facilitating multi-tenancy and improving network coverage to meet adaptability and flexibility requirements. Slicing requires operational capabilities to dynamically instantiate, modify and delete slices based on the tenants' requirements. The infrastructure provider should be able to intelligently assign and reassign resources to different requests to ensure that resources are used efficiently while ensuring that the QoS violations of service reassignment and degradation are kept to an acceptable level for the slice owners. Virtualization is one of the enablers of network slicing [35]. Each slice is dedicated to a specific service and has different requirements. The main challenge of network slicing is thus how to allocate the resources to meet tenant requirements while unlocking the opportunity to make more profit for the providers who own the resources.

1.4. Thesis outline

The rest of the thesis is organized as follows. Chapter 2 discusses the requirements and provides a critical review of the state-of-the-art. We organize the thesis into algorithmic contributions and architectural contributions. Accordingly, for the algorithmic contribution, Chapter 3 presents the accuracy and stability algorithm in rate adaptation streaming for networks with in-network storage.

Chapter 4 presents the proposed algorithm for implementing in-network storage in IP settings. For the architectural contribution, in Chapter 5, we present an architecture for provisioning in-network computing enabled slices. Finally, we conclude this manuscript in Chapter 6 and provide future directions for this research work.

Chapter 2

Related works

This chapter presents a set of requirements derived from our problems. Next, we investigate the state-of-the-art in light of these requirements.

2.1. Requirements

We categorize the requirements into two main categories: requirements specific to in-network storage integration and requirements specific to in-network computing integration to the infrastructure. For the first category, we derive the requirements related to rate estimation in dynamic adaptive streaming in networks with in-network storage and the requirements specific to implementing in-network storage. For the second category, we derive the requirements specific to provisioning the in-network computing enabled slices. It should be noted that some of the proposed requirements have only architectural dimensions, while others have both architectural and algorithmic dimensions. As for the algorithmic dimension, we consider that a given system will meet a given requirement if the algorithm has the main objective of meeting that requirement or it is part of the set of constraints the algorithm should satisfy. We discuss these aspects further as we present each requirement.

2.1.1 Requirements Specific to Networks with In-network Storage

A) Requirements Specific to Rate-based Dynamic Adaptive Streaming in Networks with In-network Storage.

The following requirements are considered to be essential for rate estimation algorithms. However, these requirements only have algorithmic dimensions.

1) Accuracy: The system should enable the estimation of a more accurate RTT compared to the typical RTT computed solely by the client. When estimated as in IP networks (the assumption is that the next segment will be retrieved from the same server), estimation of the adaptation rate in ICNs is not accurate. This inaccuracy negatively affects the DASH performance because the next segment in the ICN can be retrieved from any intermediate storage with a copy.

2) Stability: The in-network feature can introduce a significant bitrate fluctuation in the DASH streaming process. This bitrate fluctuation causes oscillation and decreases the QoE perceived by the users. To benefit from in-network storage, the system should ensure rate stability to prevent oscillations.

B) Requirements Specific to the implementation of In-network Storage in Traditional Networks.

1) Dynamicity: The system should allow infrastructure providers to dynamically instantiate, modify and delete slices with in-network storage capability based on the requirements of the application provider.

2) QoS: The proposed model should consider the peculiarities related to the CDN applications and enforce different QoSs for each level of caching, based on their distance.

3) In-network storage enabled slices: The proposed model should consider the peculiarities related to in-network storage enabled slices (i.e., ICN slices). For example, ICN network nodes have specific computing and storage capabilities, and so infrastructure providers

should be able to harness a storage node in addition to using a computing node.

4) **Profit or revenue:** The model should enhance the profit or revenue of network slicing for the infrastructure provider while satisfying various service requirements. The infrastructure provider should be able to harness in-network storage nodes to respect the QoS of the users while making a profit from providing this service.

2.1.2 Requirements Specific to Networks with In-network Computing

A) Requirements specific to the provisioning of INC-enabled slices for 360° video streaming

1) **Latency:** Provisioning INC-enabled slices for 360° video architecture should be able to reduce the delay of video streaming. This will make it possible to meet the stringent requirements of immersive services and 360° video streaming.

2) **Network load:** Reducing the network traffic is critical in CDNs, especially in immersive services and 360 videos, both bandwidth-consuming applications. This thesis focuses on the traffic reduction of video streaming applications. The architecture should be able to reduce network traffic by integrating the INC and P4.

2.2. Related works

This section presents the state-of-the-art for content delivery in future networks. The first category discusses networks with in-network storage, architectures and algorithms for solving the accuracy and stability issues of rate estimation for CDNs with in-network storage. The architectures and algorithms for implementing in-network storage in traditional networks are also reviewed. In the second category, the architecture and algorithms for implementing in-network computing in traditional networks are reviewed.

2.2.1 Networks with In-network Storage

This sub-section discusses the research on solving accuracy and stability issues of rate estimation for CDNs with in-network storage. Then in the second category, which is in-network storage integration, the general virtual network embedding problems that share several similarities with our problem are reviewed, followed by works that show the ICN as a virtual network over a substrate network. We also review studies on the dynamic slicing of the substrate network, followed by a discussion of the studies on modeling the profit and other economic aspects of network slicing and proposed machine learning solutions for network slicing.

A) Architectures and Algorithms for Solving Accuracy and Stability Issues of Rate Estimation for CDNs with In-network Storage

Few works help to solve the issues of accuracy and stability in CDNs over networks with in-network storage such as ICNs. We briefly discuss the general works on network-assisted DASH (i.e., works that do not rely on MEC). Most proposed solutions build on Server and Network Assisted DASH (SAND) [36]. In SAND, assisting information (information from the network used for a more accurate estimation of the adaptation rate) is exchanged between client and network for more accurate RTT estimation. However, no concrete procedure is proposed for this estimation, although the exchange of messages is enabled between client and network. Reference [37] builds on SAND and targets the oscillation issue of DASH over Content-Centric Networks (CCNs), a variant of ICNs. However, it does not propose a concrete procedure for estimating a more accurate RTT. Reference [38] is a follow-up of reference [37]. It targets scalability but does not provide a solution to the adaptation rate challenge. An example of work that does not build on SAND is DASH-INC [39]. It targets DASH's rate stability issue over an ICN and does not address the rate estimation issue. They propose a framework for transcoding and multiple throughputs. Each router stores the highest bitrate in its content store in its model. When it receives a request for a lower rate, the router transcodes the highest bitrate into the lower bitrate on the fly and sends it to the client. To solve the cache oscillation problem, the authors propose increasing the observation windows to smooth out

the rate changes. Another solution they proposed is to set the rate transitions at some predefined intervals to restrict the oscillation. In this way, the rate could not fluctuate between the cache and the server unless the next transition is allowed. This framework met our second requirement, rate stability, but it does not address the adaptation rate issue.

B) Architectures and Algorithms for Implementing In-network Storage in IP Settings

1) General virtual network embedding problem

Reference [40] considers embedding nodes with computing and storage capabilities. However, it assumes that the substrate nodes have both computing and storage capabilities, and it does not meet any of our requirements. The authors in [41] include storage and caching (without having ICN slices) in their effort to maximize revenue for the infrastructure provider by optimizing caching resource allocation. They propose a new dynamic scheme for network slicing in a 5G core network, but their model does not consider the different QoSs for varying levels of caches, nor using ICN slices, and does not meet our second and third requirements.

In [42], the authors provide an efficient resource allocation in 5G slicing. They propose a model to perform joint slicing of 5G and edge computing resources. Their aim is to minimize latency for multiple classes of traffic. They evaluate their proposal based on different parameters such as the type of traffic, tolerable latency, network topology, etc. Their results confirm that their proposal can provide efficient resource allocation, but their model and its parameters are static over fixed time slots and so it does not meet any of our requirements. A queuing-based system model for 5G slicing is presented in [43]. The authors studied the case when a subset of VNFs is shared by multiple services that have different QoS requirements. The proposed heuristic is designed to make joint VNF placement and CPU/storage allocation, with the objective of minimizing latency. While they have different QoS levels for different services, they do not meet our second requirement of considering CDN peculiarities. Their model can allocate storage to the slices, but it does not have ICN slices and so does not meet our third requirement. In addition, they did not model the revenue advantages of slicing and thus do not meet our fourth requirement.

2) ICN as a virtual network over a substrate network

Embedding ICN virtual networks is a challenging problem that several researchers have addressed. In [44], researchers propose an information-centric virtual content network (IVCN) slicing framework. The IVCN slicing use case in their model is a heterogeneous fog-enabled RAN. Their proposed model for performance optimization on the IVCN slicing work is based on Service Function Chaining (SFC) and is designed to optimize the mapping of VNFs and virtual content placement to maximize the weighted hops for content storage. The performance evaluation considered different metrics (e.g., the hit rate, the average weighted hops per request on the data plane and on the control plane, and the average content redundancy). They compared the results with other caching schemes to show how their model outperforms them. Although they model dynamic ICN slicing and used VNFs, they do not consider CDN peculiarities or profit. Further, the work in [45] proposed an ICN network slice embedding solution over 5G networks without the need for any advance knowledge about the topology or resource provisioning. In [45], the authors only considered ICN slices, as opposed to our work, where we consider two types of slices, ICN and IP. Moreover, in [45], even though a VNF is used for the ICN slices, the embedding is not organized dynamically, and they did not consider CDN peculiarities or profit opportunities for the infrastructure providers.

A general 5G-ICN architecture and a detailed network slicing architecture for the embedding and deployment of models is proposed in [46]. While this work does discuss a use case on mobility as a service, it does not cover the resource allocation problem. This work is not dynamic nor does it consider CDN peculiarities. In [47], the authors propose a slicing framework in software-defined information-centric networks with the aim of simplifying the computational overhead for latency-sensitive applications. Ref. [47] considers device-to-device (D2D) networks as a solution to improve the performance and latency for mobile users, while we consider cellular networks in a more general way. This work did not tackle the CDN peculiarities nor the profit aspects of network slicing.

Reference [48] proposes an architecture in which a virtual wireless ICN and a virtual traditional wireless network co-exist on the same physical 5G wireless network. They offer a scheme to

jointly optimize both caching and resource allocation, with the objective of maximizing profit while addressing resource allocation for individual users. This work also assumes a 5G substrate core network that provides both computing and storage. In addition, their substrate network is composed of real resources, while in our case it is made up of an NFV Infrastructure (NFVI). Thus, they did not consider either CDN peculiarities or profitability.

3) Dynamic slicing of the substrate network

To best reflect how a system can scale with human behavior and all its varied demands, it is important to study dynamic slicing. Reference [49] proposes two heuristics for dynamic slicing over a 5G substrate network, with the first heuristic goal being to minimize the wavelength resource usage in the physical network. They defined the degradation values for the virtual networks already mapped in the physical network. Their second heuristic objective is to minimize this degradation value, the number of light path re-configurations and wavelength resource usage. Their work emphasizes the advantages of dynamic slicing, which can reduce the virtual network rejection probability. Although they mentioned that these improvements can help network providers increase their revenue, they offered no further elaboration.

In [50], the authors tackle the problem of dynamic slicing in a 5G network and explore that problem under the heterogeneous cloud Radio Access (H-CRAN) architecture, which is energy- and cost-efficient compared to traditional decentralized cellular architectures. Their scheme considers the tenant's priorities, the baseband resources, fronthaul and backhaul capacities, the QoS and the interfaces. Simulation results show that the proposed scheme provides better throughput, fairness and QoS compared to baseline schemes. Their work is dynamic and considers different QoSs for different services. In [51], the dynamic allocation of resources to 5G network slices is modelled by predicting the slice bandwidth requirements. This work investigates a way to prevent revenue loss to network operators. When it predicts errors, the proposed algorithm reallocates bandwidth resources from lower priority slices to higher priority slices. To ensure the QoS, it also

predicts the end-to-end delay and admits each flow into slices that can satisfy specific delay requirements. However, this work does not consider CDN peculiarities and does not meet our second or fourth requirements.

4) Modeling the profit of slicing over a substrate network

Several profit models have been proposed for the different dynamic slicing business model actors over a 5G substrate network. For instance, in [52] the authors propose an algorithm that allocates and admits requests for network slices while maximizing the infrastructure providers' revenue and ensuring that SLAs are met. Their main contributions are their model, based on the Semi-Markov Decision Process (SMDP), and their algorithm, based on Q-learning, which achieves close to optimal performance. They did not consider CDN or ICN peculiarities.

Reference [53] focuses on the profit opportunities related to the network slicing market as a means to assess the feasibility of the new 5G business model. They propose an admission control mechanism based on bid selection, maximizing the revenue of infrastructure providers and providing a fair slice provision to competing service providers. Their evaluation is based on comparing their results with other admission policies, such as First Come First Serve (FCFS) or Always Admit (AA) in on-demand or periodic slicing. Although their model is dynamic and considers the profitability of slicing, it does not consider CDN or ICN peculiarities. In [54], the authors propose a profit model for network operators to guide the deployment of each slice. Their proposed value chain in the sliced network is designed to maximize profit and model good resource management. While they do discuss some of the challenges, their model is not dynamic, nor does it consider caching in an explicit manner, and so they only meet our fourth requirement.

A model that considers the profits of both the slice provider and the slice customer is presented in [55]. These authors studied network slice dimensions along with resource pricing policies to explore the trade-off between resource efficiency and maximizing profit. Although their model covers profit maximization for different actors, they did not consider any different components for the provider's expenditures. They also did not consider storage resources, and so they only

meet our fourth requirement. Another profit model for slicing a 5G substrate is offered in [56], in which a profit-aware resource allocation model in a virtualized multi-tenant environment is proposed. These researchers studied profit based on the number of users per slice, the coverage radius and the maximum transmit power. They considered a radio access network, but their model is not dynamic. A model for evaluating the profit of slicing a 5G network while setting different prices for each class of slices is proposed in [57]. These researchers formulate the problem using fundamental economic principles and present an algorithm to solve it. Their simulation confirms an improvement in providers' profit. However, they did not consider CDN or ICN peculiarities.

A model for evaluating the profit of Radio Access Network (RAN) slicing for multiple tenants in a 5G network is proposed in [58]. Their algorithm includes the decision making process in order to gain additional income by granting additional capacity beyond the SLA level, and assigns a penalty for potential SLA breaches. These researchers did not consider CDN or ICN peculiarities. The last work we evaluate here is another effort to jointly optimize resources and revenue, offering an auction model applied to the pricing of virtual networks to solve the problem [59]. Even though the general problem of jointly optimizing resources and revenues is briefly discussed, this work does not consider the peculiarities of embedding ICN slices or of having different QoS levels, and so they only meet our first and fourth requirements.

5) Machine Learning approaches for network slicing

The authors of [60] study network slicing integrated with caching resources, which, to some extent, is the same as having ICN core network slices with cache-enabled routers. More specifically, a novel machine learning-based management framework for dynamic slicing and content placement is proposed in [60]. The problem is formulated as a Markov decision process. The focus is on cache resource provisioning, with a customized content placement algorithm to find a specific placement for each service provider. While Ref. [60] aims to realize a proper balance between satisfying QoS requirements and obtaining optimal resource utilization of the infrastructure providers, it does not investigate the economic aspect of slicing (e.g., revenue, expenditure, and profit). Furthermore,

the method of solving the problem in Ref. [60] is fundamentally different from our work, mainly because their proposed solution relies on machine learning, whereas our work proposes a heuristic to achieve the near-optimal solution of the developed ILP.

In another work, [61], the authors focus on a machine learning solution to solve the resource allocation problem by designing a network slice admission control algorithm. The algorithm learns the best acceptance policy for maximizing the infrastructure revenue. They do not discuss slicing caching resources or ICN peculiarities. They consider satisfying QoS requirements as an SLA for admitted slices, with the QoS remaining fixed, while in our work we impose different QoSs for different levels of caching. In another recent work [62], researchers study the network slicing problem of two 5G network services, Ultra-Reliable Low Latency Communications (URLLC) and enhanced Mobile Broad Band (eMBB). The authors of [62] formulate the network slicing problem as an optimization problem that aims at maximizing the data rate while satisfying the reliability constraints of URLLC. A DRL-based framework is then proposed to solve the problem, and simulation results illustrate that the proposed method has high reliability while satisfying URLLC constraints.

2.2.2 Networks with In-network Computing

Very few works have focused on using in-network computing for infrastructure provisioning. To the best of our knowledge, this thesis is the first attempt to harness INC for 360° video streaming service provisioning.

A) Architectures and Algorithms for In-network Computing Integration.

Most of the papers reviewed in this section use a P4-enabled SDN-based network as an infrastructure. The SDN infrastructure enables programmability and dynamic network management with the critical goal of reducing delay. For example, in [63], the authors presented ZipLine, an approach to design and implement compression at line speed using a P4-enabled network. They evaluated their system by measuring throughput, latency, and compression measurements in real-world scenarios. In [64], the authors propose an architecture that benefits from INC to achieve

a better performance in terms of high-throughput and low-latency in industrial IoT systems. The architecture is an Information-Centric Network (ICN) powered mobile edge architecture. The critical tasks are offloaded to the ICN devices, while the rest go to the Multi-access Edge Computing (MEC). The ICN, as a networking scheme, decouples information from its sources and provides in-network storage in all the routers. They evaluated the feasibility of their architecture using two industrial use cases, in-network robotic motion and in-network fire detection. They did not discuss traffic reduction in the network.

The authors of [65] focused on tile-based panoramic live video streaming leveraging in-network resources. They studied the benefit of moving transcoders near the end-users. This work measured cache hits, tile size, and traffic. However, it did not consider the specific challenges related to 360° videos, such as latency.

In [66], the authors used In-band Network Telemetry (INT) and presented an SDN-based network where network statistics were gathered and used for network control and slice management. The presented system can improve the QoS by reducing the delay. The INT layer in this model is implemented using the click modular router. It can act as a source, intermediate, or sink node. The communications between the source and sink nodes and the Application Handler are via an API that gathers statistics about the flow. To satisfy the required level of QoS, dynamic slicing can reconfigure the network slices based on live statistics about the INT layer. Although this work presents the framework as an addition to the SDN architecture, they do not propose a complete architecture and do not meet our requirement about reducing the network load. Authors in [67] proposed a new architecture over 5G networks to stream video using SDN-assisted IP multicasting. The service model selectively duplicates and forwards a video upstream at SDN switches instead of a central selective forwarding unit. This is managed by bandwidth and delay-aware multicast trees maintained by the SDN controller. Their evaluation results show that the proposed model reduces delay and overall network resource utilization. At the same time, bottlenecks are avoided between uploading peers. The SDN controller has complete control over the network traffic and the switch queues. A network provider with SDN infrastructure can implement centralized path

computation over network slices that fulfill stated bandwidth requirements with minimum latency. The presented architecture is only proposed and evaluated based on scalable video streaming and did not consider network load requirements. The authors of [68] also used SDN controller knowledge of the network to reduce delays for the use case of 360° video streaming. In this model, the SDN controller keeps updated statistics of the available paths for each node that joins the SDN network. Each connection is given a rank based on the delays. The flow allocation is performed based on the delay ranks to establish a balanced multi-path connection. Their evaluation shows that their proposed model increases the QoE of 360° video streaming users by reducing playback stalls and improving network efficiency and scalability; however, they do not consider network load reduction using INC.

