Performance Improvements of In-network Caching in ICN-Based Networks

by

Zhe Zhang, B. Eng., M. Sc.

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
August, 2019

Acknowledgement

Pursing my Ph.D. degree in Canada was an exciting and amazing journey for me, and it started when I wrote my first email to my supervisor, Prof. Chung-Horng Lung. Hence, I would like to begin this section by expressing my deeply heartfelt thanks and gratitude to him, not only for his wholehearted guidance on my life as a student, but also for his profound influence on my values as a human being. From him, I learned how to perform research with rigorousness, and how to treat others with patience. Without his encouragements and supports, I would not have such a wonderful study.

I would also like to thank my co-supervisor, Prof. Marc St-Hilaire, for his valuable comments and guidance. The academic writing skills and research methodologies he taught me helped me during my entire Ph.D. study and will continue to benefit my future work. More importantly, his rigorous attitude to academic is the most valuable treasure that I learned from him.

Special thanks to Prof. Ioannis Lambadaris for offering me the internship opportunity at Ericsson. Besides, his immense knowledge and kind suggestions not only improve the quality of the research but also broaden my mind for my future career.

I would also like to thank the committee members, Prof. Amiya Nayak, and Prof. F. Richard Yu for their helpful and insightful questions and suggestions.

Many thanks also go to my friends Andy Yin, Chenguang Yu, Chris Dydula, Decheng Zhang, Ming Liu, Qiao Lu, Robbin Gao, Rick Luo, Xiaoying Chang, Yanran Li, Yi Lin, Yingzhe Wang, Yundu Cao. With their company, my life here becomes colorful and is full of joy and happiness.

Besides, I would like to thank my Master’s supervisor, Prof. Yuchun Guo, for encouraging and supporting me to pursuit a Ph.D. degree.

I would also like to thank my family members. There is no word to express my gratitude to my family, and in particular, my grandfather, my grandmother, my parents, and my cousin. Their love is and will always be my source of power and courage.

Abstract

The continuous development of networking technologies and smart devices has led Internet traffic, especially for the multimedia content traffic, to increase drastically both in wired and wireless networks. Similarly, the fast developing of wireless networks, such as 5G, has led the Internet of things (IoT) to be growing at an unprecedented pace. However, the traditional host-centric IP Internet is based on host-to-host communications which is not suitable for satisfying the requirements of content delivery. Hence, information-centric networking (ICN), one of the emerging next-generation Internet paradigms, is proposed to overcome these challenges. With the ubiquitous in-network caching, ICN can facilitate content delivery and reduce network delay. Both 5G and IoT can use the concept of ICN to constitute ICN-5G and ICN-IoT networks respectively.

However, the requirements of in-network caching may vary for different networks. This thesis focuses on designing in-network caching approaches for different networks, including pure ICN networks, ICN-5G networks and ICN-IoT networks, from a theoretical perspective and a practical perspective. Both reactive and proactive caching approaches are discussed in this thesis. Moreover, by leveraging the concepts of software-defined networking (SDN) and machine learning (ML), the efficiency of in-network caching can be significantly improved.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

5G	Fifth Generation of Cellular Mobile Communications
ABC	Age-based Cooperative Caching
ARMA	Auto-Regressive and Moving Average
AV	Autonomous Vehicle
BS	Base Station
CDM	Caching Decision Module
CF	Collaborative Filtering
CR	Content Router
DNS	Domain Name System
EPC	Evolved Packet Core
EU	European Union
FIB	Forwarding Information Base
FIFO	First-in First-out
HD	High-Definition
ICN	Information-centric Networking
ILP	Integer Linear Programming
IoT	Internet of Things
IP	Internet Protocol
LAC	Latency-aware Caching
LCC	Lifetime-based Cooperative Caching
LCD	Leave Copy Down
LCE	Leave Copy Everywhere

LFF	Least Fresh First
LFU	Least Frequently Used
LRU	Least Recently Used
MaaS	Mobility as a Service
MCD	Move Copy Down
MF	Matrix Factorization
ML	Machine Learning
MPC	Most Popular Content
MU	Mobile User
NAT	Network address translation
NDN	Named Data Networking
NMF	Non-negative Matrix Factorization
NP-hard	Non-deterministic Polynomial-time Hard
P-CLS	Popularity-driven Caching Location and Searching
PIT	Pending Interest Table
PURSUIT	Publish-Subscribe Internet Technology
QoE	Quality of Experience
RAN	Radio Access Network
RPC	Router Position-based Cooperative Caching
RNN	Recurrent Neural Network
RS	Recommender System
RSU	Road Side Unit
SDN	Software-defined Networking

SVD	Singular Value Decomposition
TE	Traffic Engineering
UFRPM	User Future Ratings Prediction Module
UHD	Ultra-high-definition
UMPM	User Mobility Prediction Module
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
VNI	Visual Networking Index
VoD	Video on Demand

List of Symbols

Symbols for Chapter 3

α	Configurable weight for t_r
β	Configurable weight for l_i
A	Average number of hops
$a_{mn} \in \{0,\ 1\}$	If node *n* retrieves video from node *m*
$a_{mn}(t) \in \{0,\ 1\}$	If node *n* retrieves video from node *m*
$B_n(v)$	The delay that can be reduced by caching video v at node n
C	Caching state set of all nodes
$c_n(v) \in \{0,\ 1\}$	Caching state of node *n* for video *v*
$c_n(v,t)$	Caching state of node *n* at time t
$D(n)$	Reduced video transmission delay by performing caching at node *n*
$D(n,t)$	Reduced video transmission delay by performing caching at node *n* from time 0 to time *t*
$DC(v)$	Total delivery delay for delivering video v from the video server to all users at current time
d_{0n}	The delivery delay from the video server to node n
$d_{mn}(v)$	Video transmission delay for video *v* from node *m* to node *n*
$d_{mn}(v,t)$	Video transmission delay for video *v* from node *m* to node *n* at time *t*
F	Total amount of traffic for all requests transmitted from the video server if no caching is used
G	Gain (the total reduced transmission delay)

$G(t)$	Total reduced video transmission delay (gain) from time 0 to time *t*
h_i	The number of hops needed to deliver video request i
I	The number of interest packets
l_i	The topology level value of router i
L_v	Length of video *v*
N	Set of nodes
$\|N\|$	Total number of nodes
n_{obj}	Objective node
P	Caching decision policy
R	Reduced video server load ratio
$R(n)$	Number of requests can be served from node *n*
Req	Set of requests for the videos
$\|Req\|$	Total number of requests
$req_n(v)$	Total number of requests for video *v* from node *n*
$req_n(v,t)$	Number of requests for video *v* from node *n* at time t
T	Simulation time
Tc	Video cost rank table
Td	Caching decision calculation table
TH	Caching threshold for MPC
t_i	Threshold of router i
t_r	Threshold of root router
V	Set of videos
$\|V\|$	Total number of videos

Z_n	Cache size of node *n*

Symbols for Chapter 4

α	Angle between user moving direction and the BS
$\bar{C}$	The average number of choppy playback
c_i	The number of choppy playback for video i
$\overline{D}$	The average retrieval delay
d	Distance that a user moves in the coverage area of the BS
d_i	The retrieval delay of video i
d_{t0-bs}	Distance between a mobile user and the BS at time $t0$
d_{t1-bs}	Distance between a mobile user and the BS at time $t1$
d_{t1-t0}	Distance moved by a mobile user between time $t0$ and $t1$
H	Handoff indicator
$\overline{M}$	The average miss ratio
$m_i \in \{0,\ 1\}$	If video i is cached at a BS or a CR locally
P_0	The position when user enters the coverage area of a BS
P_1	The position of user at time t_1
P_2	The predicted position when user is about to leave the coverage area of a BS
R_i	Bit rate of video i
$\|Req\|$	Total number of requests
r	Radius of the BS coverage
S_i	Size of video i
x_{bs}	x coordinate at of BS
x_{t0}	x coordinate at time $t0$

x_{t1}	x coordinate at time $t1$
y_{bs}	y coordinate at of BS
y_{t0}	y coordinate at time $t0$
y_{t1}	y coordinate at time $t1$

Symbols for Chapter 5

$\alpha, \beta, \gamma \in \{0,1\}$	Parameters
θ	Configurable weight
σ	Configurable weight
A	Caching policy
a	Path loss factor
$age(i)$	Set of data lifetime for intermediate node *i* in a sliding time window
b	Energy cost of the transmitter amplifier
$C(i,t)$	Caching status of node i at time t
$c_{d_l}(i,t) \in \{0, 1\}$	Decision variable representing if data item d_l is cached at node i at time t
D	Set of data items
$\lvert D \rvert$	Total number of data items
D_j	Euclidean distance between IoT device *j* and the gateway node
d_l	The *l*th data item
e_{awake}	Energy consumed when transferring from the sleep mode to the active mode
e_j	Energy consumed by IoT device *j*
$e_{sensing}$	Energy consumed by IoT devices for sensing one bit
e_t	Energy consumed by a transmitter for transmitting one bit
f	The freshness requirement
$f_e(\,)$	Threshold decision function of the edge nodes

$f_m(\,)$	Threshold decision function of the middle-level nodes
$f_r(\,)$	Threshold decision function of the root nodes
$\overline{Hop}$	Average number of hops
$hop_n(t)$	Number of hops from the content producer to the user at time t
I	Set of intermediate nodes
$\|I\|$	Total number of intermediate nodes
J	Set of IoT devices
n_j	Number of times that IoT device *j* is activated
P	Packet size
$Req(t)$	Requests for the IoT data items at time t
r_{it}	Request rate of node i at time t
$r_n(i,t) \in \{0,\ 1\}$	If the *n*th request can be served from the intermediate nodes at time t
$req_n(t)$	The *n*th request at time t
s	Cache size of node i
T	Total time
T_g	The time that the data item is generated
TH_i	Threshold of node i

Symbols for Chapter 6

λ	A regularization parameter
τ	The time that a video chunk can be played
ξ	Size of a video chunk
A	Caching decision
B	Total benefit that can be achieved by performing proactive caching with the node cache condition
$BR(v_m)$	Bitrate of video v_m
BW_h	Available bandwidth of the link for the h^{th} hop
b_{v_m}	The benefit of caching video v_m
$C(i,t)$	Caching status of node i at time t
$c_{v_m^k}(i,t)$	If video chunk v_m^k is cached at node i at time t
cs_i	Cache size of node i
D	Total distance
$D_{ij}(v_m^k)$	Delivery cost for video chunk v_m^k from node i to node j
$Dist_h$	Physical link length for the h^{th} hop
d_{ij}	End to end delay from node i to node j
$F(x,y)$	The integral of $f(x,y)$
$f(x,y)$	Planned route for the vehicle
H	Set of hops that a video needs to be delivered
$H(i,t)$	Number of requests that can be served from node i at time t
$\overline{Hops}$	Average number of hops
h	The h^{th} hop in H

I	Set of nodes
$K(v_m)$	Number of chunks that a video v_m can be divided
k_l	The last chunk that the AV user watches before arriving node i
M	Total number of videos
$MaxRate$	The maximum rating in video set V
$MaxReq$	The maximum number of requests in video set V
$MaxSize$	The maximum size of video in V
N	Total number of AV users
$nc(u_n, i, v_m)$	Number of video chunks that will be played during the period $t_d(u_n, i)$ and $t_a(u_n, i)$
ns_{v_m}	The normalized size of video v_m
P	User feature matrix
$Pop(v_m)$	Historical popularity of video v_m
$Pred(v_m)$	The normalization of the predicted ratings for video v_m
p_n	A nth row vector
p_{nw}	The nth row wth column element in P
Q	Video feature matrix
Q_h	Queueing delay for the hth hop
q_m^T	A mth column vector
q_{mw}	The mth row wth column element in Q
R	Rating matrix
$\tilde{R}$	The estimated rating matrix
$\overline{Rate(v_m)}$	The average predicted rating of video v_m

$Req(i,t)$	Set of requests for all videos from node i at time t
$\lvert Req(i,t)\rvert$	Total number of requests of node i at time t
$r(u_n, v_m)$	Rating of user u_n on video v_m
$r(\widetilde{u_n, v_m})$	The estimated rating of user u_n on video v_m
$req_{v_m^k}(i,t)$	Total number of requests for the kth chunk of video v_m of node i at time t
s_{v_m}	Size of video v_m
$t_a(u_n, i)$	The arrival time of an AV user u_n at an edge node i
t_b	The time that AV user needs to finish watching the buffered video chunks before fetching new chunks from RSUs
t_c	Current time
$t_d(u_n, i)$	The departure time of an AV user u_n at an edge node i
U	Set of AV users
u_n	AV user n
v_m	Video m
V	Set of videos
$\overrightarrow{v(t)}$	Current velocity vector of an autonomous vehicle
W	Number of features
(x_c, y_c)	Current position of the vehicle
(x_p, y_p)	Predicted position of the vehicle

Chapter 1: Introduction

Information-centric networking (ICN) is proposed as a significant common approach of several future Internet research activities. The ICN architecture leverages in-network caching to improve the efficiency of content distribution, and uses name-based routing to support mobility by nature.

This chapter is divided into four parts. The first part presents a brief background on current Internet shortcomings and several future network scenarios for ICN in-network caching. The second part describes the motivation of the research on ICN in-network caching. The third part presents the contributions of this research. Finally, the last part shows the thesis organization.

1.1 Background

With the tremendous increase of content transmissions, the current Internet traffic has shifted from host-centric to content-centric [1]. According to the Cisco's Visual Networking Index (VNI) report [27], the Internet traffic will increase around threefold from 2016 to 2021, the sum of all forms of video and multimedia traffic will account for 82% of global consumer traffic by 2021. Further, the mobile video will account for 78% of the total mobile traffic by 2021. This trend causes a huge challenge for today's Internet on how to distribute videos efficiently. Moreover, billions of devices will be connected to the Internet over the next five years, which leads the current IP-based Internet to facing tremendous challenges, such as limited expressiveness of IP addressing multicast, complex mobility support and the requirements of energy efficiency for IoT resource-constrained devices.

To address these challenges, many future Internet architectures have been proposed. Among all the proposed architectures, information-centric networking (ICN) [1] is a promising paradigm to facilitate content delivery, reduce retrieval delay and save energy for resource-constrained devices by performing in-network caching. Since ICN supports name-based routing, it decouples contents from the locations, which means that ICN supports mobility by nature. Therefore, by combining ICN and 5G, we can get ICN-5G networks [76]; by combing ICN and IoT, we can get ICN-IoT networks [1]. With the help of emerging technologies, such as software-defined networking (SDN) [45] and machine learning algorithms (e.g., recommender system algorithms [98]), the efficiency of in-network caching can be further improved.