SDN-based networks are used in another paper [69] to improve content delivery and increase the end-user QoE. They explored a multimedia gateway acting as a complementary element of the OpenFlow controller. This gateway uses its statistics from the network and routes multimedia traffic to be delivered via different paths from other flows, prioritizing the flows and allocating the required bandwidth to the different flows. This work does not cover dynamic bandwidth allocation and it does not consider latency. The authors in [70] aim at increasing QoS in video streaming over SDN-based networks. They propose a framework that uses the network's information for load balancing and to avoid bottlenecks when congestion occurs. The framework is based on a new design of Openflow controller, which continuously monitors the loads of each node to detect an overload and perform load balancing. Their results illustrate that the framework improves the QoE of the video streaming to end-users; however, they only considered video streaming cases over a single server and did not consider reducing the latency. The authors of [71] proposed an elastic system architecture with a five-layer abstraction for interactive video applications. Interactive video application is a generic term for videos in cloud gaming and Virtual Reality (VR), as well as 360° videos. These videos have stringent latency requirements, which require a very limited elapsed time between user instruction and workflow modification. A model for efficient task representation is designed to increase system flexibility in this work. The system performance is analyzed using

a set of novel latency measurement metrics. The results show that measured end-to-end latency satisfies the restrictions of most interactive video applications provided by an edge. This work does not discuss the network load or the efficiency of network resource usage at the edge.

2.3. Conclusion

In this chapter, we first derived a set of architectural and algorithmic requirements from our problem and then surveyed the literature accordingly. Table 2.1 provides a summary of the reviewed architectural and algorithmic papers. For each paper, we show the requirements that are met and those that are not met. None of the reviewed papers satisfy all our requirements.

Table 2.1: Related Work Evaluation.
(met ✓, not met ✗)

Requirements / Related works	Algorithmic Requirements						Architectural Requirements	
	Accuracy	**Stability**	**Dynamicity**	**QoS**	**storage-enabled slices**	**Profit**	**Latency**	**Load**
[36]	✓	✗						
[37]	✗	✓						
[38]	✗	✗						
[39]	✗	✓						
[40]			✗	✗	✗	✗		
[41]			✓	✓	✗	✗		
[42]			✗	✗	✗	✗		
[43]			✓	✗	✗	✗		
[44]			✓	✗	✓	✗		
[45]			✓	✗	✓	✗		
[46]			✗	✗	✓	✗		
[47]			✓	✗	✓	✗		
[48]			✓	✗	✓	✗		
[49]			✓	✗	✗	✓		
[50]			✓	✗	✗	✓		
[51]			✓	✗	✓	✗		
[52]			✓	✗	✗	✓		
[53]			✓	✗	✗	✓		
[54]			✗	✗	✗	✓		
[55]			✗	✗	✗	✓		
[56]			✗	✗	✗	✓		
[57]			✓	✗	✗	✓		
[58]			✓	✗	✗	✓		
[59]			✓	✗	✗	✓		
[60]			✓	✗	✓	✗		
[61]			✓	✗	✓	✗		
[62]			✓	✗	✗	✗		
[63]							✓	✗
[64]							✓	✗
[65]							✗	✓
[66]							✓	✗
[67]							✓	✗
[68]							✓	✗
[69]							✗	✓
[70]							✗	✓
[71]							✓	✗

Chapter 3

Algorithm for Rate-based Adaptive Streaming in Networks with In-network Storage

[1]

3.1. Introduction

Information-centric based CDNs face many challenges. For instance, the rate-based DASH framework currently in use in CDNs does not function well in ICN settings. A DASH client computes the RTT of each segment it requests and receives and then uses it to estimate the RTT of the next segment. The quality requested for the next segment is decided accordingly. In IP networks, the assumption is that all segments are retrieved from the same server, i.e., the surrogate server. Unfortunately, this assumption does not hold in ICNs because segments might be retrieved from intermediate routers due to the presence of caches in ICN routers. Furthermore, the ICN in-network caching feature can introduce oscillations in rate estimation. In this chapter, we tackle these issues

[1] This chapter is based on a published paper: Rayani, Marsa, Roch H. Glitho, and Halima Elbiaze. "ETSI multi-access edge computing for dynamic adaptive streaming in information-centric networks." Globecom 2020-2020 IEEE global communications conference, 2020.

by using the ETSI's MEC paradigm. MEC provides cloud computing capabilities at the edge of mobile networks. This chapter first proposes an overall system view that includes the MEC server. Next, we put forward a mathematical formulation and a novel rate adaptation algorithm in the proposed system that includes a MEC server and an ICN. Finally, the evaluation scenario and results show that our suggested rate adaptation algorithm, which relies on MEC capabilities, leads to a better Quality of Experience (QoE) for end-users due to a more accurate and stable RTT estimation. We conclude the chapter in the last section.

3.2. Overall System View

Fig. 3.1 shows a high-level view of the system model. The considered system is organized in five layers: (i) the MEC-enabled DASH clients, (ii) the Radio Access Network, (iii) the DASH MEC application, (iv) the MEC-enabled ICN edge router and (v) the CDN surrogate server.

DASH enabled clients are located at the bottom layer. They interact with the MEC edge server and routers located in layer 3 and 4 respectively. These connections are enabled through the 5G Radio Access Network (RAN). RAN is mainly composed of base stations and is illustrated in layer 2 of Fig. 3.1. Each base station allocates available radio resources in a proportionally fair manner to clients [72]. Beyond the RAN, interactions between DASH clients and both the MEC edge server and the ICN core network are wired. The MEC layer consists of a MEC application that computes the accurate RTTs with the RTT calculator, stores them in the RTT repository, and manages the requests for RTTs using the RTT request handler. The content delivery is done over an ICN core network (the fourth layer). The ICN core network is composed of MEC-enabled ICN edge routers. In the ICN, the edge routers are the first set of routers to receive the requests in the network. Finally, the videos reside on the CDN surrogate servers in the fifth layer, connected to the ICN core network. The functioning of the three main components is described in detail here.

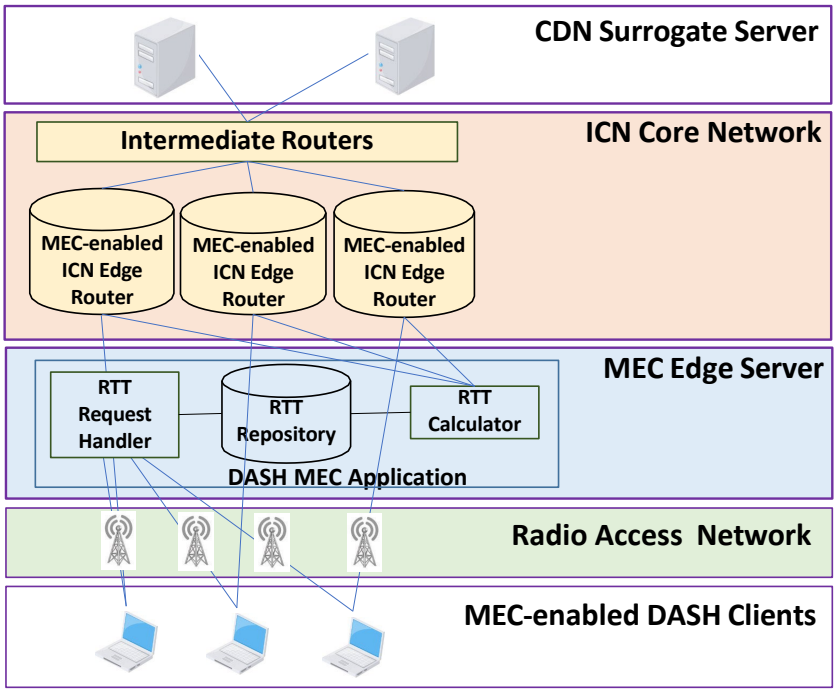


Figure 3.1: High-level view of the system model.

A) MEC-enabled DASH Client

After receiving the MPD, the DASH client starts sending its requests for video segments to the surrogate servers through the MEC-enabled ICN edge router. To determine the proper quality for each segment, the client can communicate with the DASH MEC application. It obtains the RTT from the MEC application. This RTT may be a previously-stored one, or it may have been estimated by the traditional method if none has been stored.

B) DASH MEC Application

The DASH MEC application subscribes to notifications for the time tags whenever a request for a video segment (or the video segment itself) crosses the MEC-enabled ICN edge routers. It uses these time tags to compute the accurate RTTs and store them in the repository. Whenever the DASH MEC application receives a request for the most accurate RTT, the RTT request handler checks the repository to find a stored RTT for that segment from previous rounds. Suppose the

RTT is not available in the repository. In that case, the DASH MEC application replies with the calculated RTT, which is less accurate for the client.

C) MEC-enabled ICN Edge Router

First, the router receives a subscription from the DASH MEC application. The request for the segment then traverses the MEC-enabled ICN edge router. As soon as this happens, the router notifies the DASH MEC application about the time tag of the request's arrival. Finally, the request is satisfied, and the data packet of the segment traverses the edge router again. Upon receiving the data packet, the MEC-enabled ICN edger router notifies the time tag of the segment's arrival at the DASH MEC application.

The envisioned scenario includes a MEC-enabled DASH client who uses DASH MEC application assistance to perform a more stable adaptive streaming. The DASH MEC application assists the client by providing a more accurate RTT, allowing the DASH algorithm to make better-informed decisions and thus be better adapted to network conditions. To illustrate this scenario, we introduce a sequence diagram showing the interactions of the main components as depicted in Fig. 3.2. The client requests and receives the MPD (Fig 3.2: actions 1 and 2). The client informs the DASH MEC application about the video session that is in the process (actions 3 and 4). The DASH MEC application now subscribes to the ICN edge router for the video to receive the notifications required for RTT calculation (actions 5 and 6). The client decides on the quality of the first segments based on the default value in the adaptation algorithm (action 7) and then sends the request and receives the data packet for that segment (action 7). After requesting and receiving the segment (actions 8 and 9), the ICN router notifies the arrival time of the request and the retrieval time of the data packet for that specific segment to the DASH MEC application (action 10). Next, the RTT calculator in the DASH MEC application can calculate the RTT based on the time tags (action 11). Then, the client pulls the RTT to decide on the quality of the next segment (action 12). Upon receiving the request for the RTT, the DASH MEC application checks its repository to fetch more accurate RTTs from the previous runs for the same segment. If no accurate RTT is available, the DASH MEC

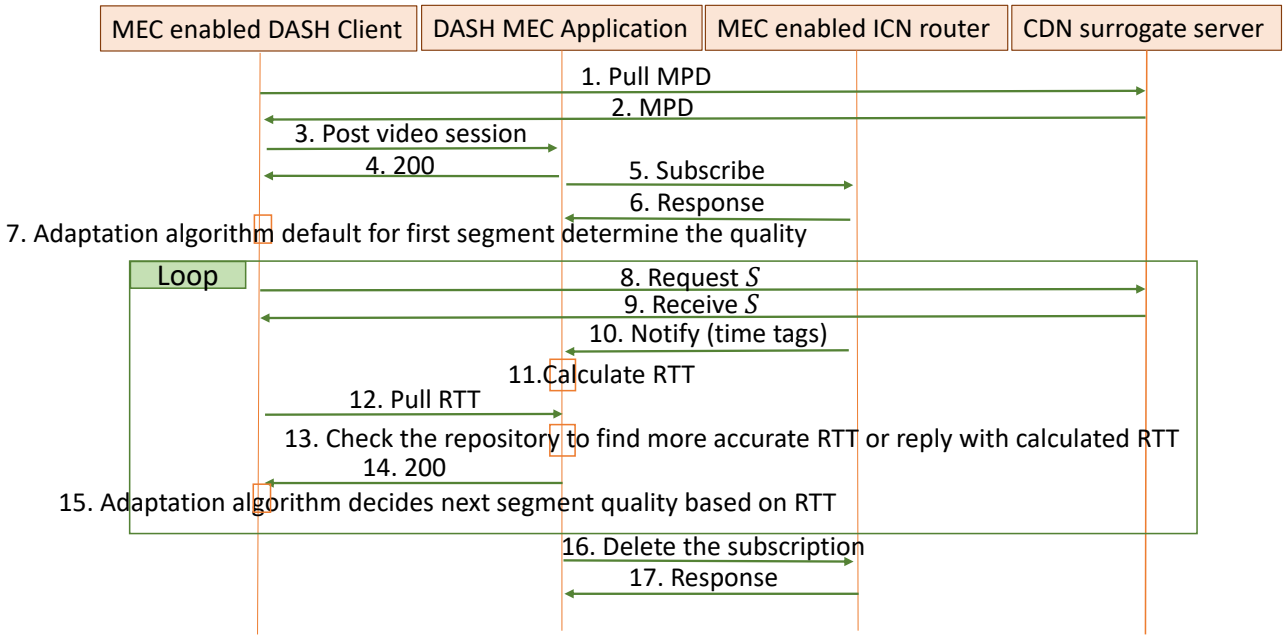


Figure 3.2: Illustrative sequence diagram.

application sends the calculated RTT to the client (actions 13 and 14). The adaptation algorithm can now decide on the next segment's quality based on the received RTT (action 15). This loop continues until the whole video is downloaded segment by segment. After completing the video, the subscription between the DASH MEC application and the ICN edge router is ended (actions 16 and 17).

3.3. Problem Formulation

Our DASH video streaming system operates over time slot t. We assume that we have a video v consisting of $s \in S_v(t)$ segments at time t. Each segment is available at different quality with equal playback duration. We denote by $Q_s(t)$ the set of qualities available for the given video segments at time t. We write $q_s^{max}(t) \in Q_s(t)$, $q_s^{min}(t) \in Q_s(t)$ for the representations with the highest and lowest qualities in $Q_s(t)$. We define $q_s^{\uparrow}(t)$ and $q_s^{\downarrow}(t)$ for the representations with the next higher and lower qualities, respectively. Let $R_{qs}^{j}(t)$ refer to the request for segment s with quality q_s by client j at time t. The arrival time of the request $R_{qs}^{j}(t)$ by client j to the ICN edge router is defined as $A(t)$. This edge router is the same one that passes the requested segment to client j due to the

symmetric routing in ICN. Therefore, we consider $D(t)$ as the retrieval time of the segment s by client j.

In ICN, some contents will be retrieved from the intermediate caches; if they are not available the request will traverse all through the network to the servers. We define a binary variable to map the clients to surrogate servers which have the requested video segments such that $x_{j,k}(t) = 1$ if client j is allocated to server k at time t, and $x_{j,k}(t) = 0$ otherwise. We define the base stations by $b \in BS$. The data transmission between the base station b and server k and different clients goes through a shared link. It is to be noted that $W_{b,k}(t)$ refers the spectrum allocated (in number of resource blocks) in the frequency domain, at time slot t. We also assume that clients are stationary throughout this work. $T_{b,j}(t)$ is capacity of the links and it is computed as follows where the SNR is Signal to Noise Ratio:

$$T_{b,j}(t) = BW * log(1 + SNR) \tag{3-1}$$

Table 3.1 lists the notations of our model along with their descriptions.

There are four major factors that can significantly affect the quality of experience perceived by DASH clients [73]: video quality, startup delay, stalling time and bitrate switching. In our model, we aim to consider above factors to improve QoE. The main goal of CDNs is to achieve optimal QoE for end users by maintaining QoE metrics at an acceptable level.

(1) Segment Quality

Segment quality has the highest direct impact on the QoE perceived by the client. The average video quality over $|S_v(t)|$ downloaded segments from the caches or surrogate servers by client j at time t is obtained using the following relation:

$$\bar{q}_j(t) = 1/|S_v(t)| \sum_{s=1}^{|S_v(t)|} \sum_{k=1}^{K} q_s(t).x_{j,k}(t) \tag{3-2}$$

(2) Startup Delay

Table 3.1: Key Notations

Notation	***Description***
t_v	Total duration of video v.
$s \in S_v(t)$	Segments of video v at time t.
$q_s(t) \in Q_s(t)$	Quality of segment s of video v at time t.
$q_s^{max}(t)$	Representation of segment s with the highest quality at time t .
$q_s^{min}(t)$	Representation of segment s with the lowest quality at time t.
$q_s^{\uparrow}(t)$	Representation of segment s with the next higher quality at time t : $q_s^{\uparrow} \neq q_s^{max}$.
$q_s^{\downarrow}(t)$	Representation of segment s with the next lower quality at time t: $q_s^{\downarrow} \neq q_s^{min}$.
$j \in J(t)$	Set of clients at time t.
$R_{qs}^{j}(t)$	Request generated by client j for the segment s with quality q_s at time t .
$W_k(t)$	Allocates spectrum in number of resource blocks at the base station connected to server k at time t.
$A(t)$	Arrival time of the segment s request at time t.
$D(t)$	Retrieval time of the segment s at time t.
k	CDN surrogate Server.
$T_{b,j}(t)$	Capacity of wireless links between client j and base station b at time t
$Th_s(t)$	Throughput of wired links while downloading segment s at time t.
$B_j^{max}(t)$	Maximum buffer level for the client j at time t.
$B_j(t)$	Buffer level for the client j at time t.
Q_j	QoE perceived by the client j.

This is the time it takes video segments to fill the client's buffer to a target level after its arrival. The client experiences delay as the waiting time from their click to the start of the requested video. While startup delay has a lower impact on the QoE compared to other factors, to minimize the startup delay our algorithm considers a minimum quality for the first segment to fill the buffer [74], a measure which will be described in the next section.