1.1.1 Information-centric Networking

The evolution of social networking, mobile applications, multimedia streaming services, and IoT networks has caused the current IP-based Internet to shift from host-centric to content-centric. However, the current IP-based Internet which was originally designed for host-to-host communications, is a host-centric network. The current host-centric IP-based Internet is inefficient for today's content transmissions, hence ICN has been proposed to provide efficient content transmissions for future networks.

The basic principle of ICN is that content is identified by using a unique and location-independent identifier, so that users can fetch the content by its name instead of its IP address as used in the current Internet. Therefore, ICN can provide native support for scalable and highly efficient content retrieval, support mobility, and overcome the limited expressiveness of IP addressing of IP-based networks. Many projects or architectures have been proposed for ICN, such as the European Union (EU) funded project Publish-Subscribe Internet Technology (PURSUIT) [31], and the United State funded project called Named Data Networking (NDN) [8]. However, NDN is currently the dominant one and it has been widely accepted in the research community.

Routers in ICN route and forward packets based on names, which eliminate three problems caused by addresses in the IP-based Internet: address space exhaustion, NAT traversal, and address management [113]. Since the namespace is unbounded, there is no address exhaustion problem in ICN. There is no NAT traversal problem since NDN does away with addresses, public or private. Finally, address assignment and management is no longer required in local networks. Conventional routing protocols, such as OSPF and BGP, can be adapted to route on name prefixes by treating names as a sequence of opaque components and doing component-wise longest prefix match of a name in an Interest packet against the forwarding information base (FIB) table. The research of the Name-based routing is beyond the scope of this thesis. Although the details of name-based routing are outside the scope of this research, interested reader is referred to [11].

In general, when a node receives an interest packet, it first checks if the requested content has been cached locally. If it is cached, this node will directly return the content back to the user. Otherwise, this node will check if the name of the content is in its pending interest table (PIT). The PIT table stores the interest packets that the node has received but not satisfied yet. If there is a matching entry in the PIT, this node will simply add the incoming interface of this interest packet in the corresponding entry. Once the requested content is available, it will be sent back to users through all the interfaces that are recorded in the PIT entry. On the other hand, if there is no matching entry in its PIT, it will forward the interest packet toward the content provider based on information in its FIB.

In-network caching is one of the most important features of ICN to reduce duplicated content transmissions and network delay. Contents can be cached at every node (e.g., router) in ICN. For example, users can fetch the contents from a nearby node instead of the remote content provider. Therefore, the workload of the content provider and the network delay can be reduced simultaneously. Caching decision policies play a vital role in ICN. This thesis exploits the concept of in-network caching in different network architectures (e.g., pure ICN networks, ICN-5G networks and ICN-IoT networks), and proposes novel caching decision policies for autonomous vehicle (AV) users.

1.1.2 ICN In-network Caching

Generally, an in-network caching approach includes two parts: a caching decision policy and a caching replacement policy. The caching decision policy decides what contents should be cached at which node. The caching replacement policy decides which content should be evicted when the cache is full. For efficiency reasons, the replacement policy should be performed as fast as possible, which means that complicated replacement policies are unsuitable for ICN [115]. Moreover, even a simple random replacement policy can achieve similar performance results compared to the Least Recently Used (LRU) replacement policy [78]. On the other hand, an efficient caching decision policy can improve the performance of ICN in-network caching significantly [115]. Therefore, how to design an efficient caching decision policy is a crucial issue in the ICN in-network caching research field.

Caching decision policies can be divided into two categories: reactive caching approaches and proactive caching approaches. They are discussed in the next subsections.

1.1.2.1 Reactive Caching Approaches

For reactive caching approaches, contents will only be cached if they were repeatedly requested in the past. If a content has never been requested before, then there is no copy of this content in the cache. Therefore, the first request of a video will have to be served by the remote content provider. Moreover, processing and caching videos at a cache also needs additional time since checking if videos are cached locally (reading) and writing video into the storage memory take time. This means that during a small time period Δt, when a particular content is being cached, requests for the same video will not be served by the cache either.

1.1.2.2 Proactive Caching Approaches

Proactive caching approaches try to predict future contents that will be requested by the users, and pre-cache videos before users are requesting them. Consequently, proactive caching is more efficient than reactive caching, especially for bursts of requests during peak hours. Obviously, how to make accurate predictions is the main issue for proactive caching approaches.

1.1.3 ICN-based Networks

To date, 5G is considered as a key enabling technology for the development of current networks. Using 5G in conjunction with ICN could provide significant performance improvements. As a result, ICN is a promising next-generation network architecture, where future networks could be built on top of ICN. In addition, next-generation networks should deal with the transition from host-centric communications to content-centric communications. By combing ICN with recent technologies in wired and wireless networks, next-generation network architectures could be classified into pure ICN networks (wired networks), ICN-5G networks (wireless networks) and ICN-IoT networks (wireless and wired networks).

1.1.3.1 ICN-5G Networks

Since the amount of wireless traffic is increasing at a fast pace and ICN is a promising candidate network architecture to realize various 5G objectives [76], ICN-5G has great potential for future wireless networks. Compared to the traditional IP network, ICN supports name-based routing, in-network caching and mobility by nature, which makes ICN suitable for wireless networks. The name-based routing naturally decouples contents from the locations. In-network caching enables every node in ICN-5G networks to cache contents. Consequently, 5G-ICN can provide contents to mobile users (MU) with lower latency than 5G networks, i.e., better quality of experience (QoE).

1.1.3.2 ICN-IoT Networks

IoT networks are content-centric in nature. Users or applications focus on "what" not "where". In other words, IoT users (or applications) care about the data itself not where the data is stored in. ICN works in a receiver-driven model. This means users send an interest packet to the network to retrieve a content, and any node in the network that has the requested content can send the content back to the user. Moreover, the content in ICN is named by using a unique and location-independent identifier, so that users can fetch the content by its name instead of its IP address. This feature allows ICN to overcome the limited expressiveness of IP addressing of IP-based networks, which is suitable for IoT networks. Based on these advantages, ICN-IoT networks also have potential to be the next-generation IoT networks, and some pioneer works have already been conducted [8] [69] [35] [64].

1.1.3.3 Software-defined Networking

Software-defined networking (SDN) is an emerging network architecture that decouples the control plane from the data plane [45]. The network intelligence and states are logically centralized to provide a global view of the network. This feature can potentially overcome the drawbacks of the existing works in the field of in-network caching [15] [35] [64] [80] [109]. Some pioneer research works about the combination of ICN and SDN have been conducted recently [22] [41] [94] [102]. However, they only proposed an SDN-based ICN architecture without discussing the caching decision policy. This thesis proposes to leverage the global view provided by the SDN controller to improve the efficiency of in-network caching.

1.1.4 Matrix Factorization Techniques

Recent advances in machine learning algorithms and their applications will have profound impacts on computing, networking and caching [49]. For example, the future popularity of content can be predicted by extreme-learning machine techniques (e.g., matrix factorization (MF) [12], feedforward neural networks [86]). Since the future popularity of content can be predicted, popular contents can be pre-fetched before users request them. In this way, the caching efficiency can be improved.

Since users' preferences are the direct reason that makes videos have different levels of popularity, the popularity of videos can be predicted by using users' ratings on those videos. Singular value decomposition (SVD) and non-negative matrix factorization (NMF) are two typical MF techniques that can be used to predict user ratings in recommender systems (RS) [43]. SVD and NMF can achieve similar performance. The only difference between SVD and NMF is that SVD may generate negative ratings for low rated videos, which is considered not practical in real life networks [37].

The idea of the NMF technique is that there are W latent features that have impacts on the rating conducted by a user on a video. NMF tries to explain the ratings by characterizing both users and videos [43]. By learning the latent features, NMF can predict the ratings of videos that have not been watched by users.

1.2 Motivations

Although a great deal of research on caching has been conducted for traditional IP-based networks, most of it cannot be applied directly to ICN due to its specific features such as caching-transparency, ubiquity and fine-granularity [115]. Hence, it is necessary to conduct research on caching decision policies for ICN. Moreover, with the potential of ICN to incrementally replace the current IP-based Internet architecture, the combination of 5G and ICN, IoT and ICN have become the current trends [8] [35] [64] [69]. However, the in-network caching decision policies for the pure ICN are not suitable for those combinations due to the particular caching requirements and special challenges of the combined network technologies. Hence, it is necessary to design caching decision polices for different potential network scenarios based on their particular caching requirements and special challenges.

Since ICN supports mobility by nature, ICN is more suitable for 5G than the IP-based Internet. Meanwhile, in-network caching is a key component of 5G. There are existing in-network caching approaches [2] [20] [38] [72] [55] [87–88] [100] [101] [106] [120], but they are based on the IP-based Internet which only provides weak support for mobility. On the other hand, although some recent works [42] [63] [84] [108] [114] propose ICN-based caching approaches, they only focus on paradigms for ICN-5G networks without focusing on the caching decision policies. Moreover, 5G users will experience more frequent handoffs and shorter connection durations in 5G networks due to the short transmission range of millimeter wave (mmWave) [3]. Unfortunately, most of the existing caching approaches do not consider the impact of frequent handoffs in 5G networks. Due to the aforementioned problems, it is essential to design efficient caching decision policies for ICN-5G networks.

Another challenge for next-generation networks is related to ICN-IoT networks. As devices in ICN-IoT networks are typically battery-powered, energy efficiency is a major challenge for ICN-IoT networks. Through caching IoT data at different nodes (such as a content router, a base station (BS), etc.), IoT devices can stay in sleep mode for a longer period of time and therefore reduce the overall energy consumption. However, caching IoT data is more challenging than caching traditional Internet data since it is only valid for a limited period of time after being generated by the content producer [90]. Once the IoT data has expired, it becomes meaningless for users and will be dropped immediately. Some pioneering research works have been performed recently [8] [35] [64] [69] [90] to leverage in-network caching to gain benefits (i.e., energy efficiency) for ICN-IoT networks. Unfortunately, they only used simple caching decision policies, such as random caching and LCE (leave copy everywhere) [1], which are inefficient for saving energy in ICN-IoT.

Motivated by the lack of research on caching decision policies for ICN-IoT networks, this thesis designs a novel caching decision policy for ICN-IoT networks.

The motivation of this thesis is to make original contributions to the ICN community, shed light on designing caching decision policies for different scenarios in next-generation networks.

1.3 Contributions of this Research

The research focuses on developing ICN in-network caching decision policies that can be used to simultaneously improve the performance of different network scenarios and the QoE of the end users. Some advanced techniques are also leveraged to enhance the efficiency of ICN in-network caching. To date, the contributions to the literature resulted from this research are listed below.

- Z. Zhang, C.H. Lung, I. Lambadaris, M. St-Hilaire, S.S. Rao. "Router Position-Based Cooperative Caching for Video-on-Demand in Information-Centric Networking", *Proceedings of the 2017 conference on 41st Annual IEEE Computer Software and Applications Conference* (COMPSAC), pp. 523-528, July 2017. (Chapter 3)
- Z. Zhang, C.H. Lung, I. Lambadaris, M. St-Hilaire, "When 5G Meets ICN: An ICN-based Caching Approach for Mobile Video in 5G Networks", *Computer Communications* (Elsevier), 118:81–92, 2018. (Chapter 4)
- Z. Zhang, C.H. Lung, I. Lambadaris, M. St-Hilaire, "IoT Data Lifetime-Based Cooperative Caching Approach for ICN-IoT Networks", *Proceedings of the 2018 IEEE International Conference on Communications* (ICC), pp. 1-7, May 2018. (Chapter 5)

- Z. Zhang, C.H. Lung, M. St-Hilaire, I. Lambadaris, "Smart Caching: Empower the Video Delivery for 5G-ICN Networks", *Proceedings of the 2019 IEEE International Conference on Communications* (ICC), pp. 1-7, May 2019. (Chapter 6)
- Z. Zhang, C.-H. Lung, M. St-Hilaire, I. Lambadaris, "An SDN-based Caching Decision Policy for Video Caching in Information-centric Networking", to appear in *IEEE Transactions on Multimedia*. (Chapter 3)
- Z. Zhang, C.-H. Lung, M. St-Hilaire, I. Lambadaris, "Smart Proactive Caching: Empower the Video Delivery for Autonomous Vehicles in ICN-based Networks", ready for submission. (Chapter 6)

1.4 Thesis Organization

The rest of this thesis is organized as follows:

In Chapter 2, related research works for ICN in-network caching in pure ICN networks, ICN-5G networks and ICN-IoT networks are provided. Advanced techniques (e.g., SDN and machine learning techniques) which can be used to improve the efficiency of ICN in-network caching are also presented in this chapter.

In Chapter 3, the caching decision problem is formulated as a 0-1 static integer linear programming (ILP) problem. By introducing the notation of time, the formulated 0-1 static ILP problem becomes a 0-1 dynamic ILP problem which is NP-hard. As future video requests cannot be known in a real network, this chapter uses the next time slot's video requests as the input and uses the current time's optimal solution as the caching decision. Therefore, a more accurate optimal solution for the dynamic networks which change its states dynamically can be found. In order to overcome the high computational complexity of finding optimal solutions, a light-weight cooperative caching decision policy is proposed. The proposed approach (called router position-based cooperative caching (RPC)) is based on the router's topology position to cache popular video on the edge routers. Since the proposed approach does not require knowledge of the popularity of videos a priori, it is more practical compared to existing approaches [1] [7] [17] [25] [47–48] [62] [79] [103]. This chapter also evaluates the proposed approach with a realistic topology and real data traces. Simulation results show that the proposed approach outperforms existing approaches in terms of the average number of hops and server load ratio.

Since the proposed RPC approach makes caching decisions locally, the efficiency can be improved if the caching decisions are made from a controller with a global view. The concept of SDN is leveraged in this chapter to improve the performance of caching. With the global view of the network, a more efficient caching approach is proposed. Through simulations, the proposed SDN-based approach is more efficient than RPC, and the performance is close to the optimal solution.

In Chapter 4, motivated by a few research works that aim at reducing the retrieval delay due to frequent handoffs in ICN-5G networks, an ICN-based caching decision policy for ICN-5G networks is proposed. Since user mobility has rarely been considered in existing works [2] [20] [38] [72] [87–88] [100] [101] [106] [120], this chapter exploits user mobility to reduce the retrieval delay caused by frequent handoffs. Videos can be retrieved from the router which is directly connected to the BS instead of the original content provider when a handoff happens. Simulation results show that the proposed caching decision policy outperforms the traditional IP-based RAN (radio access network) caching and a recent proposed ICN-based caching decision policy [114] in terms of retrieval delay and network traffic reduction.