(3) Stalling Time

A stall occurs in video streaming when the client's buffer has emptied before the current video segments have finished playing. Stalls cause playback to be interrupted until further video segments are loaded in the buffer. We formulate the buffer level of client j at time t as follows.

$$B_j(t) = x_{j,k}(t).(B_j(t-1) + (Th_s(t) - q_s(t))), \tag{3-3}$$

(4) Bitrate Switching

Bitrate switching for client j at time t is considered as the difference between the two consecutive downloaded segments [75] as follows:

$$F_j(t) = \sum_{s=1}^{|S_v(t)|} \sum_{k=1}^{K} q_s(t).x_{j,k}(t) - q_{s-1}(t).x_{j,k}(t-1) \tag{3-4}$$

With the model parameters described, the objective function of our model is defined as follows. We assume that each client will download the total segments of a video each time he request it. The objective function (5) maximizes the QoE of DASH over ICN while considering the average quality and bitrate switching parameters; α and β are the weighting parameters. We can tune the objective function using these weighting parameters, for instance, to take into account that the average segment quality has the highest direct impact on the QoE than the bitrate switching we can use $\alpha > \beta$ or to highlight bitrate switching as an important metric as the average segment quality in ICN we can use $\alpha = \beta$

$$Maximize\, Q_j(t) = \alpha.\bar{q}_j(t) - \beta.F_j(t) \tag{3-5}$$

Subject to:

$$\sum_{k=1}^{K} x_{j,k}(t) \leq 1, \tag{3-6}$$

$$\sum_{j=1}^{J(t)} x_{j,k}(t).\lceil q_s(t)/(T_{b,j}(t)\rceil.W_{b,k}(t) \leq W_{b,k}(t), \quad \forall\, 1 \leq k \leq K, \forall\, 1 \leq s \leq |S_v|, \forall\, 1 \leq b \leq BS \tag{3-7}$$

$$0 \leq B_j(t) \leq B_j^{max}(t), \tag{3-8}$$

Constraint (6) states that at time t, the ICN DASH client is allocated to only one server for downloading the segments. Constraint (7) enforces that in time t the total resources allocated to the clients by the base station do not exceed the available resources. Finally, constraint (8) guarantees that limited or no stalling occurs during streaming.

We verify the NP-hardness of the special case of our problem at time t by the mapping our problem to the multidimensional knapsack problem (MDKP) as follows: consider m items $\{c_1, c_2, \ldots, c_m\}$ and K knapsacks, and a case of MDKP in which there are m copies available for each item. The problem is allocating items to knapsacks for objective of maximizing the profit, while satisfying the constraints that each item should appear in exactly one of the allocated copies. Maximum capacity of each knapsack is a maximum of one item. Now, the set of m items and K knapsacks are mapped to the set of clients and servers and the profit maximization in the MDKP is corresponding to the QoE maximization/objective function in our model. Each copy of the item matches the allocation of the corresponding client to a given server. The mapping satisfies the constraint that each client allocates to exactly one server. Since the MDKP is NP-hard [76], therefore this validates NP-hardness of our problem when we consider all time slots. Also the quality of segments is from a discrete set which verifies the hardness of the problem formulation.

3.4. Rate Adaptation Algorithm

In this section, we describe our rate adaptation algorithm to improve users' QoE. This algorithm is designed to achieve higher accuracy and stability in the rate estimation process while avoiding stalls and startup delays. In this proposed algorithm, the accurate RTT of each segment is computed by the RTT calculator and stored in an RTT repository on the MEC layer. The goal is to eventually re-use it as more accurate RTT in future video sessions when the same segment is requested. When a client wants to determine the quality of the next segment, the previously-stored RTT may no longer be completely accurate when it is used again, it is still much more accurate than the one estimated in the traditional way, as shown by the results of our simulations.

The rate adaptation algorithm runs on the client side. It also runs over snapshot which is the time that takes the algorithm to decide about the quality of the next segment based on RTT of previously downloaded segments. The algorithm takes the segments and the segments' quality as its input. The output of the algorithm is the quality of the next segment to be downloaded by the client. The algorithm starts by iterating over all the segments. To avoid any startup delay, we consider minimum quality for the first segment of the video. This will allow the buffer to fill to a certain level to start the streaming.

For subsequent segments, the RTT of each previously downloaded segment is obtained using the function $getRTT()$. The $getRTT()$ function takes as its input the segments along with the arrival time of the request for that segment and the departure time of that segment from the same router. This function returns the most accurate RTT available for that segment which is available in the repository. If there was no RTT available from the previous runs of the same video it will calculate it using the two timestamps it has as input and returns the value: $RTT[s] = D(t) - A(t)$. The algorithm then uses that RTT and calculates the throughput using the size of the segment with the quality of $q_s(t)$. Next, the algorithm reacts to the comparison of segment quality and the throughput of the previously downloaded segment by selecting a lower or higher quality for the next segment.

Algorithm 3.1: ICN Rate Adaptation Algorithm

Algorithm 1 : ICN Rate Adaptation Algorithm

Input: $S_v(t)$,$q_s(t) \in Q_s(t)$; ▷ segments, segments quality
Output: $q_{s+1}(t) \in Q_s(t)$; ▷ quality of the next segment
for *(t in T)* **do**
  if *(s==1)* **then**
    $q_s(t) \leftarrow q_s^{min}(t)$;
  end
  else
    $RTT[s]$=$getRTT()$; ▷ returns updated RTT
    $Th[s]$=$calThroughput(RTT[s])$; ▷ calculating throughput
    if $(q_s^{\uparrow}(t) \leq Th[s])$ && $(q_s(t) != q_s^{max}(t))$ **then**
      if *(allocation of* $q_s^{\uparrow}(t)$ *satisfies* (7) *and* (8)*)* **then**
        $q_{s+1}(t) \leftarrow q_s^{\uparrow}(t)$; **return**
      end
    end
    if $(q_s^{\downarrow} \geq Th[s]$ && $(q_s(t) != q_s^{min}(t))$ **then**
      if *(allocation of* $q_s^{\downarrow}(t)$ *satisfies* (7) *and* (8)*)* **then**
        $q_{s+1}(t) \leftarrow q_s^{\downarrow}(t)$; **return**
      end
    end
  end
end

3.5. Performance Evaluation

This section describes the simulation scenarios used for performance evaluation of our rate adaptation algorithm, followed by the obtained results. The results in this chapter are all applicable to the fixed aspects of CDN delivery. This is due to the fact that the proposed solutions rely on ETSI MEC which caters to both fixed and wireless networks. However, we perform the current validation in a wireless environment, since wireless environments are known to be more challenging than their fixed counterparts.

3.5.1 Evaluation Scenario

We have implemented our proposed solution in a realistic simulation scenario using an extended version of ndnSIM that supports DASH, AMuSt ndnSIM [77]. The dataset for the experiment is the Big Buck Bunny (BBB) dataset [78]. The video has two seconds segment lengths with a

resolution of 1280*720 pixels and five different representations with AVC codec. The MPD file that is included in this dataset is based on the ICN URI namespace. We considered the maximum transmission power of $3.6 * 10^5 mW$ for our base stations, with a loss exponent value of $\omega = 2$. The total LTE downlink resource blocks at each base station follow the uniform distribution $U[50, 100]$ [79] per timeslot. In all scenarios, when the first client requests the segments, all the segments are retrieved from the surrogate servers. These segments are stored in the caches on the paths based on the cache size and random policies. When other clients retrieve the video later, some of the segments may be downloaded from the nearest caches.

We simulate DASH over ICN with the aforementioned general settings performed by a different number of clients who request the same video (BBB). The video resides in 12 CDN surrogate servers connected to the ICN network. We considered a rectangular area of $400m$×$1km$, in which the base stations and the mobile clients are distributed randomly. There are 60 ICN routers between the clients and the servers with random topology. The virtual payload data rate of the links is considered 1 $Mbps$ and the random policy is used for intermediate caches with capacity of 30. We run the test with 10, 20, 30, 40 and 50 clients to investigate the behavior of the model when the number of clients increases. Figure 3.3 illustrates our network topology in the simulations.

3.5.2 Results and Discussion

To compare our results we simulate a DASH over ICN from the literature [80] run it with our topology and same settings. Used it as a baseline and we call it normal DASH for comparison. A normal/traditional DASH over ICN is when the adaptation algorithm decides on the quality of the next segment based on the previous segment's RTT (estimated RTT) without any constraints over stalling events or startup delay. Our proposed DASH provides the client with more accurate RTTs in the MEC repository from the previous run(s) of the same segment while imposing constraints to avoid stalling events and startup delay. The results are taken over 20 times of simulation runs to achieve a higher confidence interval.

We start by comparing the average quality of the two methods when different numbers of the

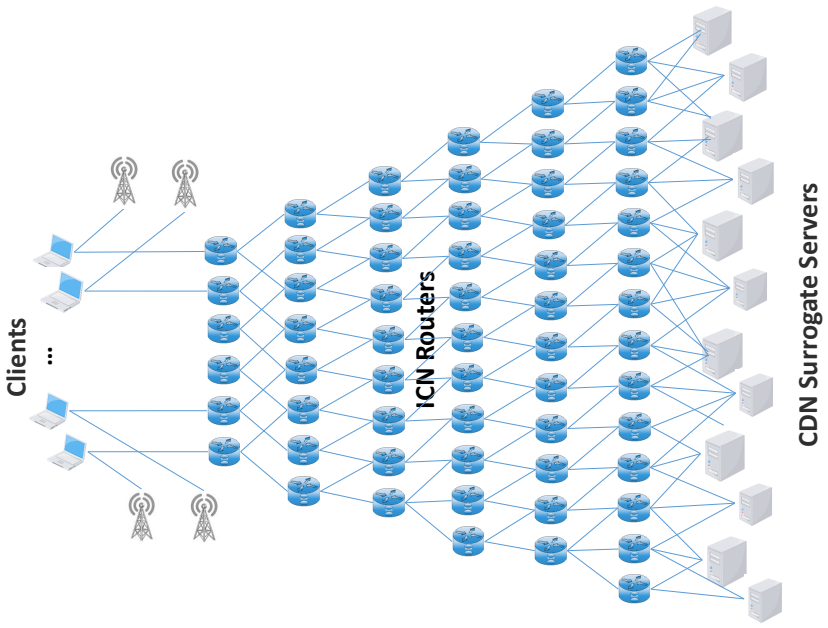


Figure 3.3: Simulation topology.

client are performing DASH. The results are derived with respect to the same simulation settings and the same topology for our work and the baseline. Figure 3.4 indicates the average segment quality for normal DASH in ICN compared to our proposed DASH in ICN with 10, 20, 30, 40 and 50 clients. The simulation results confirm the higher average quality of the segments in our method; which leads to a better quality video playout and higher QoE for the users.

Next, we compare our method with literature (normal DASH), considering the bitrate switching for each set of clients during the whole streaming process. Figure 3.5 depicts the magnitude of the bitrate switching for each set of clients in traditional DASH compared with our proposed DASH. In traditional DASH, the number of switches is much higher than that of our method, which has significantly fewer switches. Fewer switches increase the QoE perceived by end users because there are less oscillations and smoother playout.

Finally, we compare two methods, considering the stalling time. Figures 3.6 depicts the stalling time for each segment of the video for the last client in normal DASH while stalling is prevented in our method. In order to evaluate these figures, the most tolerable level of stalling event based on previous studies which is 3 seconds [81] is added to the figures. In traditional DASH the duration

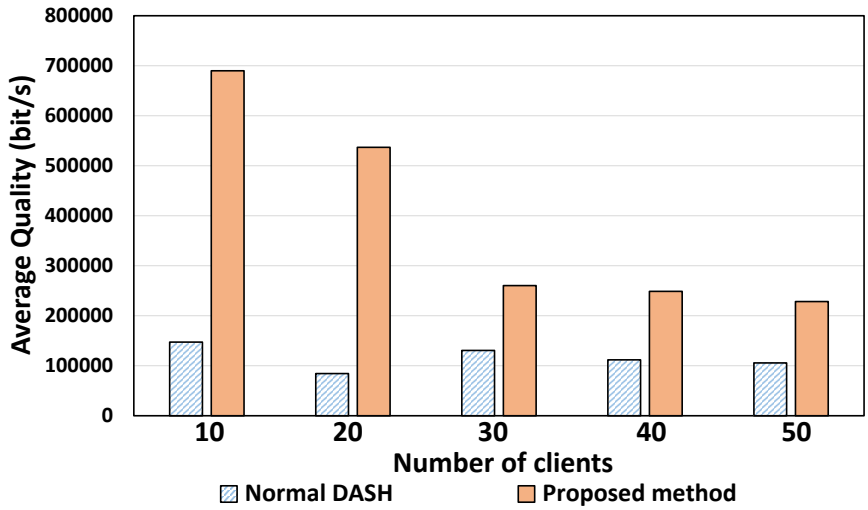


Figure 3.4: Average segment quality in two DASH methods for five set of clients.

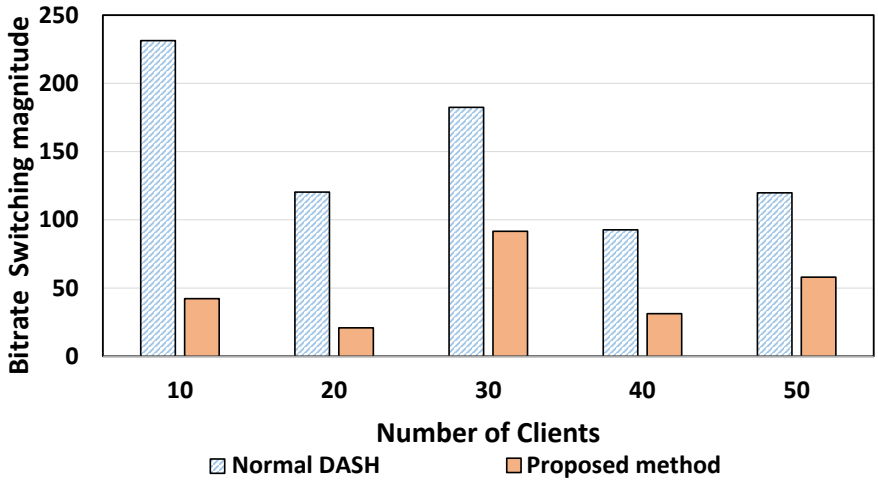
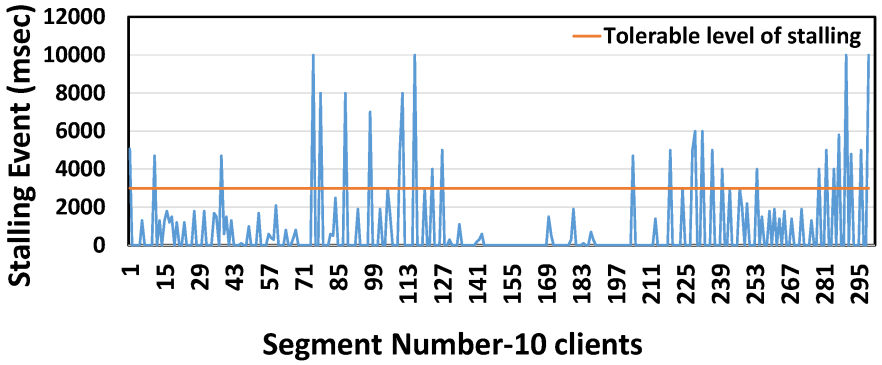


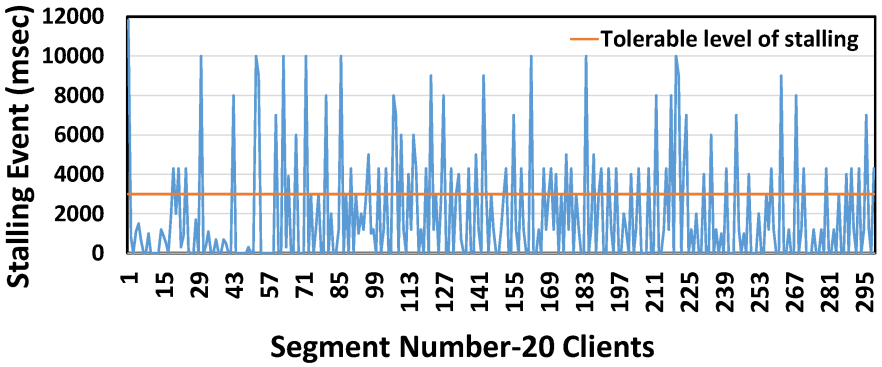
Figure 3.5: Bitrate switching in two DASH methods for five set of clients.

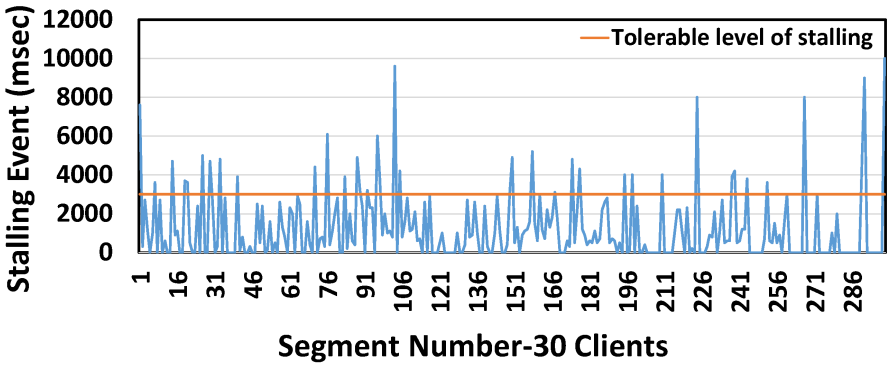
and frequency of the stalling times are high and exceeded the threshold many times while our method has no stalls.

There are 60 ICN routers with caching capability between each set of clients and servers. We used random caching policy in the simulation so some segments with different qualities randomly are cached throughout the network. When segments with better qualities are caches in the nearby caches to the clients, the stalling events decrease. Due to the caching replacement randomness and misunderstanding of the network condition some cases such as the second scenarion in figure 3.6

with 20 clients experience more stalling. It happens that the requested segments are not available in the nearby caches, this causes depleting the buffer resulting in re-buffering and more stalling events. We can say that fewer stalls ensure smoother playout and increase the QoE perceived by end users.