Chapter 5 applies the concept of in-network caching in ICN-IoT. Specifically, this chapter proposes a novel cooperative caching decision policy based on the IoT data lifetime and user request rate to improve the energy efficiency of ICN-IoT networks. By caching IoT data at different nodes (such as content routers, BSs, etc.), IoT devices can stay in sleep mode for a longer period of time and, therefore, reduce the overall energy consumption. With the help of an auto-configuration mechanism, the proposed IoT data lifetime-based cooperative caching (LCC) decision policy can dynamically adapt to the change of request rate. Extensive evaluations were performed and the simulation results show that LCC outperforms existing approaches in terms of total energy consumption (reduction up to 40%) and average number of hops (reduction up to 20%), which is also directly related to the response time.

In Chapter 6, a novel hierarchical proactive caching approach is proposed for autonomous vehicle (AV) users. By using the NMF technique, the users' future ratings on videos can be predicted. To solve the shortcoming of the NMF technique that generates inaccurate predictions for high rated but unpopular videos, the proposed approach also takes video historical popularity into consideration. Thus, the users' future demands can be predicted based on the user preferences (i.e., the predicted ratings) and the historical popularity of videos. Since the traveling route, velocity and current location information of AV users can be easily obtained from the self-driving system of AVs, the future position of AVs can be predicted based on this information. As a result, the proposed approach can decide what videos should be cached at which road side unit (RSU) before the AV users arrive. The proposed approach is evaluated in two scenarios: a highway scenario and a grid street scenario. The simulation results show that the proposed proactive caching is more efficient in terms of cache hit ratio and the average number of hops compared to existing approaches.

Finally, Chapter 7 concludes the thesis by providing an overview of the main results and discusses the potential research directions.

Chapter 2: Literature Review

2.1 ICN in-network Caching

Caching can be defined as having data, information and object temporarily stored in a location for predictive usage on frequent or closely related interval [1]. In ICN, every node (e.g., content router (CR)) has the capability to cache contents or a part of a content locally, which makes caching in ICN become in-network caching. Since ICN works in a receiver-driven model and uses content's name instead of IP address for routing, it decouples contents from the locations and supports mobility. When requesting content, the user issues an interest packet which carries the name of the content to his/her neighbor CRs. If any of the neighbor CRs has the content, the user can fetch the content from the storage memory of that CR. Otherwise, this interest packet will be forwarded based on the information in each of the forwarded interest packets in the PIT. The PIT stores the interest packets that CR has received but not satisfied yet. When users request contents, they don't care where the contents are (e.g., at the BS, router, gateway, or the original content provider), they only care about how fast they can fetch the contents. Therefore, if the nearest node has the requested content, the user can fetch the content from the node directly instead of from the original content provider (typically located far away). In this way, the retrieval delay and network traffic can be reduced considerably.

As in–network caching is a key feature of ICN, how to improve the efficiency of in-network caching has become a curial problem. Since the caching decision policy plays a key role in in-network caching, how to design an efficient caching decision policy is a challenging issue. In fact, many caching decision policies in ICN have been proposed in recent years [1] [7] [17] [25] [47–48] [62] [79] [103]. Generally, the in-network caching can be categorized into two types: coordinated and non-coordinated.

Non-coordinated caching approaches have less overhead and are simpler than coordinated approaches. However, they have more overlapped contents and lower hit ratios. For example, Jacobson et al. [1] proposed leave copy everywhere (LCE). This approach lets every router cache all the packets that go through it, which leads to a high redundancy. Leave copy down (LCD) [47] and move copy down (MCD) [48] have been used to replace LCE for the sake of reducing object redundancy. In [7], *Prob caching* is proposed to cache content by a fixed probability. Carofiglio et al. [17] proposed the latency-aware caching (LAC) policy which takes popularity and latency into consideration. They calculate a probability according to the content's popularity and latency, and let the routers decide to cache the content according to this probability. Evaluation results showed that they achieved a higher hit ratio and lower delivery time compared to LCD. However, all the approaches mentioned above let each router make its own decision locally. In other words, there is no cooperation between the routers which prevents them from having a global view to make better caching decisions.

On the other hand, coordinated caching approaches can achieve higher hit ratios and lower network delay at the expense of higher overhead and complexity. "Coordination", "cooperation" and "collaboration" are all used in ICN literature for coordinated caching approaches. Coordinated in-network caching carefully picks contents to store with the intention of avoiding duplicates in the cache. In WAVE [25], an upstream router recommends to its downstream router the number of chunks to be cached. As a popularity-based cache replacement policy, WAVE can achieve effective performance compared to *Prob caching* [7]. However, WAVE has a high cooperation overhead for each recommendation from an upstream router to a downstream router. Salah et al. [79] proposed a centralized coordinated caching approach by designing a caching controller in each domain. The caching controller is responsible for deciding and advertising the caching decisions to each caching node. This approach is complicated and has a high overhead.

Liu et al. [58] took video drop ratio into consideration to design their user-behavior driven caching approach. Li et al. [51] presented a cooperative caching strategy for ICN video delivery which combines directory-based and traditional hash-based caching approaches. However, the cooperation only happens among the one-hop neighbors instead of the whole network. Besides, this approach also introduces some extra delay and overhead due to the fact that a router only caches a part of the segments of a video. It needs to collaborate with other routers to satisfy a user's request for an entire video.

Several light-weight approaches are designed to simplify the coordinated caching approach to suit ICN caching requirements. For example, Ming et al. [62] proposed a light-weight aged-based cooperative caching approach (ABC) with an aim to spread popular contents to the network edge. This policy assigns an age to each content according to the content's popularity and the distance to the content server. However, ABC requires knowledge of content's popularity a priori which is impractical.

Xu et al. [107] designed the popularity-driven caching location and searching (P-CLS) approach which considers router's position and video's popularity. However, similar to LCD [47], once a hit occurs, the hit chunk will be pulled down from the upstream router to the downstream router. They did not regard router position as a parameter which makes a notable impact on the performance of ICN in-network caching.

A partial popularity-based approach is proposed in [70], they only cache part of the total requested contents by comparing content's popularity. They also consider the dynamic content popularity. Authors in [14] present MPC, a popularity-based caching approach which will cache a content once its number of requests exceeds the popularity threshold.

But similar to WAVE [25], MPC also induces high overhead during each recommendation from a router to its neighbors. Moreover, it is inefficient to push popular contents to edge routers as all routers have the same popularity threshold.

2.2 In-network Caching in ICN-5G networks

The rapid development of wireless networking technologies and mobile devices has led the dominated traffic in cellular network to be changing from voice and text to data content, especially the big files such as video files. The rich bandwidth and high downloading speed provided by 5G technologies make more and more MUs watch videos on their mobile devices. However, the wireless network will be facing a tremendous traffic. Therefore, how to reduce the traffic load for wireless network and improve the user QoE have become major challenges.

Deploying caches at the edge of wireless networks, especially in the RAN, is regarded as a promising way to alleviate the increasing pressure of wireless network traffic growth and improve MU QoE. Generally, RAN devices have storage and computing capabilities in 5G networks, therefore it is possible to deploy caches at the RAN. With RAN caching, MUs can fetch cached contents and thus significantly improving the MU QoE and reducing the backhaul traffic load.

The existing IP-based RAN caching approaches are based on the packet-level. Unfortunately, they are not content-aware and they suffer from a scalability problem [96]. Moreover, the packet-level RAN caching is inefficient to cope with todays' content-oriented 5G networks. To overcome these shortcomings, the name-based forwarding and routing mechanisms of ICN can be leveraged to realize content awareness for 5G and make

5G more scalable. With the help of its in-network caching, the content retrieval delay and the amount of mobile traffic can be reduced effectively.