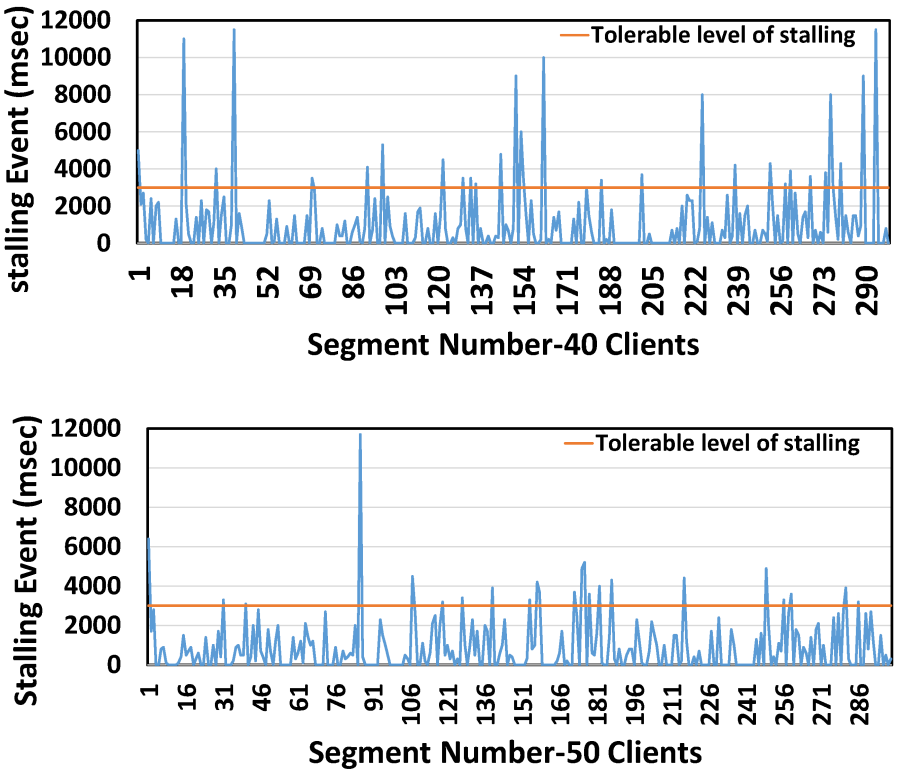


Figure 3.6: Stalling events in normal DASH.

3.6. Conclusion

This chapter addresses the main challenges faced by rate-based DASH in in-network storage enabled CDNs, namely accuracy, and stability in rate estimation. We introduce an overall system view that includes an ETSI MEC server. In addition, we propose a novel rate adaptation algorithm. We evaluated the results by conducting a set of experiments and compared them with results from the state of the art. Our method performed better than the traditional approach in the simulations. It provides higher accuracy and stability in the rate estimation. The results show that our method offers improvements in the QoE metrics in rate-based DASH for information centric based CDNs.

Chapter 4

An Algorithm for Implementing In-Network Storage in IP Settings [1]

4.1. Introduction

The tremendous growth of video traffic is bringing new challenges (e.g., scalability and content distribution efficiency) that current CDNs are less and less able to meet. CDNs are designed to deliver requested content at a reasonable cost and at the required QoS. However, CDNs face challenges in terms of performance, reliability, and scalability [82]. ICNs have inherited an in-network storage feature, so requests for content in ICN-based CDNs would not need to go to surrogate servers as in IP-based CDNs. Instead, they could be served by several routers that are usually closer to end-users than to surrogate servers, thereby enhancing performance and scalability. Unfortunately, while they are attractive for Greenfield CDN providers, the deployment of ICN-based CDNs is very costly for Brownfield providers, as it requires the deployment of cache-enabled routers. We rely on slicing to build a virtual ICN core network (i.e., a core network with in-network storage) on top of a physical infrastructure that supports traditional IP routers but does

[1] This chapter is based on two published papers:[1] Rayani, Marsa, et al. "Ensuring Profit and QoS When Dynamically Embedding Delay-Constrained ICN and IP Slices for Content Delivery." IEEE Transactions on Network Science and Engineering 9.2 (2021): 769-782; and [2] Rayani, Marsa, et al. "Slicing virtualized EPC-based 5G core network for content delivery." IEEE Symposium on Computers and Communications (ISCC). IEEE, 2018.

not support ICN routers with in-network caching. We assume that storage is available in the physical infrastructure. The key benefit is that traditional IP-based CDNs and ICN-based CDNs can co-exist on the same infrastructure in a flexible and cost-efficient manner. The basics of network slicing and ICNs are discussed next. The rest of this chapter is organized as follows.

After our discussion of the overall system view, we present our problem formulation and describe the proposed dynamic slicing algorithm. Next, we present the performance evaluation and finally, in the last subsection, we conclude this chapter.

4.2. Overall System View

Network slicing can be performed statically or dynamically. With static slicing, each slice reserves a fixed amount of resources for its complete life cycle. Static slicing leads to an inefficient utilization of resources, as it does not allow the allocation of resources to adapt to the changing resource demands of slices. In contrast to static slicing, dynamic slicing facilitates such adaptability. Dynamic slicing allows for an efficient use of infrastructure resources with possible cost-based and profit-oriented optimizations. We aim to dynamically allocate resources for a set of traditional IP and ICN core network slices over a substrate network. We consider that resource allocation is done on a snapshot basis, where a snapshot represents the aggregated resource demand of slices over a specific time period. To embed each slice, resource allocation is conducted at the beginning of each snapshot. We do not impose any restrictions on the duration of a snapshot. It can be on the order of seconds, minutes or hours. We solve the problem in consecutive snapshots to show the dynamicity of the provisioning process of network slicing. For instance, a provider may divide the 24 hours of a day into three snapshots based on the incoming traffic pattern of the network. To adapt to the traffic demand fluctuations, the provider may consider the beginning of a snapshot as the time when the traffic elevates. Thus, the size of the requested slices can be proportional to the traffic demand at that snapshot. This approach therefore does not restrict the providers with any specific length for a snapshot and instead leads to a dynamic configuration based on the incoming traffic pattern.

The substrate network offers computing and storage resources. Each network slice is formed by a set of VNFs. A traditional IP core network slice only contains traditional IP routers, and thus only contains computing VNFs. In contrast, an ICN core network slice contains cache-enabled routers, capable of caching content. Each cache-enabled ICN router is formed by one computing and one storage VNF, and thus each ICN slice includes a mixture of computing and storage VNFs. Our objective is to enable the dynamic resource allocation of slices in order to create the highest profit for the infrastructure provider while still meeting the QoS requirements for content delivery.

Figure 4.1 depicts our overall architecture: a physical substrate network that offers computing resources and storage; ICN core network slices with cache-enabled routers and traditional IP core network slices; an ICN-based CDN built on the ICN core network slice; and a traditional IP-based CDN built on the IP core network slice. These computing resources may be traditional IP routers, and the storage can be available in the network or even in servers at the edge.

The reader should note that these slices may be used by other applications that have requirements similar to those of content delivery. At the beginning of a snapshot we admit all the slices' requests accumulated during the previous snapshot. We assume that the number of slices that can be allocated simultaneously does not surpass the maximum number of slices allowed for the infrastructure. Each slice request includes the VNF instances, their types and their resource demands, as well as the links between the VNFs and their bandwidth demands. A slice may be requested over consecutive snapshots with the same set of VNF instances and the same set of links, but with different resource and bandwidth demands. Accordingly, we allocate infrastructure resources for each slice in a snapshot t on a First Come First Serve (FCFS) basis.

We do not allow for a partial embedding, meaning that each slice will be fully served and embedded at a given snapshot when admitted by the infrastructure provider.Our model considers the following mechanisms with which to adapt to changes: (1) allocate resources to a new set of admitted slices, (2) de-allocate resources for VNFs removed from past slices that remain in snapshot t, and (3) migrate VNFs from past slices that remain in snapshot t to new infrastructure nodes. We present our system model below.

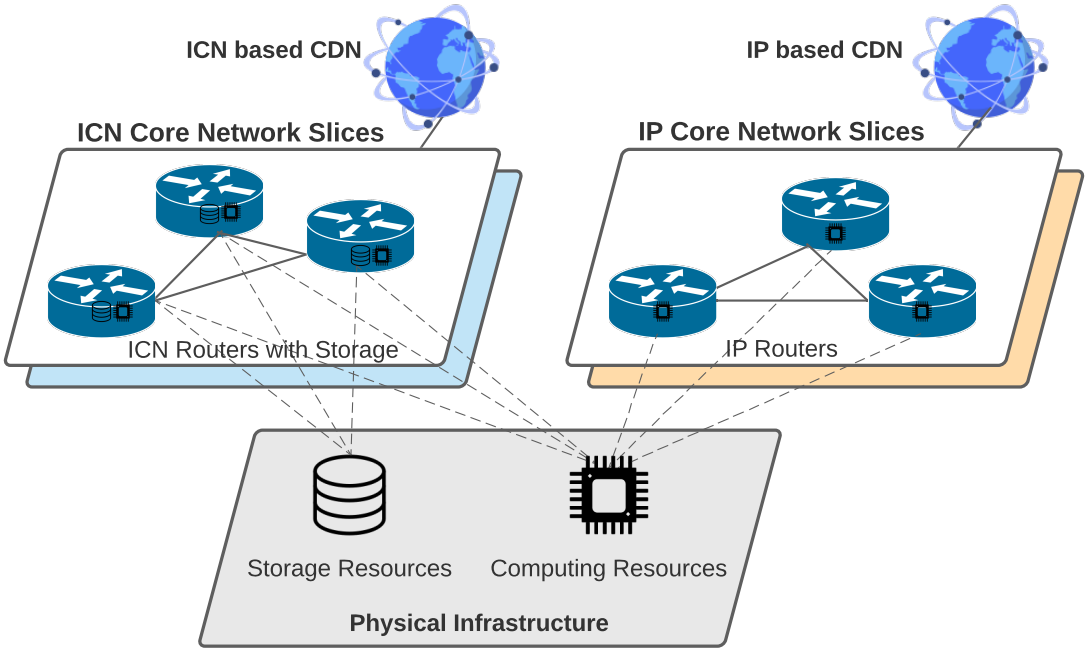


Figure 4.1: High-level architecture.

(1) Substrate Network: We represent the substrate network as an undirected graph $IG = (N, L)$ where N is a set of nodes, with each node n representing a computing resource or storage resource at the substrate network and where L is a set of edges linking them. An edge $(n, n') \in L$ linking nodes n and n' represents a logical communication link between them. Each node n has a type $k_n \in K$, where $K = \{computing, storage\}$. We use R_n to refer to the resource capacity of node n. We employ c_n^t and $c_{n,n'}^t$ to denote the cost of one resource unit at node n and one unit of network bandwidth over the edge (n, n') in snapshot t. We use $B_{nn'}^t$ and $d_{nn'}^t$ to represent the bandwidth capacity and delay, respectively, of edge (n, n') in snapshot t.

(2) Slices: We define S^t as the set of two different types of slices (IP slice and ICN core network slice) to map to the substrate network in snapshot t. Each slice $s^t \in S^t$ includes a set of VNFs V_s^t to map to the substrate network nodes. We associate a set of user crowds U_s^t and a set of given surrogate servers $W_s^t$to a slice $s^t \in S^t$.

We build a graph $SG_s^t = (H_s^t, E_s^t)$ to represent slice s^t, in which H_s^t is a set of nodes defined as $H_s^t = V_s^t \cup U_s^t \cup W_s^t$ and $E_s^t = E_{(s,V)}^t \cup E_{(s,U)}^t \cup E_{(s,W)}^t$ is a set of edges that link nodes in H_s^t. $E_{(s,V)}^t$ are edges that exist between the VNFs in slice s. $E_{(s,U)}^t$ denotes the set of edges that exist between users and VNFs in slice s. $E_{(s,W)}^t$ denotes the set of edges connecting the VNFs and CDN surrogate servers in snapshot t. The parameter $k_v \in K$ is used to represent the type of VNF $v \in V_s^t$. For instance, a traditional IP network slice only offers computing-type VNFs. Instead, since an ICN slice relies on cache-enabled routers, it has both computing and storage types of VNFs. We use r_v^t to refer to the resource demand of a VNF $v \in V_s^t$ in snapshot t. Each link $(v, v') \in E_s^t$ requires $b_{vv'}^t$ bandwidth units in snapshot t. Table 4.1 lists the notations used in this section.

Figure 4.2 and 4.3 illustrate our slice graphs to clarify the definitions of different types of slices and the relation between the user crowds, slice and surrogate servers. The IP core network slice graph is illustrated in Fig. 4.2 which is made of routers with computing capabilities, surrogate servers and user crowds. Figure 4.3 shows the ICN core network slice graph, which is a network of routers with storage and computing capabilities, surrogate servers and user crowds connected to the slice.

Table 4.1: Key Notations.

Notation	Description
$IG = (N, L)$	NFVI graph with nodes N and edges L linking them.
$n \in N$	A node in NFVI.
$(n, n') \in L$	An edge in L.
k_n	A type of node $n \in N$.
R_n	Resource capacity of a node $n \in N$.
c_n^t	Cost of one resource unit at node $n \in N$ in snapshot $t \in T$.
$c_{nn'}^t$	Cost of one unit of network bandwidth over the edge $(n, n') \in L$ in snapshot $t \in T$.
$B_{nn'}$	Bandwidth capacity of edge $(n, n') \in L$.
$d_{nn'}^t$	Delay of edge $(n, n') \in L$ in snapshot $t \in T$.
S^t	Set of slices in a snapshot at time $t \in T$.
$SG_s^t = (H_s^t, E_s^t)$	Graph of slice s with VNFs H_s^t and edges E_s^t linking them.
V_s^t	Set of VNFs in slice s in snapshot $t \in T$.
U_s^t	Set of user crowds in slice s in snapshot $t \in T$.
W_s^t	Set of surrogate servers in slice s in snapshot $t \in T$.
$E_{S,V}^t$	Set of edges between VNFs in slice s in snapshot $t \in T$
$E_{S,U}^t$	Set of edges between VNFs and user crowds in slice s in snapshot $t \in T$.
$E_{S,W}^t$	Set of edges between VNFs and surrogate servers in slice s in snapshot $t \in T$.
v	VNF $v \in H_s^t$.
$(v, v') \in E_s^t$	An edge in E_s^t.
k_v	Type of VNF $v \in V_s^t$.
r_v^t	Resource demand of a VNF $v \in V_s^t$ in snapshot $t \in T$.
$b_{vv'}^t$	Bandwidth demand for the link $(v, v') \in E_s^t$ in snapshot $t \in T$.
$SG_{s'}^t = (H_{s'}^t, E_{s'}^t)$	Graph of sub-slice s' with VNFs $H_{s'}^t$ and edges $E_{s'}^t$ linking them.
M_{k_v}	Bandwidth consumed in migrating the VNF of type k_v.
$c_{mig,v}^t$	Migration cost of the VNF $v \in V_s^t$ in snapshot $t \in T$.
$c_{rea,v}^t$	Reassignment penalty of the VNF $v \in V_s^t$ in snapshot $t \in T$.
$D_{s'}^t$	Maximum tolerable delay between a crowd of users and the surrogate server for each sub-slice s'.

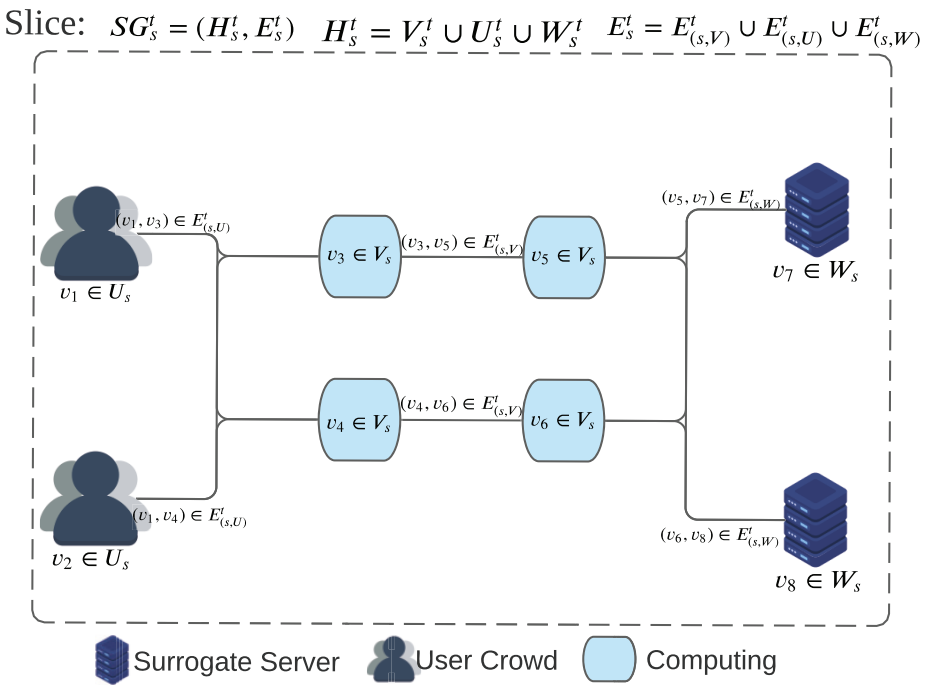


Figure 4.2: Illustration of IP slice graph.

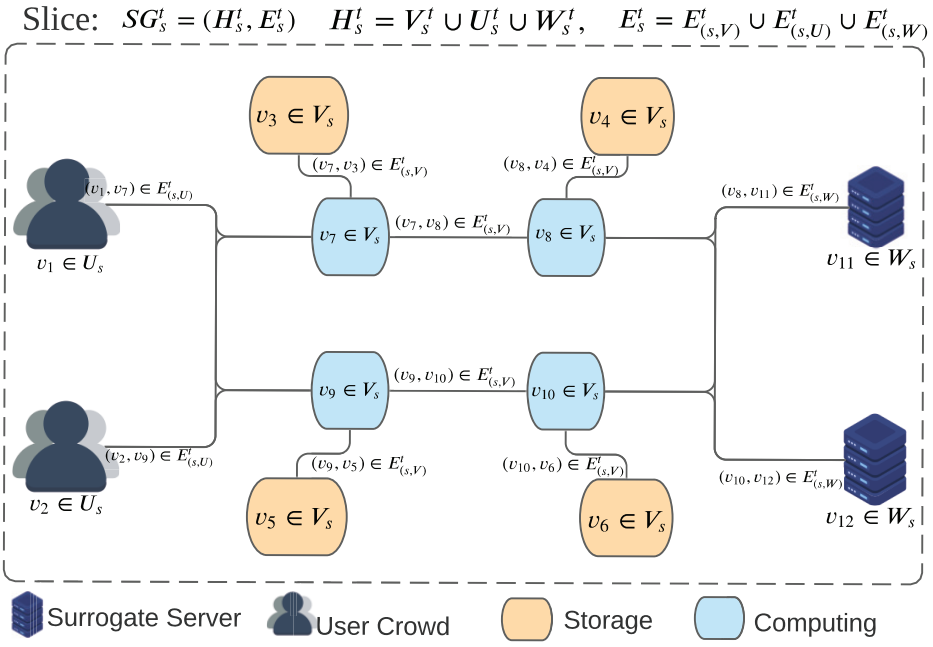


Figure 4.3: Illustration of ICN slice graph.