ICN is a state-of-the-art networking paradigm. It naturally supports client mobility and can make 5G mobility management simple. For example, in traditional IP-based wireless network, the IP address of a MU will change if the MU moves to another location. To deal with this issue, cellular network service providers have to use additional methods or protocols such as mobile IP. However, mobile IP suffers from issues such as triangular routing, control overhead to manage the routing states between the current point of attachment and the home agent. In contrast, MUs can fetch contents without any mobility issue in ICN-based wireless networks, e.g., ICN-5G networks. Because ICN is a receiver-driven network, its naming mechanism decouples the location and identity. Each content, user, and content provider has a unique name in ICN, and this name will not change no matter how their location changes. MUs send interest packets to the networks to fetch contents. Any node that has the requested contents can send the contents back to them based on the name of the MU. Recently, ICN-5G (the combination of 5G and ICN) has been proposed [42] [63] [84] [108] [114]. In [42], a mobility tracking node is used to redirect consumer's request from an old position to a new position of a producer. In this way, the content retrieval delay can be reduced once a handoff occurs. However, they are focusing on the producer mobility issue. For video services, the producer (i.e. video provider) has no mobility while consumers (i.e. users) have high mobility. Nishiyanma et al. [68] have proposed a routing-based mobility architecture to provide seamless mobility management for 5G by adopting ICN. The evaluation results show that their proposed architecture reduces signaling overhead and paging overhead significantly. In [76], an application-driven framework is introduced to realize ICN-5G. The ICN-5G architecture can achieve the mobility as a service (MaaS) objectives. The work in [63] proposed an efficient access control framework for ICN-5G. Legitimate users can access the content directly without verification/authentication by the content provider authentication mechanism, which can reduce the delivery latency.

In-network caching is the key feature of ICN and as a result, a lot of work has been conducted in ICN caching. However, there are only a few papers discussing ICN-5G caching algorithms [40] [84] [96] [108]. In [114], a cooperative caching approach is proposed to reduce cache redundancy and improve the diversity of content distribution. Content popularity and availability are considered to make the caching decision in a probabilistic way. However, they did not propose any method to reduce the retrieval delay once a handoff occurs, which is a crucial issue for 5G video caching. For the sake of improving the performance of video distribution, Si et al. [84] investigated the use of harvested bands for proactively caching videos closed to the users. They formulated the allocation of harvested bands as a Markov decision process. Based on the Markov decision process, a spectrum management mechanism is developed to improve the efficiency of proactive video caching and spectrum utilization. This thesis focuses on the caching algorithm, therefore, the spectrum management is beyond the scope of investigation. In [108], an innovative video streaming solution for ICN mobile networks is proposed. In order to achieve better performance, a content-centric multi-region video content management method and a mobility-adaptive content-centric video delivery strategy is designed. The optimal video provider and delivery path can be achieved by these two methods. However, their caching strategy only considers the caching space which is inefficient in improving the performance of video distribution. Wang et al. [96] explored current content delivery and caching techniques in 5G networks. Based on their trace-driven evaluation results, the deployment of in-network caching into 5G networks can potentially help reduce mobile traffic compared to RAN caching and evolved packet core (EPC) caching. Conclusively, ICN-based caching can cope with the ever growing demand of mobile users for huge amount videos efficiently. However, they did not propose any new caching decision strategy.

2.3 In-network Caching in ICN-IoT networks

The rapid development of networking technologies, such as 5G, also boosts the development of IoT networks. Billions of devices will be connected to the Internet, about 44 trillion GB traffic will be generated over the next 5 years [27], which will bring a huge pressure for the current IP-based networks. The three major challenges that IoT networks are facing are limited expressiveness of IP addressing, complex mobility support and the requirement of energy efficiency. The first two challenges can be easily resolved by using ICN as the infrastructure for IoT networks. Then, the third challenge will be the only challenge left.

There are two major methods to improve the energy efficiency of IoT devices: energy-saving mechanisms and charging solution which is beyond the scope of this thesis. Designing particular protocols [81] for IoT networks is a basic method to improve the energy efficiency for IoT devices. Since the radio module is the main component that causes energy consumption of IoT devices, some researchers have tried to optimize radio parameters to make the hardware more efficient [104]. Moreover, idle states are major sources of energy consumption at the radio component. Therefore, letting IoT devices stay in sleep mode can also save energy. Duty cycling schemes are the basic methods to schedule the IoT devices in different states based on the network activity [5].

Deploying in-network caching at intermediate nodes, e.g., CRs and BSs, is another efficient method to let IoT devices stay in sleep mode. When the cached data are requested by users or applications, the intermediate nodes can send the requested data back to them directly. Hence, these requests will not be forwarded to wake up the IoT devices. Consequently, IoT devices can stay in sleep mode most of the time to reduce their energy consumption.

Although the caching decision policy in ICN has been extensively studied [1] [7] [17] [25] [47–48] [62] [79] [103], the ICN caching decision approaches cannot be applied to IoT directly, since the IoT data items are usually transient and small [75]. Unfortunately, very few studies have been conducted related to the ICN-IoT caching decision approach.

Baccelli et al. [8] explore the feasibility, advantages, and challenges of an ICN-based approach in IoT. Through ICN experiments in a life-size IoT deployment, they show that caching provides significant benefits to ICN-IoT in terms of energy efficiency. However, the paper does not consider the temporal properties of IoT data for caching decision making, even though they mention the freshness requirement of IoT data.

The study in [90] proposes a probability-based caching decision which makes a trade-off between IoT data freshness and multi-hop communications cost so that IoT data can be cached at the content router. An auto-configuration mechanism is used to adjust the data caching probability by comparing the data freshness and the multi-hop communication cost. Least Fresh First (LFF) replaced policy is used as the replacement policy. However, their caching decision approach is quite complicated and needs heavy computation, which is not suitable for the content router.

Similar to [90], the authors in [69] also consider the IoT data freshness by maintaining a timer. However, they regard the capacity of sensor energy as a parameter when making a caching decision for the IoT with energy harvesting. As their caching approach is threshold-based, a threshold adaptation is introduced to allow the nodes to dynamically adjust the parameter of caching to achieve better performance. However, the approach only caches data at the wireless gateway which is inefficient due to its limited amount of resources. The ideal caching decision approach should not only cache data at the gateway node but also at all intermediate nodes between the gateway node and the content consumers. Through cooperating among the nodes, the caching efficiency can be significantly improved, and IoT devices can spend more time in sleep mode without being activated too frequently, hence reducing the overall energy consumption.

The recent research work in [35] leverages the in-network caching of ICN to gain benefits for IoT energy efficiency. The authors propose a simple side protocol called cooperative caching side-protocol (CoCa) to exploit data names together with the interaction between cooperative caching and power-save sleep capabilities on IoT devices. By performing extensive, large-scale experiments on real hardware with IoT networks, they report that the IoT devices can significantly reduce their energy consumption while maintaining recent IoT data availability above 90%. However, their caching decision approach is based on random caching with a probability $p = 0.5$, a widely used approach in previous studies. The random caching approach is inefficient compared to existing approaches, such as [90]. Since the request rate varies, some contents may have a higher request rate in a certain time period. If they are not cached at this time, and the approach cannot adjust the caching probability dynamically to cache them, then this will lead to poor caching performance.