4.3. Problem Formulation

Here we develop our problem formulation. Given that we use a linear objective function along with linear constraints and integer decision variables, our formulation leads to an ILP [83] by defining the following decision variables:

$$x_{nv}^{t} = \begin{cases} 1, & \text{if slice VNF } v \in V_s^t \text{ is placed at infrastructure node } n \in N \text{ in snapshot } t \\ 0, & \text{otherwise.} \end{cases} \tag{4-1}$$

$$y_{vv'nn'}^{t} = \begin{cases} 1, & \text{if slice link } (v, v') \in E_s^t \text{ is mapped to infrastructure link } (n, n') \in L \text{ in snapshot } t \\ 0, & \text{otherwise.} \end{cases} \tag{4-2}$$

Operational cost: The operational cost has four different components, defined as follows.

(1) Deployment cost (C_{dep}^{t}): Represents the cost of resources over infrastructure nodes, allocated for VNFs in snapshot t, and given by:

$$C_{dep}^{t} = \sum_{s^t \in S^t} \sum_{n \in N} \sum_{v \in V_s^t | k_v = k_n} c_n^t r_v^t x_{nv}^t \tag{4-3}$$

(2) Communication cost (C_{com}^{t}): Represents the cost of the network bandwidth in the core slices in snapshot t, expressed as:

$$C_{com}^{t} = \sum_{s^t \in S^t} \sum_{v,v' \in E_s^t} \sum_{n,n' \in L} c_{nn'}^t b_{vv'}^t y_{vv'nn'}^t \tag{4-4}$$

(3) Migration cost (C_{mig}^{t}): Represents the cost of migrating VNFs from one infrastructure node

to another while switching from snapshot $t-1$ to t, given by:

$$C^t_{mig} = \sum_{s^t \in S^t} \sum_{v \in V_s^{t-1} \cap V_s^t} \sum_{n \in N} \sum_{n' \in N} x^t_{n'v} x^{t-1}_{nv} c^t_{mig,v} \tag{4-5}$$

where $c^t_{mig,v}$ is the migration cost of node $v \in V^t_s$ in snapshot $t \in T$.

(4) Reassignment cost (C^t_{rea}): Any Migration between snapshots $t-1$ and t of two VNFs implies a QoS violation. This QoS violation in turn implies a penalty from the infrastructure provider to the slice owner. We refer to this penalty as the reassignment cost/penalty, obtained as follows:

$$C^t_{rea} = \sum_{s^t \in S^t} \sum_{v \in {V_s}^{t-1} \cap {V_s}^t} \sum_{n \in N} \sum_{n' \in N} x^t_{n'v} x^{t-1}_{nv} c^t_{rea,v} \tag{4-6}$$

where $c^t_{(rea,v)}$ represents the reassignment cost of node $v \in V^t_s$ in snapshot $t \in T$.

The expenditure of the infrastructure provider in snapshot t can be calculated as:

$$EXP^t = C^t_{dep} + C^t_{com} + C^t_{mig} + C^t_{rea} \tag{4-7}$$

The revenue of an infrastructure provider is a weighted aggregation of deployment, communication and migration costs in snapshot t and can be calculated as:

$$REV^t = \beta(C^t_{dep} + C^t_{com} + C^t_{mig}) \tag{4-8}$$

By subtracting the expenditure from the revenue, the profit of the infrastructure provider P^t generated by allocating resources for a set of slices can be calculated as:

$$P^t = REV^t - EXP^t \tag{4-9}$$

The objective of our problem is to maximize the overall profit to be gained by provisioning

slices over a snapshot t for the infrastructure provider who owns the resources:

$$\max \ P^t \tag{4-10}$$

Constraints: The following constraints are considered in our problem. Each VNF instance should be mapped to one infrastructure node, as indicated in constraint (9):

$$\sum_{n \in N | k_v = k_n} x^t_{nv} \leq 1; \forall v \in V^t_s, s^t \in S^t, t \in T \tag{4-11}$$

Constraint (10) ensures that each slice link should be mapped to one infrastructure link:

$$\sum_{(n,n') \in L} y^t_{vv'nn'} \leq 1; \forall (v, v') \in E^t_{s,V}, s^t \in S^t, t \in T \tag{4-12}$$

Constraint (11) ensures that the number of VNFs placed over an infrastructure node does not exceed its capacity:

$$\sum_{s^t \in S^t} \sum_{v \in V^t_s | k_v = k_n} r^t_v . x^t_{nv} \leq R_n; \forall n \in N, t \in T \tag{4-13}$$

Constraint (12) is imposed so that the bandwidth capacity of an infrastructure link is not exceeded:

$$\sum_{s^t \in S^t} \sum_{(v,v') \in E^t_s} B^t_{alloc} + B^t_{mig} \leq B_{nn'}; \forall (n, n') \in L, t \in T \tag{4-14}$$

where B^t_{alloc} and B^t_{mig} are given by:

$$B^t_{alloc} = b^t_{vv'} y^t_{vv'nn'}$$

and

$$B^t_{mig} = x^t_{nv} x^{t-1}_{n'v} M_{k_v}$$

We assume that the locations of user crowds and surrogate servers are known in advance, specified by matrices **P** and **Q**, respectively, given by:

$$p_{nv} \in \mathbf{P} = \begin{cases} 1, & \text{if } v \in U_s^t \text{ is located at } n \in N \\ 0, & \text{otherwise} \end{cases}$$

$$q_{nv} \in \mathbf{Q} = \begin{cases} 1, & \text{if } v \in W_s^t \text{ is located at } n \in N \\ 0, & \text{otherwise} \end{cases}$$

We thus link variables x_{nv}^t and $y_{vv'nn'}^t$ to edges in $E_{s,U}^t$ and $E_{s,W}^t$ as follows:

$$y_{vv'nn'}^t = x_{nv}^t p_{n'v'};$$
$$\forall s^t \in S^t, (v, v') \in E_{s,U}^t, (n, n') \in L, t \in T,$$

$$y_{vv'nn'}^t = x_{nv}^t q_{n'v'};$$
$$\forall s^t \in S^t, (v, v') \in E_{s,W}^t, (n, n') \in L, t \in T.$$

We link the variables x_{nv}^t and $y_{vv'nn'}^t$ for edges (v, v') in $E_{s,V}^t$ via the following constraint:

$$y_{vv'nn'}^t = x_{nv}^t x_{n'v'}^t;$$
$$\forall s^t \in S^t, (v, v') \in E_{s,V}^t, (n, n') \in L, t \in T \quad \text{(4-15)}$$

Given that Constraint (4-13) is not linear, we linearize it as follows:

$$y^t_{vv'nn'} \leq x^t_{nv};$$
$$\forall s^t \in S^t, (v,v') \in E^t_{s,V}, (n,n') \in L, t \in T,$$

$$y^t_{vv'nn'} \leq x^t_{n'v'};$$
$$\forall s^t \in S^t, (v,v') \in E^t_{s,V}, (n,n') \in L, t \in T,$$

$$y^t_{vv'nn'} \geq x^t_{nv} + x^t_{n'v'} - 1;$$
$$\forall s^t \in S^t, (v,v') \in E^t_{s,V}, (n,n') \in L, t \in T.$$

To reflect different QoS for different levels of caching, we consider a maximum latency $D^t_{s'}$ imposed to meet users' demands in snapshot t. For an ICN network core slice, the threshold needs to be met when serving a user from a storage VNF or from the surrogate server. For a traditional IP network slice, the threshold needs to be met when serving users from the surrogate server. To formulate this difference, we define $SG^t_{s'} = (H^t_{s'}, E^t_{s'})$ as a sub-graph of SG^t_s with $H^t_{s'} \subset H^t_s$ and $E^t_{s'} \subset E^t_s$. $SG^t_{s'}$ represents the flow from one cache or the flow from the original server to a crowd of users. The sub-graphs $s'^t \in S'^t$ represent the sub-slices. The maximum latency for each sub-slice is then imposed by the following constraint:

$$\sum_{v,v' \in E^t_{s'}} \sum_{n,n' \in L} d^t_{nn'}.b^t_{vv'}.y^t_{vv'nn'} \leq D^t_{s'}; \quad \forall s'^t \in S'^t \tag{4-16}$$

4.4. Dynamic Slicing Algorithm

Our problem shares several similarities with the VNE problems. Exact, approximation and heuristic algorithms have been employed to solve VNE problems as ways to solve the challenge

of resource allocation. However, VNE problems are known to be NP-hard [84]. As a result, in large-scale scenarios, deriving optimal solutions becomes unfeasible, leading to the development of algorithms.

To solve our problem, we introduce a heuristic that aims to place slices where they will create the highest possible profit for the infrastructure provider. The heuristic is run at the beginning of each snapshot and has four phases. It iterates over the set of slices to place its corresponding VNFs while considering resource capacity ($n.cap$), the bandwidth requirement ($Bwdreq$) and the QoS limits ($QoS[s]$). We describe the details of the heuristic below. The corresponding algorithm is presented in the four phases of Algorithm 1.

The heuristic takes the infrastructure graph $IG(N, E)$ that comes with node types ($n.type$), the node capacity ($n.cap$), and the bandwidth capacity ($Bwdcap$) as its first input. Its second input is the slice graph $SG(N_v, E_v)$, which has information about the VNF slice types ($v.type$), the VNF demands ($v.req$), the bandwidth demand ($Bwdreq$) and the QoS thresholds. The final results of the heuristic are the placement of the slice functions on the infrastructure nodes n, (x) and the placement of the links between them, (y).

The heuristic starts by iterating over all the slices. The $subslices()$ function takes the slice as its input and returns an array of corresponding sub-slices sorted in ascending order of their QoS thresholds. This sorting occurs because the heuristic prioritizes the placement of sub-slices with more stringent QoS requirement. The heuristic then iterates over the sorted list of sub-slices $(subsliceToMap)$. For each sub-slice the function $VNF()$ returns all the VNFs. The phase one of the heuristic begins, which forms a set of potential nodes for each VNF in which to place them. A set of potential nodes is first formed for each of the functions belonging to that specific sub-slice. The potential nodes are infrastructure nodes where the functions can be placed. The placement on these nodes will satisfy the following set of constraints: be of a same type (computing or storage), have an adequate node resource $(n.cap)$ and link bandwidth capacity $(Bwdcap)$, and finally, if the delay incurred by using them $(TotalDelay)$ satisfies the QoS requirement. To verify the delay constraint, the $delay()$ function is used; it returns the delay between the infrastructure node in

question and the crowds of users to be served. For a specific function in a sub-slice and an infrastructure node in question, if the delay is smaller than the QoS threshold, the infrastructure node is considered as a potential node.

Algorithm 4.1: Dynamic Slicing Algorithm

Input: $IG(N, E)$ ▷ Infrastructure graph
S^t ▷ slices $SG(H_v, E_v)$ at time t
Output: x, y ▷ Placement of all admitted slices
$RejectNodes() = []$;
$subsliceToMap() = []$;
for (S in S^t) **do**
  $subsliceToMap$(S) = **subslices**(S); ▷ **subslices**(S) returns all sub-slices of slice S
  $VNFtoMap() = []$;
  while ($subsliceToMap$(S) $\neq \varnothing$) **do**
    $VNFtoMap$(s) = VNF(s); ▷ **VNF**(s) returns all VNFs of sub-slice s

Phase 1 – Finding potential nodes n to place VNF v

    for (v in $VNFtoMap$(s)) **do**
      $PotentialNodes(v) = []$;
      $RejectNodes(v) = []$;
      $TotalDelay \leftarrow 0$;
      v'=**PreviousVNF**(s, v); ▷ **PreviousVNF**(s, v) returns preceding function of VNF v in a sub-slice s
      $n' =$ **Infra**(v'); ▷ **Infra**(v') returns infrastructure node where previous VNF is placed
      for ($n = 0$ to $|N| - 1$) **do**
        if ($v.type$=$n.type$)&($v.req \leq n.cap$)&($Bwd_{req}[vv'] \leq Bwd_{cap}[nn']$)&($n \notin RejectNodes(v)$) **then**
          $TotalDelay \leftarrow$ **delay**(n, v); ▷ **delay**(n, v) returns delay between a node and crowds of users
          if ($TotalDelay \leq QoS[s]$) **then**
            $PotentialNodes(v).add(n)$;
          end if
        end if
      end for
      if ($|PotentialNodes(v))| \neq 0$)) **then**

After forming the set of potential nodes, we calculate the profit for each node by using the $calculateProfit()$ function. If we find a potential node for a function, phase 2 starts. In this phase, the heuristic finds the potential node that has the maximum profit ($\hat{n}$) and places the VNF on it. At this stage, the placement decision for the node (x) and link (y) will be updated. Following the placement, the amount of resource requirements $(v.req)$ and the bandwidth demand $(Bwdreq)$ of the function will be deducted from the node resources $(n.cap)$ and the bandwidth capacity $(Bwdcap)$ of the infrastructure. The VNF is now removed from the set of VNFs to map. If all the VNFs in a sub-slice are placed, that sub-slice can be removed from the set $subslicesToMap$.

Phase 2 – Finding infrastructure node $\hat{n}$ with maximum profit

$P() = []$;
for (n in $PotentialNodes(v)$) **do**
  $P(n)$=**Profit**(n, v); ▷ **Profit**(n, v) returns profit of infrastructure node n
end for
$P(\hat{n}) = max(P(n))$; ▷ node $\hat{n}$ with maximum profit
$x[\hat{n}][v] \leftarrow 1$; ▷ VNF v is placed on node $\hat{n}$
$y[\hat{n}n'][vv'] \leftarrow 1$; ▷ edge $[vv']$ is mapped on link $[\hat{n}n']$
$\hat{n}.cap \leftarrow \hat{n}.cap - v.req$;
$Bwd_{cap}[\hat{n}n'] \leftarrow Bwd_{cap}[\hat{n}n'] - Bwd_{req}[vv']$;
$VNFtoMap$(s).$remove(v)$;
$subsliceToMap$(S).$remove(s)$;
if ($|subsliceToMap$(S)$|$)=0 **then**
  $S^t.remove(S)$;
end if
else

Phase 3 – Reconsidering previous decisions

$x[n'][v'] \leftarrow 0$; ▷ removes previously placed VNF v' from node n'
$y[nn'][vv'] \leftarrow 0$; ▷ removes previously placed slice link vv' from infrastructure link nn'
$VNFtoMap(s).add(v')$;
$RejectNodes(v').add(n')$;
$n'.cap \leftarrow n'.cap + v'.req$;
$Bwd_{cap}[nn'] \leftarrow Bwd_{cap}[nn'] + Bwd_{req}[vv']$;

If all the sub-slices in a slice are mapped, the slice will be removed.

If there is no potential node for the selected VNF in a sub-slice, we enter phase 3 and reconsider the decisions of previously placed functions inside the same and other sub-slices. This phase makes some infrastructure nodes available to be a potential node for the placement. The previous function (v') is obtained, along with the infrastructure node (n') where the preceding function was placed, using functions $PreviousVNF()$ and $Infra()$. After removing the VNFs from the infrastructure node where they were previously placed, we add that node to a set of rejected nodes to prevent placing the same function on the same node again. We then update the infrastructure resources.

One VNF can be in more than one sub-slice at a time, and so it is necessary to check if the removed VNF belongs to another sub-slice in phase 4. If there are dependent sub-slices they must be mapped again. To determine if a previous function has any other dependent VNFs after it in its sub-slice, the $getDependingVNFs()$ function is used to identify and return any dependent VNFs. These dependent VNFs must be added to the set of VNFs to be mapped again. The infrastructure resources also need to be updated after the removal. These phases are all performed iteratively until a feasible placement is found.

Phase 4 – Reconsidering sub-slices composed of removed VNF v'

for $(s \in subsliceToMap(\mathrm{S}))$ **do**
  if $(v' \in VNFtoMap(s))$ **then**
    $subsliceToMap(\mathrm{S}).add(s)$;
  end if
end for
$DependingVNFs(s, v') =$ **getDependingVNFs**(s, v'); ▷ **getDependingVNFs**(s, v) returns placed VNFs after v in sub-slice s
for $(v \in DependingVNFs(s, v'))$ **do**
  $n =$ **Infra**(v);
  $x[n][v] \leftarrow 0$;
  $y[nn'][vv'] \leftarrow 0$;
  $Bwd_{cap}[nn'] \leftarrow Bwd_{cap}[nn'] + Bwd_{req}[vv']$;
  $n.cap \leftarrow n.cap + v.req$;
  $VNFtoMap(s).add(v)$;
end for
end if
end for
end while
end for

4.5. Performance Evaluation

This section presents our evaluation scenario followed by results and discussion of the simulation.

4.5.1 Evaluation Scenario

In the dynamic network slicing experiment, our model aims to: (1) evaluate the proposed algorithm in terms of the quality of the solution and the execution time; (2) study the profit that an infrastructure provider can achieve by re-configuring the placements when the slicing process evolves over time; and (3) investigate the effect of QoS violation penalties on infrastructure provider profits.

The evaluation parameters were selected from the literature. We built onto the WonderNetwork [85] to construct the substrate network used in our evaluation. The WonderNetwork is a network provider that operates over a network of 230 servers in different locations in several countries. We considered locations of this network in the US as our infrastructure's nodes. This gives our substrate network a total of 60 nodes with 3600 (logical) links between them spread all over the US, with 20 storage resources at the edge [86] and the remaining set of core routers, all offering

computing resources.

The WonderNetwork provides hourly ping delay in real time for each pair of nodes in the network. We considered the latency of the links among substrate nodes to map to the delay of the ping time for each snapshot ($d^t_{nn'}$), obtained from global statistics data from the WonderNetwork. The cost of infrastructure resources (c^t_n) for each infrastructure node was found to be in the price range of 8.7 to 18.4 $\$/$kWh, which is within the real-world hourly electricity price range for US states, obtained from real data in [87]. We assumed that each edge has 10 Mbps of bandwidth capacity ($B^t_{nn'}$), and that the bandwidth unit cost ($c^t_{nn'}$) for all edges is equal to 0.155 $\$/$GB [87].

We considered the presence of two classes of core slices in the system, ICN and Traditional IP. For the ICN slices, the VNF types are both storage and computing (because ICN routers are all cache-enabled). For the Traditional IP slices, only computing VNFs are present. In the experiment for the process of live migration, a VM is hosting a function in its running state (e.g., memory and a virtual disk) and should be transferred [88]. We consider that the virtual machine hosting the slices' VNFs is a medium-sized server with 2 CPUs, a disk size of 40 GB and 4 GB of memory [89]. The bandwidth consumed in migrating (M_{k_v}) a VNF from one node to another is equal to the aggregation of its memory and disk size, i.e., 44GB. The migration cost $c^t_{mig,v}$ is equal to the cost of the bandwidth consumed during the migration: 44 $\times 0.155$ [89].