2.4 Machine Learning for Proactive Caching

Proactive caching is an efficient approach to improving users' QoE and reducing the network backhaul load for 5G networks, hence a great deal of research has been conducted on the topic. Most of the existing research works about proactive caching for 5G networks are based on traditional IP networks. However, IP-based networks are inefficient for caching, because users cannot retrieve the cached videos without any other additional techniques, e.g., domain name system (DNS) redirection. Since ICN uses content name instead of IP address for communications, it makes it suitable for in-network caching and mobile content delivery. Hence, it is believed that ICN is a more suitable candidate for 5G networks. Although a great deal of research has been conducted on in-network caching [14] [96], proactive caching for 5G networks has not been well investigated yet. Some pioneer works about the combinations of ICN and 5G networks [42] [63] [84] [108] [114] have been conducted, but most of them focus on the architecture level.

In previous studies [9–10], videos are only cached based on historical popularity. But even a popular video cannot be guaranteed to remain popular in the future. To address this issue, recent research efforts [5] [12] [36] [37] [59] [71] [110–111] [119] use machine learning techniques to predict the future demands of users.

Using ML algorithms for improving the performance of traffic engineering is a hot topic nowadays [5] [12] [36] [37] [59] [71] [110–111] [119]. The traditional ICN in-network caching approaches are usually reactive. Nodes in ICN decide "where" to cache "what" contents based on the previous requests. Even proactive caching approaches pre-cache contents based on the previous requests without any predictions. However, the popularity of contents and the location of users are changing over time, which lowers the efficiency of traditional ICN in-network caching. If the popularity of videos and the mobility of users can be known in advance, then popular contents can be cached at nodes that are closer to the user future location. Therefore, the caching efficiency can be significantly improved. Thanks to the rapid development of ML techniques, both the content popularity and the user mobility can be predicted by using proper ML techniques. Based on those predictions, proactive caching can be more intelligent and more efficient.

Statistical models, such as auto-regressive and moving average (ARMA) [36] have been used for predicting the video popularity. Similarly, methods based on neural network are also popular to predict the video popularity. For example, Yin et al. [110] use echo state networks (ESN), a type of RNN, to predict users' future demands. Although neural network model-based methods can achieve more accurate predictions compared to approaches using historical popularity only, the prediction accuracy of those methods highly depends on the configured parameters. A slight change of in the configured parameters can lead to inaccurate predictions. Moreover, finding the best set of configured parameters is also challenging. Furthermore, neural network model-based approaches do not consider users' preferences which also plays a vital role in improving QoE of mobile users.

In recent years, RS has been used to improve the efficiency of caching since it can shape the network traffic. For example, RS is used as a TE tool to shape the content demand in [21]. The authors in [21] first formulated the caching problem and the recommendation problem as a joint optimization problem. To provide a light-weight algorithm for its solution, they also proposed a practical algorithm. An experiment with 40 YouTube-using volunteers was conducted in [46]. The experiment results show that users may change their original content request when they are recommended videos that are cached locally and may attract them. Hence, the efficiency of caching can be improved by implementing RS on a cache system. Authors in [105] leverage the algorithms of RS to predict what a single user is going to watch in the future. Based on this prediction, they can pre-fetch videos before user requests come. However, their prefetching algorithm is designed for a single user, which does not reduce the network traffic, since duplicate videos still need to be transmitted from the content provider to the users. In [97], an important user is selected as the helper to cache the recommended contents which are generated by a RS in a mobile network. Other users can fetch their interested contents from the important user. A more recent work [83] proposed a soft cache hits approach which can provide users a relevant content when the requested content is not cached at the local cache.

Instead of implementing RS on a cache system, this thesis plans to leverage the algorithms of RS to predict the future user demands. Collaborative filtering (CF) [82] is the basic algorithm of RS due to its efficiency and simplicity to implement. In order to improve the performance of RS, matrix factorization (MF) technique is proposed in [44]. Singular value decomposition (SVD) [44] and non-negative matrix factorization (NMF) [44] are two major techniques to perform MF. Even though SVD can achieve similar performance in general, NMF is believed to be more suitable for preference prediction due to the fact that SVD will generate negative predicted ratings, which does not apply to ratings for this problem in the real world [37]. Generally, MF-based approaches can predict if a video will be liked by a user through learning the latent features of users and videos, and predicting the rating of videos based on these two latent features. Since MF-based approaches consider the users' preference, they can achieve a notable improvement in terms of the user satisfaction, the average video retrieval delay and the hit ratio compared to traditional proactive caching approaches [12] [37].

Chapter 3: Caching Approach for Pure ICN Networks

3.1 Introduction

Nowadays, video traffic accounts for a huge volume (more than 73%) of the total Internet traffic. Moreover, the ultra-high-definition (UHD) will attribute to 20.7% of video traffic in 2020 [27]. This trend causes a huge challenge for today's Internet on how to distribute videos efficiently. Unfortunately, the current IP-based Internet paradigm is designed for host-to-host communications, it is not suitable for content delivery, especially for the video delivery due to the fact that videos are usually large files.

Videos can be cached on each router in ICN. This means that users can fetch the content from a nearby router instead of the remote video server (i.e. content provider), so that the video server load and the network delay can be reduced simultaneously.

A caching approach includes two parts: a caching decision policy and caching a replacement policy. The caching decision policy is used to decide what content should be cached, whereas the caching replacement policy is used to decide what content should be evicted. Because of the requirement of efficiency, the replacement policy should be performed as fast as possible, which makes complicated replacement policies unsuitable for ICN [115]. Moreover, even a simple random replacement policy can achieve similar performance results attained by the Least Recently Used (LRU) replacement policy [78]. On the other hand, an efficient caching decision policy can improve the performance of ICN caching significantly [115]. Therefore, how to design an efficient caching decision policy is a crucial issue in the ICN research field.

In this chapter, the caching decision problem is first formulated as a 0-1 integer linear programming (ILP) problem. Then, this thesis introduces the notion of time and divides it into time slots. By finding the optimal solution for each time slots, a more accurate optimal solution can be achieved as the theoretical bound for further research.

Since the formulated ILP problem is NP-hard, a router position-based cooperative (RPC) caching decision policy which is a novel and practical approach is proposed in this chapter. The aim of the proposed approach is to allow edge routers to cache the most popular videos to reduce the server load, network usage, and network delay.

Software-defined networking (SDN) is a promising technology that can enhance traffic engineering. As SDN can provide a global view of the network, and forwarding decisions are made by the centralized controller [45], SDN is a suitable technology that can be used to overcome the drawbacks of the existing approaches in pure ICN. Hence, an SDN-based practical approach is also proposed in this chapter.

3.2 System Model and Problem Formulation

This section first describes the system model for the proposed SDN-based centralized caching decision policy. Then, it formulates the caching decision problem as a 0-1 integer linear program problem. The symbols and their definition that will be used in the rest of this chapter are summarized in Table 3.1.

3.2.1 System Model

In this thesis, V is used to indicate the set of videos where v ($v \in V$) stands for a specific video and $|V|$ is the total number of videos. Users send interest packets to edge routers directly to retrieve videos. The set of requests for the videos is denoted as Req, where $|Req|$ is the total number of requests. Each request in this set is represented by $req_n(v) \in Req$ which indicates the total number of requests for video v from node n.