The maximum tolerable delay between the user crowds and the surrogate servers in each sub-slice ranges within 150 to 200 ms [90]. The lifetime of each slice is assumed to be 24 hours in our scenario. We set the reassignment penalty to $2, which is around one third of the function migration cost (see Table 4.2). We note that for small values of reassignment penalty, our proposed algorithm tends to do as many reassignments as possible to maximize the profit. This, as a result, leads to a poor quality of service. On the other hand, if the reassignment penalty is set to large values, the proposed algorithm tries to maintain the quality of service without making any profit. Thus, the reassignment penalty is key to realize the trade-off between making profit and maintaining quality of the service (to be further examined later, in Section 4.5.2).

For our simulation we consider that a special event, for instance, the Super Bowl, is happening

in the US and causing occasional spikes in traffic over consecutive hours of the day. Based on this event, the traffic is increasing and concentrated during the simulated hours so that SPs instantiate requests for more slices in order to scale and meet the demand. The number of user crowds is selected randomly from $[1, 10]$, each containing multiple users. The number of users within each crowd is on the order of the size of the considered wide area network (WAN) spanning several US cities. Table 4.2 lists the simulation scenario parameters.

Table 4.2: Summary of Simulation Parameters.

Prameter	Value
Infrastructure Netwrok	
Substrate nodes	60
Substrate links	3600
Delay of each link between nodes of substrate (ms)	[9-63]
Cost of substrate resources for each node ($/kWh)	[8.7-18.4]
Substrate bandwidth capacity (Mbps)	10
Bandwidth unit cost $/GB	0.155
Slice	
Class of slices	ICN and IP
Number of VNFs	10-120
Number of sub-slices	12-80
Number of slices	2-12
Migration cost for VNFs ($)	6.82
Reassignment penalty ($)	2
Maximum tolerable delay for each sub-slice (ms)	150-200

4.5.2 Results and Discussion

Here, we evaluate the performance of our heuristic for dynamic slicing by showing different plots of our obtained results and comparing them with an optimal solution from CPLEX and another solution from the literature (MPSP) [91]. The slicing algorithm is implemented in JAVA. All the tests were run using a machine with a 7-core 2.3 GHz Intel®3615QM CPU and 8GB memory.

First we compare our heuristic with the optimal solution obtained by solving the ILP model in CPLEX. Figure 4.4 shows the profit resulting from dynamic slicing in the six consecutive snapshots for the optimal and for the proposed heuristic solution. We observe that our heuristic leads to very

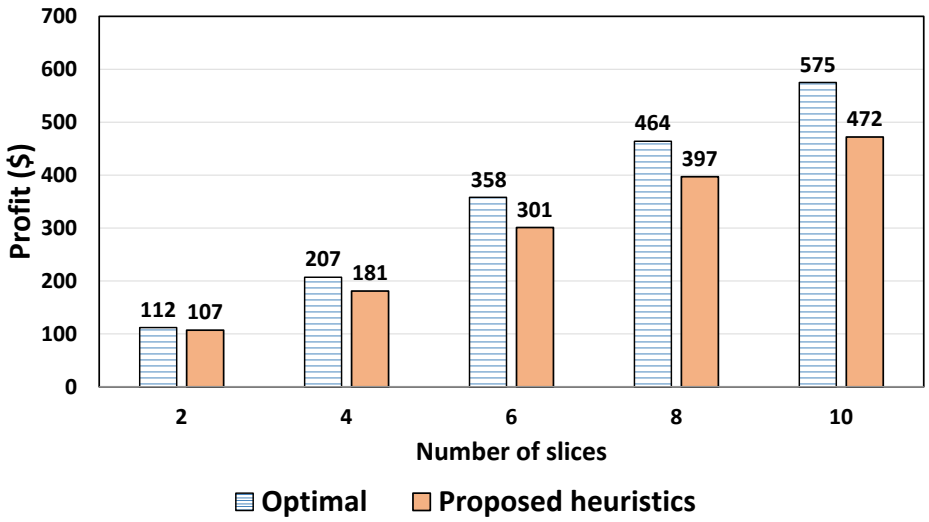


Figure 4.4: Profit (in $) of the optimal vs. the heuristic results.

high quality results in all cases, and reaches near optimal (with a maximum gap of 18%) between the heuristic and the optimal results.

We used a heuristic [91] from the literature, Maximizing the Profit of SP (MPSP) as a baseline from which to compare our dynamic slicing heuristic. The authors model profit and introduce a pricing policy to maximize it for infrastructure providers using 5G dynamic slicing. We compare the expenditures, revenues and profits of MPSP against our heuristic in Fig. 4.5,4.6 and 4.7, respectively, when traffic spikes and the number of admitted slices increases. The results confirm that our heuristic outperforms MPSP (by 33%), and that substantial gains for the InPs can be achieved by using the proposed heuristic. Our heuristic achieves such a high performance because it reconsiders its past decisions and migrates the functions or reassigns them to achieve a better profit. These changes become more significant when more slice requests are admitted to the system, as the number of possible migrations or reassignments increases.

We analyze the impact of different QoS violation penalties ($C^t_{rea,v}$) on the profitability of infrastructure providers in network slicing. In Fig. 4.8, we plot the profits for different numbers of slices for the heuristic solution. Focusing on our objective function components, migration costs and reassignment costs, we can say that different values of QoS violation penalties can have a varying impact on the number of required migrations, and eventually on the amount of the profit. The

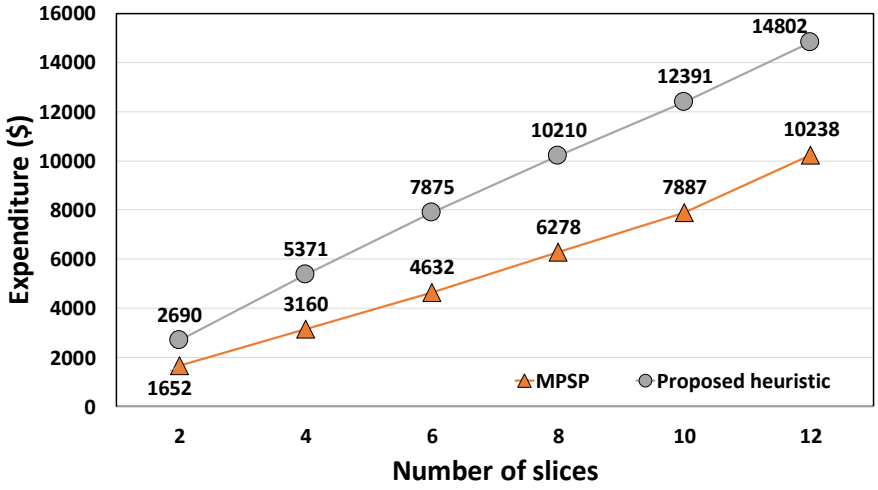


Figure 4.5: Expenditure (in $) vs. number of slices.

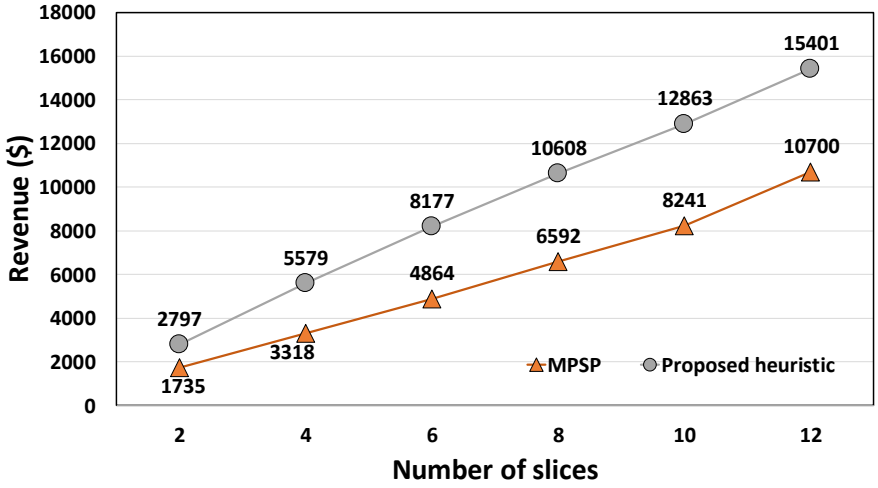


Figure 4.6: Revenue (in $) vs. number of slices.

lower the penalty, the greater the preference for a placement that maximizes profit by incorporating a higher number of migrations. A dynamic slicing policy for the infrastructure provider that considers a penalty for QoS violation will discourage a provider from incorporating a high number of migrations.

Table 4.3 shows the computational time of MPSP, our heuristic and the optimal solution. Note that we could not obtain optimal results for the 12 slices due to the long run time. We can observe from Table 4.3 that the execution times of our proposed heuristic are not only much shorter than those of the CPLEX model, but they also grow linearly with respect to the number of slices. We also

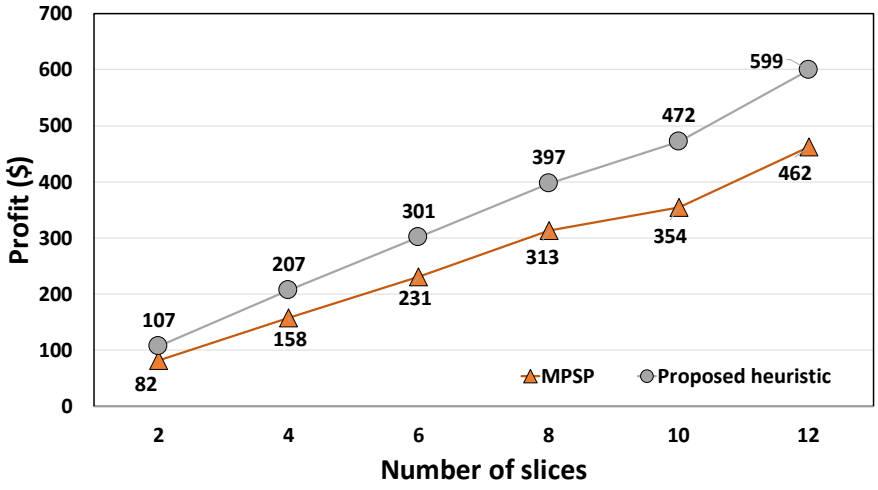


Figure 4.7: Profit (in $) vs. number of slices.

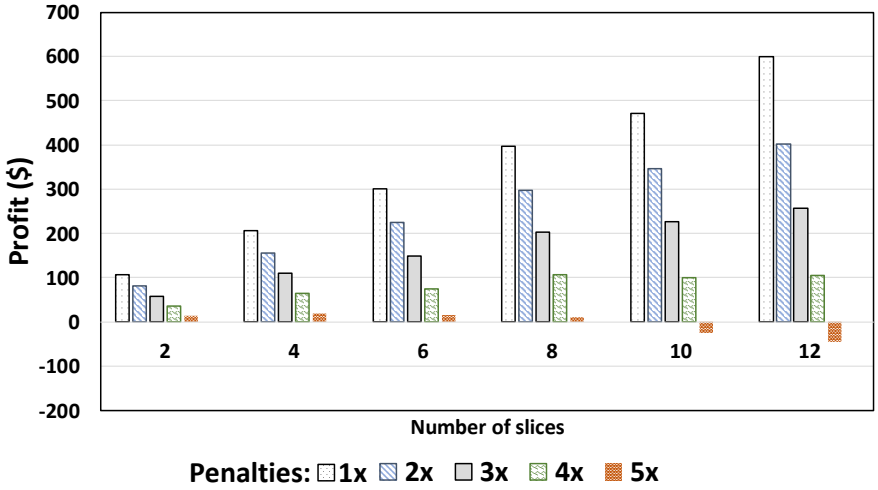


Figure 4.8: Profit (in $) vs. number of slices for different values of reassignment penalties.

note that our heuristic computational time is very close to that of the MPSP, as our heuristic adds a couple of milliseconds to migrate and reassign the placements of VNFs to realize near-optimal profit. These few milliseconds allow our heuristic to outperform the MPSP execution times in cases that require fewer migrations or reassignments (6 slices, 8 slices), and to perform very closely in other cases.

Table 4.3: Execution Times.

	MPSP	Proposed Heuristic	Optimal
2 slices	0.24 s	0.37 s	240 s
4 slices	0.45 s	0.51 s	350 s
6 slices	1.27 s	0.95 s	3480 s
8 slices	1.30 s	1.28 s	10090 s
10 slices	1.18 s	1.56 s	>32100 s
12 slices	1.63 s	1.83 s	——

4.6. Conclusion

This chapter proposed a complete framework targeting the creation of two types of slices, composed upon a content provider's request and which are placed over a physical infrastructure. Leveraging ICNs in-network storage advantages, our solution is tailored to a VNF placement context where ICN-based and traditional IP slices are dynamically created over a physical infrastructure for content delivery. We presented an Integer Linear Programming model and propose a cost-efficient dynamic slicing algorithm. The objective is to maximize the profit of infrastructure providers while meeting the different QoS requirements for different parts of each slice. In addition to the impact of profit maximization, the impacts of QoS violation and reassignment penalties on the infrastructure provider are also analyzed. The results are compared to the state of the art and an optimal solution, and show that the proposed algorithm is very promising, offering a near-optimal solution obtained in a very short time.

Chapter 5

An Architecture for Provisioning In-Network Computing Enabled Slices [1]

5.1. Introduction

Recent advancements in networking and computation have led to a significant increase in demand for 360° video streaming. Delivering 360° videos to end-users while providing a high user Quality of Experience (QoE) is very challenging. Changes to the viewport made by the user can cause motion sickness and make the immersive experience very undesirable when latency is involved. Extremely low latency is one of the critical requirements for an immersive experience. Moreover, 360° videos are high bandwidth- and computing resource-consuming applications that deal with an enormous amount of data (i.e., 25 Mbps of bandwidth and a 40 ms latency for early-stage or current 360° video streaming). These requirements make the provisioning of INC-enabled slices for 360° video streaming extremely challenging [92].

In-network computing is expected to provide lower latency, lower network traffic, and higher throughput. Therefore, the delivery of 360° video and immersive services can benefit from in-network computing. For instance, moving transcoding and head movement prediction algorithms

[1] This chapter is based on: Rayani, Marsa, et al. "An Architecture for Provisioning In-Network Computing Enabled Slices for 360° Video Streaming Services in Next Generation Networks." submitted to IEEE Communication Magazine, 2022.

to the network could reduce latency and bandwidth consumption. Moreover, running advanced compression algorithms such as context-based predictive algorithms for lossless compression, pre-fetching, and pre-caching in the network can also reduce bandwidth utilization for 360° video streaming.

This chapter proposes and validates an architecture for provisioning INC-enabled slices for 360° video streaming services in next-generation networks. We assume the next-generation networks considered in this chapter are SDN- and P4-enabled.

The second section presents the overall architecture, followed by the simulations, their results and our evaluation. The proof-of-concept prototype and measurements are presented in the fourth section. Finally, we summarize this provisioning architecture in the last section.

5.2. Overall Architecture

Figure 5.1 depicts the proposed architecture. From the business model perspective, the 360° video service providers rely on the slices offered by the infrastructure provider. The key novelty of the architecture is that the slices are INC-enabled. The architecture comprises three layers (the Slice Embedding Layer, the Adaptation Layer, and the Physical Resource Layer) and a repository. The architectural modules and procedures are presented next.

5.2.1 Architectural Modules

There are three modules in the slice embedding layer (sub-layer (a)). Together, they form the Slice Engine: Slice Embedding Service, Node Resource Service, and Telemetry Service. The Slice Embedding Service module is the most critical architecture module accessible via a REST API. The Representational State Transfer (REST) architectural style is used to design the interfaces between the layers. REST provides a uniform and lightweight interface based on the existing web technologies. The Slice Embedding Service is in charge of the slice embedding procedure. This module relies on the Node Resource Service and Telemetry modules to get the most updated information

about the infrastructure nodes and links, respectively. We aim at a heterogeneous environment, so we need an adapter layer. In the middle, the Adaptation layer (sub-layer (b)) interacts with heterogeneous physical resources and offers a homogeneous interface to the Embedding layer. The Physical Infrastructure Provider in the bottom sub-layer (c) offers physical resources. This is an SDN network with P4-enabled switches. There is also a Repository to which the Embedding layer has access and which contains the P4 programs.

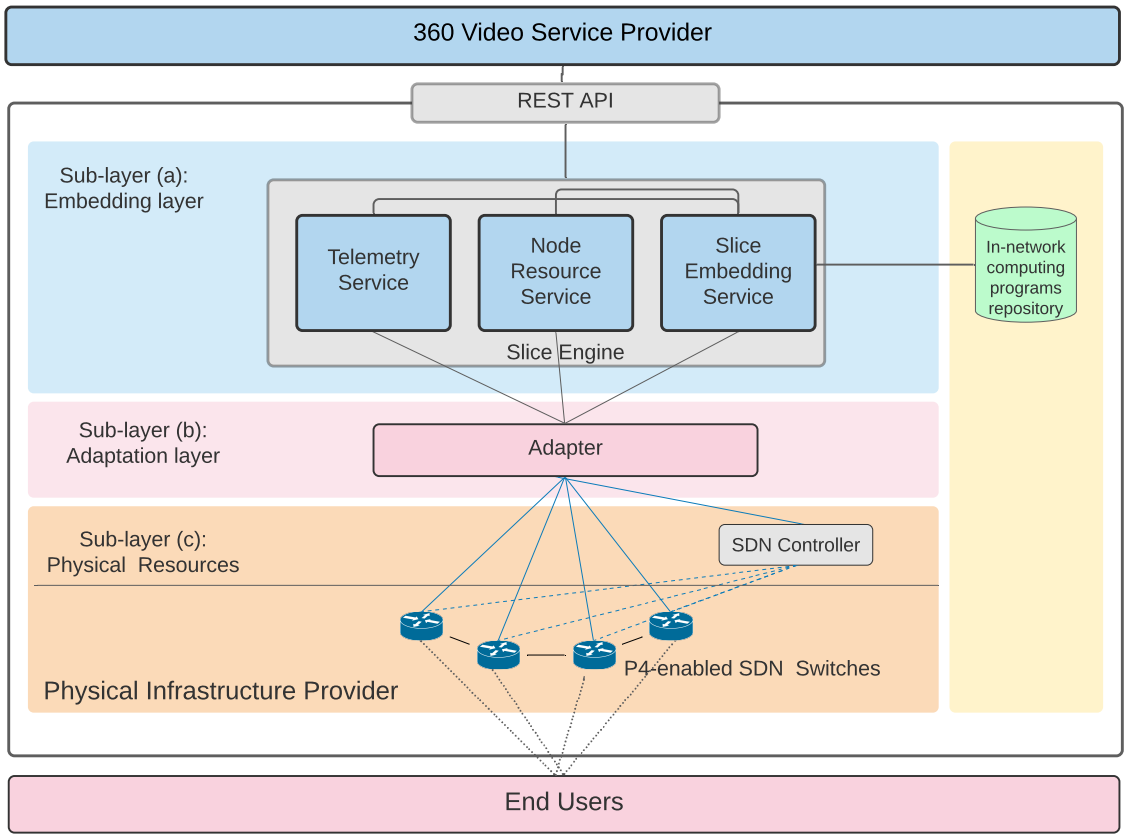


Figure 5.1: Proposed architecture.

5.2.2 Procedures

There are essentially two procedures: a) slice embedding and b) in-network processing enabling.

A) Slice Embedding Procedures

The 360° video service provider sends a request for a slice to the Slice Embedding Service. The Slicing procedure starts upon receiving the slice request. This request consists of the routers and their links, with their CPU and bandwidth requirements, respectively. Next, the Slice Embedding Service requests the CPU and links' statistics from the Node Resource Service and the Telemetry Service, respectively. The Embedding algorithm then decides to embed the slice based on the availability of infrastructure resources prior to sending the notification of the slice creation to the Slice Provider. Several algorithms in the literature can be used to input the slice characteristics and create the slice [93]. SDN controller slicing modules, such as the ONOS, can also implement their slice embedding modules i.e., VPLS.

B) In-network Computing Enabling Procedure

The Slice Embedding Service runs an algorithm to choose a P4 program from the Repository, such as a transcoder. The Slice Embedding Service then runs another algorithm to decide the placement of the P4 program on the infrastructure switches. The algorithm inputs are a set of components implemented as P4 programs, i.e., different types of transcoders, compressors, and overlay ads on the video. Other inputs include the set of infrastructure nodes and end-users with different requirements. The algorithms then decide to place the components on the infrastructure nodes based on objectives such as improving the QoS, the cost, or reducing the network load. To the best of our knowledge, no such algorithm exists today because the problem of in-network enabling of slices has not yet been considered in the literature. In our prototype, we have implemented a straightforward algorithm (described later). Note that all the Slice Embedding requests go through the Adapter, thus covering heterogeneous types of physical infrastructure.

5.3. Simulation

The goal of these simulations is first to compare INC-enabled slices vs. traditional slices, and second, to investigate the gain depending on the locations of INC in the network.

5.3.1 Simulation Scenario

For the simulation, we used a Mininet emulator to set up a P4-enabled SDN infrastructure, which is the Physical Resources layer of the proposed architecture sub-layer (c). Mininet provides an emulator to build programmable SDN infrastructures supporting realistic user traffic at scale. We ran our experiments over a real-world topology that is part of internet topologies in the real world, taken from the Topology Zoo. The simulation topology is illustrated in Fig. 5.2. The simulations were run on a machine with dual 2X8-Core 2.50GHz Intel Xeon CPU E5-2450v2 and 40GB of memory. The network includes P4-enabled switches with an equal CPU and a link's bandwidth of 10 Mbps. A 360° video is streamed from the server, containing 3600 frames. We placed the transcoder in specific slice switches to run and compare different scenarios. We used a P4 link monitoring program that enables the host to monitor the network's links' utilization for the performance evaluation. This monitoring program uses a probe packet processed to pick up the egress link utilization at each hop.

We consider and measure the following metrics: (i) Average latency and (ii) Bandwidth utilization, the most fundamental QoS parameters for video streaming. The proposed architecture is evaluated based on two different test cases. Test case 1 focuses on 360° video streaming performed with one client and one server when only the client has the transcoding capability. Test case 2 focuses on transcoding inside the network. In test case 2, to show the impacts of changing the placement of INC, we vary the transcoder placement and evaluate the latency and bandwidth utilization. We run each simulation ten times to guarantee the precision of the results.

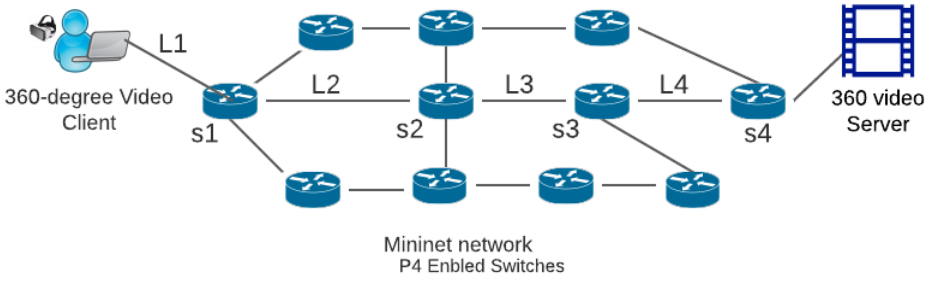


Figure 5.2: Scenario and topology.

5.3.2 Simulation Results

Here we elaborate on the role of provisioning INC-enabled slices and the locations of the INC in supporting 360° video streaming to achieve lower latency and traffic load. Fig. 5.3 depicts the average end-to-end latency for various transcoder placement scenarios, starting from the client to different switches inside the network. The transcoder location starts from the client and moves to s1, s2, s3, and s4, each closer to the server. The result confirms that transcoding inside the INC-enabled slices can achieve lower average latency. Fig. 5.3 also verifies that placing the transcoder inside the network closer to the server reduces the average latency. The average target latency for an immersive experience (such as 360° video streaming) is 10-100 ms [94]; thus, our latency obtained by integrating in-network computing does not exceed this target and thus satisfies the QoS requirements.

Next, we investigate the network load by providing valuable insights into the bandwidth utilization for different scenarios in Fig. 5.4. For the scenario of when the transcoder is placed on the client, we observe that there is a high bandwidth utilization in all links, resulting in high network load. However, the figure confirms that having a P4 transcoder inside the network is beneficial in lowering bandwidth utilization. We can conclude from this figure that in-network computing effectively decreases the network load by reducing bandwidth utilization by 96% for the best placement of the P4 transcoder. Another recommended network SLA for video streaming is to ensure a frame loss below 0.5%, which our results met.

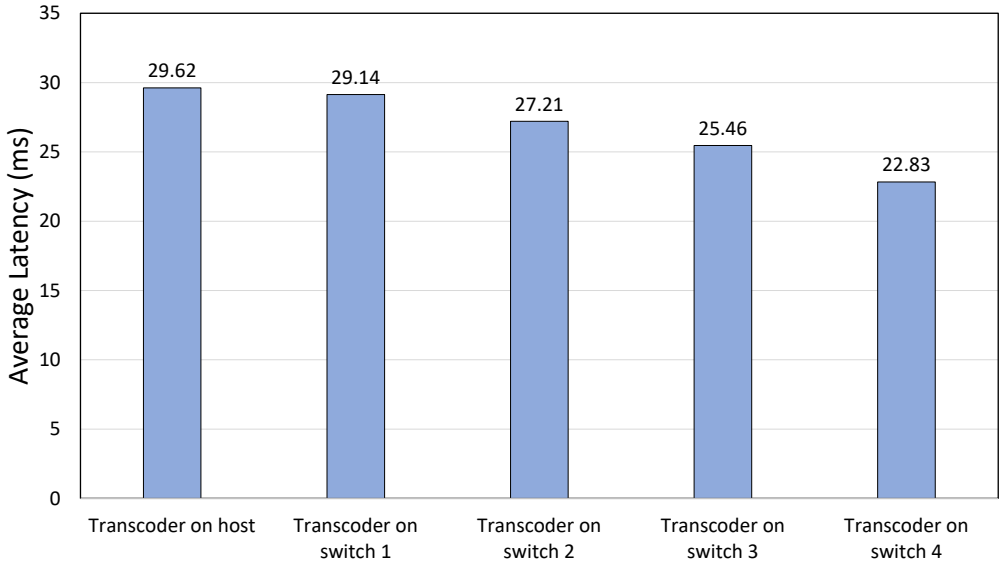


Figure 5.3: Average latency for different scenarios.

5.4. Proof of Concept Prototype

The proof of concept prototype and measurements are presented to show the feasibility of the proposed architecture and evaluate its performance . The goal of the prototype implementation and measurements is to compare the 360° video streaming on INC-enabled slices vs. traditional slices. We also emphasize the advantages of varying the placement of INC inside the network.

5.4.1 Prototype Description

We implement a slicing program to embed the requested slice using Python. There are two different approaches to add extensions to P4 to execute complex operations that P4 does not support: i) using native primitives and ii) using external instances. We have selected the external instances approach, as no modifications are required to the P4 syntax, semantics, and P4 front-end compiler. Externs also reduce modification risks in a switch pipeline, since they are implemented outside the switch core. The transcoder implemented in this work is the first P4 extern transcoder that compiles on the BMV2 switches of the programmable data plane of the SDN network. The P4 transcoder extern is stored in the Repository and will be placed on the selected BMV2 switches using the algorithm inside the Slice Engine.

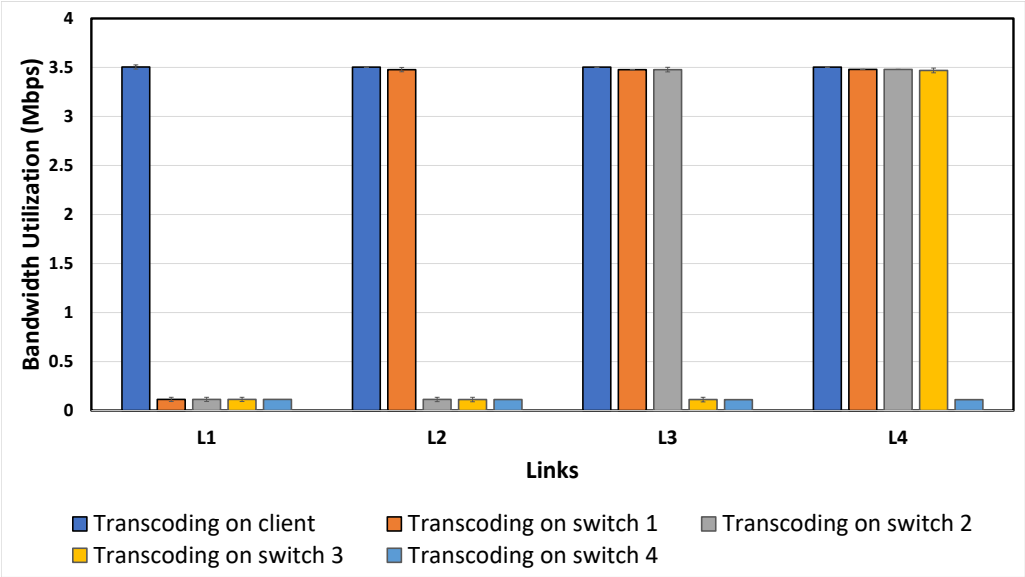


Figure 5.4: Bandwidth utilization for different scenarios.

Fig. 5.5 presents the prototype architecture to implement a simplified version of INC-enabled slice provisioning for the use case of 360° video streaming. We used an SDN network consisting of BMV2 P4-enabled switches for the infrastructure, and a Bitmovin video player for the video streaming client. The Slice Embedding Service was implemented using Python. The client-server web application interacts with Mininet through a REST API. A Node.js-based module is used for the adapter layer to ensure all the heterogeneous infrastructures can be connected to our prototype. Finally, the Repository is implemented using MongoDB and includes a P4 transcoder.

The 360° streaming application was implemented in a prototype. First, a slice request is received using a web browser. Next, the slice embedding module is used to embed the slice based on the available infrastructure resources. The prototype then uses a simplified random selection algorithm to place the transcoder on a switch of a created slice. Finally, the client connected to the slice starts to stream a 360° video.

5.4.2 Performance Metrics

The video streaming delay and slice provisioning delay were used as performance metrics to evaluate the proposed architecture's performance and compare traditional slices vs. INC-enabled

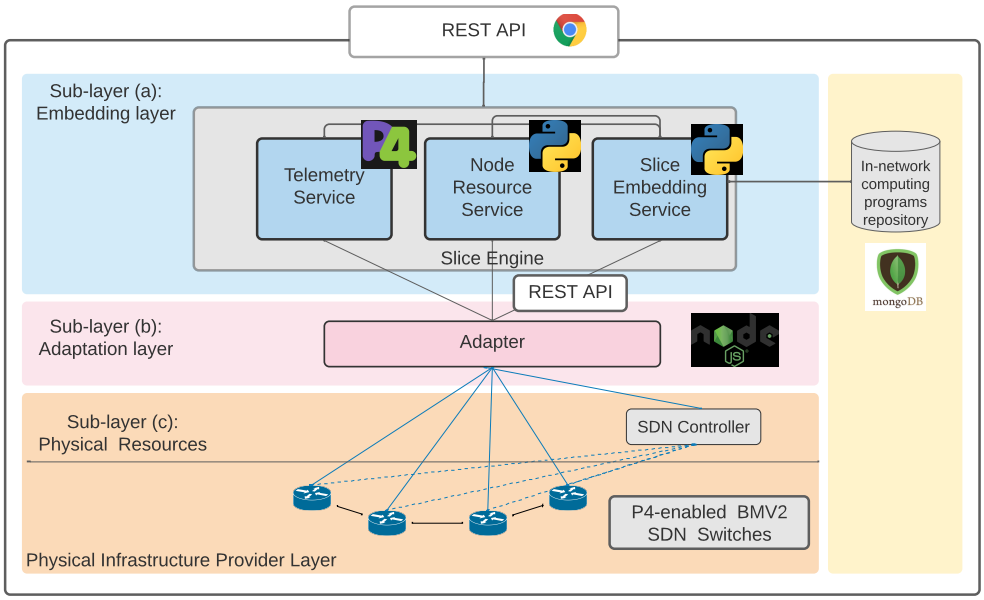


Figure 5.5: Prototype architecture.

slices.

Video Streaming Delay: Defined as the time when the 360° video streaming starts over a created slice and the time the acknowledgment message of streaming completion is received. We analyze this metric in different test cases to show the advantage of INC when we have the P4 transcoder inside the network.

Slice Provisioning Delay: Measured from the time a request is sent by the web application for the provisioning of slices, to the time the acknowledgement message of the slices being provisioned is received. This delay includes the time taken for processing the application's request, uploading the extern package of the transcoder on the selected switches, and creating the slices.

5.4.3 Measurements and Results

This subsection presents and evaluates the performance and results based on the specified metric. Figure 5.6 shows the video streaming delay for traditional slices versus INC-enabled slices. It can be observed that the delay of video streaming significantly decreases in INC-enabled slices that have the transcoder inside the network. We obtained the delay over INC-enabled slices in different

scenarios where the placement of the INC varies. For instance, when the transcoder is placed over the closest switch to the server, we have nearly 30 times less delay of the video streaming compared to a traditional slice.

Figure 5.7 shows the slice provisioning delay to compare the effect of in-network computing in different scenarios. It can be observed that for all scenarios with in-network computing, the time taken to create a slice on top of the infrastructure and place the transcoder inside the network is very close to the time taken to provision the slice when the transcoder is at the client-side. On comparing the scenarios with and without in-network computing, it can be seen that the slight difference between the obtained delay is the time taken to select the switch inside the network and compile the transcoder package on top of it. Provisioning the slices with a transcoder inside the network takes nearly 1% more time than the case of having a transcoder at the client.

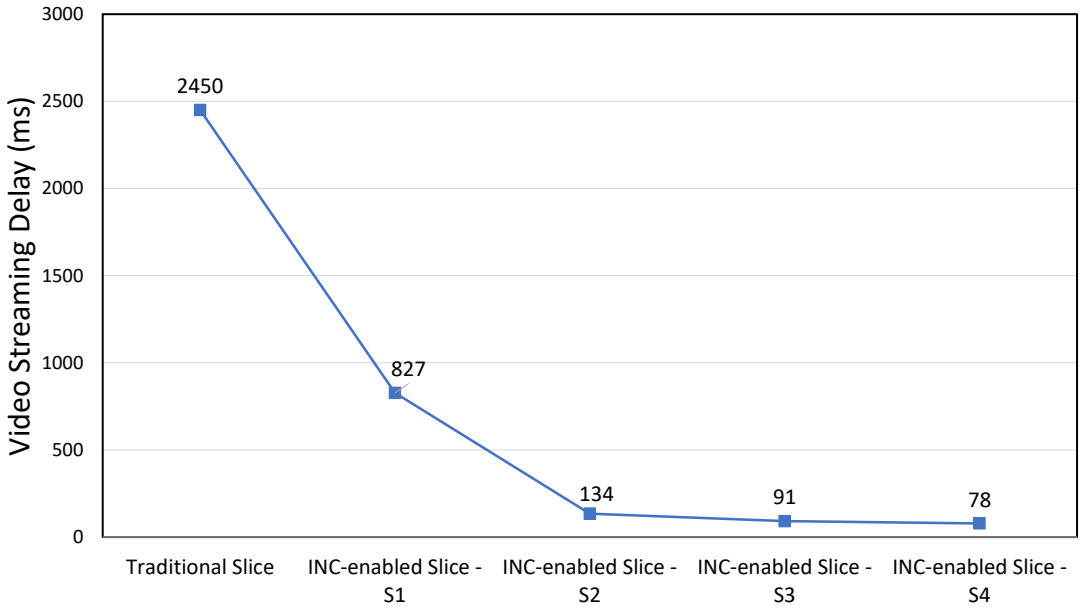


Figure 5.6: Video streaming delay for traditional slice vs. INC-enabled slices in different scenarios.

5.5. Conclusion

In this chapter we proposed a novel architecture for provisioning INC-enabled slices for 360° video streaming in next-generation networks. We used P4-enabled infrastructure to benefit from the

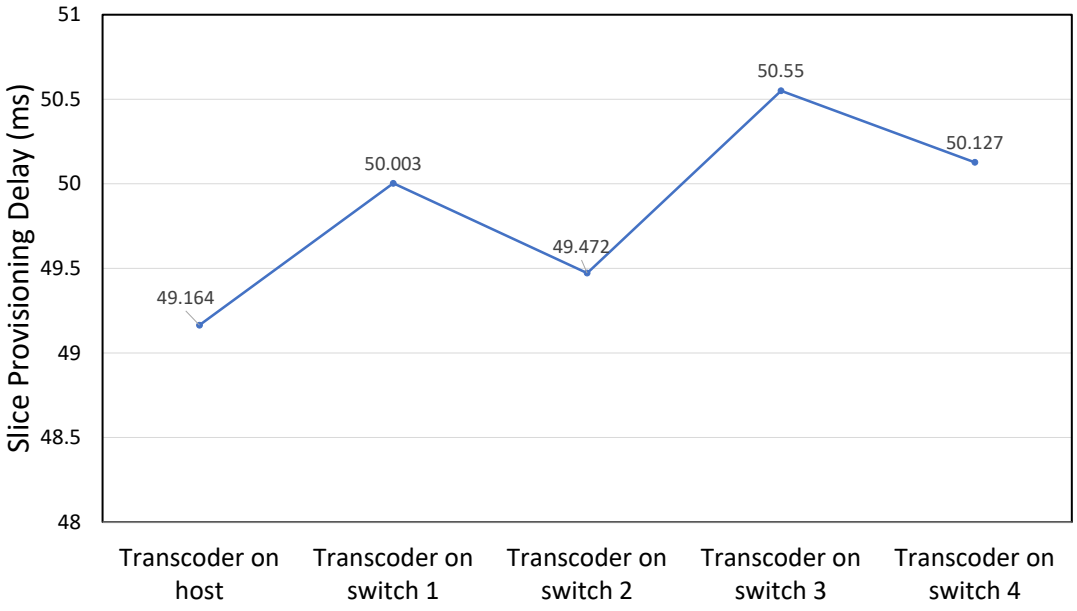


Figure 5.7: Slice provisioning delay for different scenarios.

P4's main characteristics such as agility, top-down design, visibility, less complexity, and enhanced performance. We ran extensive simulations for different scenarios to evaluate the architecture using different performance metrics such as average delay and bandwidth utilization. Finally, a proof of concept prototype was built and the architecture's feasibility was demonstrated by assessing its performance. Our results confirmed that INC effectively decreases the network load by reducing bandwidth utilization (a 96 % decrease for the best placement of the transcoder) and lower average latency. The proof of concept prototype measurements show that the gains in terms of video streaming latency could go up to 30 times.

Chapter 6

Conclusions and Future Work

6.1. Conclusions

The tremendous growth of content over the internet brings new challenges such as scalability and efficiency in content delivery that traditional CDNs are less and less able to meet. In this Ph.D. thesis, we approached these challenges in two complementary ways: architecturally and algorithmically. For the algorithmic contribution, we addressed the accuracy and stability issues in rate adaptation in infrastructures with in-network storage in chapter 3. The proposed algorithm enables more accurate and stable rate adaptation streaming over an enhanced infrastructure with in-network storage such as ICNs.

Another challenge is that the extensive deployment of infrastructures with in-network storage is very costly for infrastructure providers, as it requires a significant investment in installing routers with in-network storage.. Network Slicing can tackle this issue by enabling virtual networks' to co-exist with different characteristics on top of the same physical network. An algorithm for implementing in-network storage (a key feature of future networks) in IP settings for CDNs is proposed in chapter 4. The objective of the proposed algorithm is to maximize the profit of infrastructure providers while meeting the different QoS requirements for different parts of each slice. In addition to the impact of profit maximization, the effects of QoS violation and reassignment penalties on the

infrastructure provider are also analyzed. To enhance infrastructures with in-network computing for CDNs, we proposed an architecture for provisioning INC-enabled slices for 360° video streaming in chapter 5. We considered a P4-enabled SDN network as a physical infrastructure. A high-level description of the proposed architecture with its corresponding procedures is presented. The proposed architecture enables the provisioning of INC-enabled slices for CDNs. The first proposed P4 transcoder and using it in a network are another advantage of this architecture..

6.2. Future work

This thesis presented architectures and algorithms to improve content delivery in future networks. Several research directions remain for future advances in this area.

6.2.1 Rate Adaptation Streaming Algorithm

In a future work, we will further improve the accuracy of our proposed rate estimation algorithm by considering collaborative edge servers instead of single edge servers, for example. Improvements can also be achieved by incorporating machine learning approaches to the algorithm proposed in chapter 3. According to [95], introducing the concept of reinforcement learning at the client side allows the DASH parameters to be changed on the fly. In addition, more accurate rate estimation can be achieved by predicting the bandwidth characteristics, especially in networks with in-network storage. Taking into account information gathered from network-assisted DASH in decision-making processes can improve the accuracy and stability of the rate adaptation. Most of the proposed solutions in the literature are for DASH over HTTP and they do not consider the in-network storage challenge for adaptive streaming.

6.2.2 In-network Storage Implementation

An interesting future research direction for the algorithm proposed in chapter 4 would be to exploit machine learning to solve the problem of in-network storage implementation for CDNs.

The authors of [96] have shown that a DRL-based approach can outperform Q-learning in slicing networks with in-network storage resources (e.g.,in ICNs). DRL algorithms also work well where real datasets are available. There are some important aspects to consider when choosing an appropriate ML algorithm for the problem at hand. The definition and the design of the reward function in reinforcement learning are especially important. In addition, it is critical to investigate the effect of the slow convergence issue of DRL algorithms in this problem. Another research direction is to explore the advantages of a DRL-based dynamic slicing mechanism from the economic aspect, and to design a DRL algorithm that can learn with a good convergence rate using real-world datasets to achieve satisfactory results in increasing the profit of infrastructure providers [97].

6.2.3 Provisioning In-Network Computing Enabled Slices

Although the architecture proposed in chapter 5 aims at handling heterogeneity through the adaptation layer, the environment we used in the prototype is relatively homogeneous. Therefore, the first item for future work will be to use a heterogeneous environment with routers from different manufacturers. This will help in the design of a full-fledged adaptation layer. This paper has focused on the specific case of 360° video streaming. However, our vision is the provisioning of INC-enabled slices for a much wider range of services. Other use cases will be considered in the future. Furthermore, we will also consider extending P4 packages through new primitives.

Another interesting future research direction is to explore the algorithms for the placements of P4 packages over INC-enabled physical infrastructure. The problem is very challenging as there are different applications with P4 packages. Some of these P4 packages might be shared by several applications while others may not. Placing these packages on different switches to meet objectives such as QoS and cost may be compromising and will require sophisticated algorithms. Another future research avenue is the management layer of the architecture. While slice management is well covered in the state of the art, the slices that are considered are usually not INC-enabled.

Bibliography

[1] M. Rayani, R. H. Glitho, and H. Elbiaze, "Etsi multi-access edge computing for dynamic adaptive streaming in information centric networks," in *Globecom 2020-2020 IEEE global communications conference*. IEEE, 2020, pp. 1–6.

[2] M. Rayani, D. Naboulsi, R. Glitho, and H. Elbiaze, "Slicing virtualized epc-based 5g core network for content delivery," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 00 726–00 729.

[3] M. Rayani, A. Ebrahimzadeh, R. H. Glitho, and H. Elbiaze, "Ensuring profit and qos when dynamically embedding delay-constrained icn and ip slices for content delivery," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 2, pp. 769–782, 2021.

[4] M. Rayani, Z. Ait Hmitti, F. Aghaaliakbari, M. Sepideh, V. Maleki Raee, R. H. Glitho, H. Elbiaze, and A. Wessam, "An architecture for provisioning in-network computing enabled slices for 360° video streaming services in next generation networks," *IEEE Communications Magazine*, vol. Under Review.

[5] G. White and G. Rutz, "Content delivery with content-centric networking," *CableLabs, Strategy & Innovation*, pp. 1–26, 2016.

[6] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "The collaboration for content delivery and network infrastructures: A survey," *IEEE Access*, vol. 5, pp. 18 088–18 106, 2017.

[7] C. Westphal, S. Lederer, D. Posch, C. Timmerer, A. Azgin, W. Liu, C. Mueller, A. Detti,

D. Corujo, J. Wang *et al.*, "Adaptive video streaming over information-centric networking (icn)," Tech. Rep., 2016.

[8] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[9] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5g: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.

[10] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[11] J. Esch, "Prolog to:" software-defined networking: a comprehensive survey"," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10–13, 2014.

[12] B. Zolfaghari, G. Srivastava, S. Roy, H. R. Nemati, F. Afghah, T. Koshiba, A. Razi, K. Bibak, P. Mitra, and B. K. Rai, "Content delivery networks: State of the art, trends, and future roadmap," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–34, 2020.

[13] A.-M. K. Pathan, R. Buyya *et al.*, "A taxonomy and survey of content delivery networks," *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, vol. 4, no. 2007, p. 70, 2007.

[14] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on replica server placement algorithms for content delivery networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1002–1026, 2016.

[15] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1842–1866, 2017.

[16] T. Stockhammer *et al.*, "Dynamic adaptive streaming over http-design principles and standards," *Qualcomm Incorporated*, pp. 1–3, 2011.

[17] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for http live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, 2014.

[18] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 internet measurement conference*, 2012, pp. 225–238.

[19] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi, "Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip," *IEEE transactions on Multimedia*, vol. 19, no. 10, pp. 2166–2181, 2017.

[20] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360 video streaming: solutions, challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2801–2838, 2020.

[21] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE communications surveys & tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[22] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: Strategies, challenges, and future research directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1443–1474, 2017.

[23] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[24] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[25] M. Mosko, I. Solis, and C. Wood, "Content-centric networking (ccnx) semantics," Tech. Rep., 2019.

[26] G. White and G. Rutz, "Content delivery with content-centric networking," *CableLabs, Strategy & Innovation*, pp. 1–26, 2016.

[27] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "The collaboration for content delivery and network infrastructures: A survey," *IEEE Access*, vol. 5, pp. 18 088–18 106, 2017.

[28] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 147–158, 2013.

[29] I. Kunze, K. Wehrle, D. Trossen, and M.-J. Montpetit, "Use cases for in-network computing," *IETF, Internet-Draft*, 2021.

[30] D. R. Ports and J. Nelson, "When should the network be the computer?" in *Proceedings of the Workshop on Hot Topics in Operating Systems*, 2019, pp. 209–215.

[31] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, "In-network computation is a dumb idea whose time has come," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 2017, pp. 150–156.

[32] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[33] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[34] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5g: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.

[35] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5g communications: Slicing the lte network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 146–154, 2017.

[36] E. Thomas, M. O. Van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing mpeg dash performance via server and network assistance," *SMPTE Motion Imaging Journal*, vol. 126, no. 1, pp. 22–27, 2017.

[37] R. Jmal, G. Simon, and L. Chaari, "Network-assisted strategy for dash over ccn," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 13–18.

[38] R. Jmal and L. Chaari Fourati, "Assisted dash-aware networking over sdn–ccn architecture," *Photonic Network Communications*, vol. 38, no. 1, pp. 37–50, 2019.

[39] R. Grandl, K. Su, and C. Westphal, "On the interaction of adaptive video streaming with content-centric networking," in *2013 20th International Packet Video Workshop*. IEEE, 2013, pp. 1–8.

[40] P. Zhang, H. Yao, and Y. Liu, "Virtual network embedding based on computing, network, and storage resource constraints," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3298–3304, 2017.

[41] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "Efficient caching resource allocation for network slicing in 5G core network," *IET Communications*, vol. 11, no. 18, pp. 2792–2799, 2017.

[42] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "Joint network slicing and mobile edge computing in 5G networks," in *Proc. IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[43] S. Agarwal, F. Malandrino, C.-F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2018, pp. 1943–1951.

[44] H. Jin, H. Lu, Y. Jin, and C. Zhao, "IVCN: Information-centric network slicing optimization based on NFV in fog-enabled RAN," *IEEE Access*, vol. 7, pp. 69 667–69 686, 2019.

[45] J. Liu, B. Zhao, M. Shao, Q. Yang, and G. Simon, "Provisioning optimization for determining and embedding 5G end-to-end information centric network slice," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 273–285, 2021.

[46] R. Ravindran, A. Chakraborti, S. O. Amin, A. Azgin, and G. Wang, "5G-ICN: Delivering ICN services over 5G using network slicing," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 101–107, 2017.

[47] G. Sun, H. Al-Ward, G. O. Boateng, and W. Jiang, "Content-aware caching in SDN-enabled virtualized wireless D2D networks to reduce visiting latency," in *Proc. IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2018, pp. 149–150.

[48] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE network*, vol. 29, no. 3, pp. 68–74, 2015.

[49] M. R. Raza, M. Fiorani, A. Rostami, P. Öhlen, L. Wosinska, and P. Monti, "Dynamic slicing approach for multi-tenant 5G transport networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 1, pp. A77–A90, 2018.

[50] Y. L. Lee, J. Loo, T. C. Chuah, and L.-C. Wang, "Dynamic network slicing for multitenant heterogeneous cloud radio access networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146–2161, 2018.

[51] T. V. K. Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, "Resource allocation with admission control for GBR and delay QoS in 5G network slices," in *Proc. IEEE International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2020, pp. 213–220.

[52] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[53] M. Vincenzi, E. Lopez-Aguilera, and E. Garcia-Villegas, "Maximizing infrastructure providers' revenue through network slicing in 5G," *IEEE Access*, vol. 7, pp. 128 283–128 297, 2019.

[54] B. Han, S. Tayade, and H. D. Schotten, "Modeling profit of sliced 5G networks for advanced network resource management and slice implementation," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 576–581.

[55] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5G with network resource pricing," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2017, pp. 1–6.

[56] S. O. Oladejo and O. E. Falowo, "Profit-aware resource allocation for 5g sliced networks," in *Proc. IEEE European Conference on Networks and Communications (EuCNC)*, 2018, pp. 43–9.

[57] S. Chen and B. Krishnamachari, "Differential pricing of 5g network slices for heterogeneous customers," in *Proc. IEEE Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, pp. 0705–0711.

[58] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, "Profit-based radio access network slicing for multi-tenant 5G networks," in *IEEE Proc. European Conference on Networks and Communications (EuCNC)*, 2019, pp. 603–608.

[59] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[60] G. Sun, H. Al-Ward, G. O. Boateng, and G. Liu, "Autonomous cache resource slicing and content placement at virtualized mobile edge network," *IEEE Access*, vol. 7, pp. 84 727–84 743, 2019.

[61] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, and X. Costa-Pérez, "A machine learning approach to 5G infrastructure market optimization," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 498–512, 2019.

[62] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," vol. 20, no. 7, pp. 4585–4600, 2021.

[63] S. Vaucher, N. Yazdani, P. Felber, D. E. Lucani, and V. Schiavoni, "Zipline: in-network compression at line speed," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 399–405.

[64] T. Mai, H. Yao, S. Guo, and Y. Liu, "Wia-nr: Ultra-reliable low-latency communication for industrial wireless control networks over unlicensed bands," *IEEE NETWORK*, vol. 35, no. 1, pp. 289–295, 2021.

[65] A. Tagami, K. Ueda, R. Lukita, J. De Benedetto, M. Arumaithurai, G. Rossi, A. Detti, and T. Hasegawa, "Tile-based panoramic live video streaming on icn," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6.

[66] P. H. Isolani, J. Haxhibeqiri, I. Moerman, J. Hoebeke, J. M. Marquez-Barja, L. Z. Granville, and S. Latré, "An sdn-based framework for slice orchestration using in-band network telemetry in ieee 802.11," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 344–346.

[67] R. A. Kirmizioglu and A. M. Tekalp, "Multi-party webrtc services using delay and bandwidth aware sdn-assisted ip multicasting of scalable video over 5g networks," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 1005–1015, 2019.

[68] B. Hayes, Y. Chang, and G. Riley, "Scaling 360-degree adaptive bitrate video delivery over an sdn architecture," in *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2018, pp. 604–608.

[69] R. F. Diorio and V. S. Timóteo, "Multimedia content delivery in openflow sdn: an approach based on a multimedia gateway," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2016, pp. 612–617.

[70] S. Yilmaz, A. M. Tekalp, and B. D. Unluturk, "Video streaming over software defined networks with server load balancing," in *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2015, pp. 722–726.

[71] Y. Dong, L. Song, R. Xie, and W. Zhang, "An elastic system architecture for edge based low latency interactive video applications," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 824–836, 2021.

[72] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Joint optimization of qoe and fairness through network assisted adaptive mobile video streaming," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2017, pp. 1–8.

[73] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2014.

[74] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over http," in *2012 19th international packet video workshop (PV)*. IEEE, 2012, pp. 173–178.

[75] A. Mehrabi and M. Siekkinen, "Edge computing assisted adaptive mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 787–800, 2018.

[76] D. S. Johnson, "The np-completeness column: An ongoing guide," *Journal of Algorithms*, vol. 4, no. 2, pp. 189–203, 1983.

[77] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, "ndnsim: Ndn simulator for ns-3," *University of California, Los Angeles, Tech. Rep*, vol. 4, 2012.

[78] "Big buck bunny movie, http://www.bigbuckbunny.org," (Accessed Jun. 1, 2022).

[79] S. Sesia, I. Toufik, and M. Baker, *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011.

[80] C. Kreuzberger, D. Posch, and H. Hellwagner, "Amust framework-adaptive multimedia streaming simulation framework for ns-3 and ndnsim," 2016.

[81] M. Seufert, F. Wamser, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Youtube qoe on mobile devices: Subjective analysis of classical vs. adaptive video streaming," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2015, pp. 43–48.

[82] G. White and G. Rutz, "Content delivery with content-centric networking," *CableLabs, Strategy & Innovation*, pp. 1–26, 2016.

[83] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[84] E. Amaldi, S. Coniglio, A. M. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Elsevier Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, 2016.

[85] WonderNetwork, "Global ping statistics," wondernetwork." *Available: https://wondernetwork.com/pings*, (Accessed Jun. 1, 2022).

[86] H. Cao, H. Zhu, and L. Yang, "Dynamic embedding and scheduling of service function chains for future SDN/NFV-enabled networks," *IEEE Access*, vol. 7, pp. 39 721–39 730, 2019.

[87] E. I. Administration, "State electricity profiles - energy information administration," *Available: https://www.eia.gov/electricity/state/.*, (Accessed Jun. 1, 2022).

[88] S. Zhang, "An overview of network slicing for 5G," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[89] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "On the placement of VNF managers in large-scale and distributed NFV systems," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 875–889, 2017.

[90] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1130–1143, 2019.

[91] G. Wang, G. Feng, W. Tan, S. Qin, R. Wen, and S. Sun, "Resource allocation for network slices in 5G with network resource pricing," in *Proc. IEEE Global Communications Conference*, 2017, pp. 1–6.

[92] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360 video streaming: Solutions, challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2801–2838, 2020.

[93] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5g telecommunication networks: A survey of principles and models," *IEEE Network*, vol. 33, no. 6, pp. 172–179, 2019.

[94] M. Fiedler, H.-J. Zepernick, and V. Kelkkanen, "Network-induced temporal disturbances in virtual reality applications," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2019, pp. 1–3.

[95] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *2015 IFIP/IEEE*

International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 131–138.

[96] G. Sun, H. Al-Ward, G. O. Boateng, and G. Liu, "Autonomous cache resource slicing and content placement at virtualized mobile edge network," *IEEE Access*, vol. 7, pp. 84 727–84 743, 2019.

[97] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," vol. 20, no. 7, pp. 4585–4600, 2021.