All nodes (e.g. routers) in this thesis have the capability to cache videos that go through them. Let N denote the set of nodes, and each node is denoted by $n \in N$, where $|N|$ is the total number of nodes, including routers and the video server. Let $n = 0$ indicate the video server and $n \neq 0$ represents the routers. The caching state of node n can be represented by an array of binary values, denoted as $c_n(v)$. If video v is cached at node n, $c_n(v) = 1$; otherwise, $c_n(v) = 0$. Let C denote the caching state set of all nodes, we can easily get $\forall\, c_n(v) \in C$. Since each node (except the video server) has a limited cache size, the constraint that ensures that each node cannot cache more videos than its cache size is denoted as follows:

$$\sum_{v \in V} L_v \cdot c_n(v) \leq Z_n, \quad (n \in N, n \neq 0) \tag{3.1}$$

where L_v is the size of video v, if $n \neq 0$, Z_n is the cache size of router n; otherwise $n = 0$ means that node n is the video server. Since the video server has all videos, we assume it has infinite storage, i.e., $Z_0 = \infty$. Furthermore, we use $R(n)$, where $n \in N, n \neq 0$ to represent how many requests can be served from switch n. $R(n)$ can be calculated by Equation (3.2):

$$R(n) = \sum_{v \in V} req_n(v) \cdot c_n(v), \quad (n \in N, n \neq 0) \tag{3.2}$$

3.2.2 Problem Formulation

This subsection first formulates the caching decision problem as a 0-1 ILP. An optimal solution can be obtained without considering the effect of time. However, the solution is

Table 3.1: Notations

Symbol	**Definition**
V	Set of videos
$\|V\|$	Total number of videos
Req	Set of requests for the videos
$\|Req\|$	Total number of requests
$req_n(v)$	Total number of requests for video *v* from node *n*
N	Set of nodes
$\|N\|$	Total number of nodes
$c_n(v) \in \{0,\ 1\}$	Caching state of node *n*
C	Caching state set of all nodes
L_v	Length of video *v*
$R(n)$	Number of requests can be served from node *n*
G	Gain (the total reduced transmission delay)
$D(n)$	Reduced video transmission delay by performing caching at node *n*
$d_{mn}(v)$	Video transmission delay for video *v* from node m to node *n*
Z_n	Cache size of node *n*
$a_{mn} \in \{0,\ 1\}$	If node *n* retrieves video from node *m*
T	Simulation time
$req_n(v,t)$	Number of requests for video v from node *n* at time t
$c_n(v,t)$	Caching state of node *n* at time *t*
$D(n,t)$	Reduced video transmission delay by performing caching at node *n* from time 0 to time *t*
$d_{mn}(v,t)$	Video transmission delay for video v from node *m* to node *n* at time *t*
G(t)	Total reduced video transmission delay (gain) from time 0 to time *t*
$a_{mn}(t) \in \{0,\ 1\}$	If node *n* retrieves video from node *m*
P	Caching decision policy

not suitable for a real-time scenario due to the fact that the popularity of videos and the number of requests vary over time. Hence, this subsection divides the time into time slices.

By solving the optimal solution for each time slice, more accurate optimal solutions can be obtained and applied to dynamic networks.

3.2.2.1 Static Scenario

First, this subsection considers the caching decision problem as a static scenario like that in existing approaches [16] [28] [29] [53] [56] [87]. Since the transmission delay is the key performance metric for video streaming services in network caching, e.g., long waiting time for video buffering may lead users giving up watching the video [56], the goal of this work is to minimize the video transmission delay. By performing caching, users can retrieve videos from nearby nodes; therefore, the video transmission delay can be reduced significantly. The total reduction in video transmission delay is defined as gain (denoted as *G*). Since users can retrieve a video either from the video server, or from a router, the restriction of (3.2) can be removed, and it can be expressed as follows:

$$R(n) = \sum_{v \in V} req_n(v) \cdot c_n(v), \qquad (n \in N) \tag{3.3}$$

Therefore, *G* can be calculated as follows:

$$G = \sum_{n \in N} R(n) \cdot D(n) \tag{3.4}$$

where $D(n)$ represents the reduced video transmission delay by performing caching at node *n* which can be expressed by the following equation:

$$D(n) = \sum_{v \in V} \sum_{m \in N} (d_{0n}(v) - d_{mn}(v)), \qquad (n \in N) \tag{3.5}$$

where $d_{0n}(v)$ is the video transmission delay for video v from the video server to node *n*, $d_{mn}(v)$ is the video transmission delay for video v from node *m* to node *n*. When *m* = *0*, $d_{mn}(v)$ stands for the video transmission delay from the video server to router *n*; when *m*

= n, $d_{mn}(v) = 0$, which means the video is cached locally (i.e., there is no transmission delay).

By substituting Equations (3.3) and (3.5) into (3.4), Equation (3.4) can be reformulated as follows:

$$G = \sum_{n\in N}\sum_{m\in M}\sum_{v\in V} a_{mn}\cdot req_n(v)\cdot c_n(v)\cdot (d_{0n}(v) - d_{mn}(v)) \tag{3.6}$$

where a_{mn} is a binary value, $a_{mn} = 1$ indicates that node n fetches the video from node m; otherwise there is no video transmission between node n and m.

Then, the objective function is formulated as a 0-1 ILP, expressed as follows:

$$\max G \tag{3.7}$$

$$s.t. \sum_{v\in V} L_v \cdot C_n(v) \leq Z_n, (n \in N, n \neq 0) \tag{3.8}$$

$$\sum_{m\in N} a_{mn} = 1, \ (n \in N) \tag{3.9}$$

$$a_{mn} \in \{0, 1\}, \ (m \in M, n \in N) \tag{3.10}$$

As defined in Constraint (3.1), L_v is the length of video v. As mentioned in Section 3.2.1.1, *N* is the set of nodes. Constraint (3.9) is used to guarantee that only one node (a router or the video server) sends the requested video back to the user.

The caching problem has been proven to be NP-hard in [52]. Since the above formulated 0-1 ILP is similar to the one formulated in [52], then, we can conclude that the formulated problem in this thesis is also NP-hard. However, using exhaustive searching to find an optimal solution is unacceptable due to its high computational complexity. Instead, branch and cut method is a better method which is integrated in many general solvers, such as

CPLEX [91], to find an optimal solution. Therefore, if all the prior information is known, the maximum *G* can be found by CPLEX.

3.2.2.2 Real-time Scenario

By solving the objective function Equation (3.7), an optimal caching decision can be found. However, the achieved optimal caching decision is not suitable for a real-time scenario due to the fact that the video popularity and request pattern vary over time. Since the aforementioned optimal caching decision is achieved based on the historical data, it is only valid for a particular time period.

If we want to find a more accurate optimal solution for dynamic networks, it is essential to divide the time into time slots and use the current time slot's optimal solution as the next time slot's caching decision. In this way, the optimal caching decision can be calculated incrementally over time. Through evaluating the performance of the caching decision over time, a more accurate optimal solution for dynamic networks can be obtained.

Let $req_n(v,t)$ stand for the number of requests for video v from node *n* at time $t \in T$, where $req(v,t) \in Req$. The caching state of node *n* at time *t* is denoted by $c_n(v,t)$. By performing a caching decision policy P, $c_n(v,t) \overset{P}{\rightarrow} c_n(v,t+1)$, the new caching state of node *n* at time *t*+1 can be achieved. Hence, Constraint (3.8) can be reformulated as follows:

$$\sum_{v \in V} L_v \cdot c_n(v,t) \le Z_n, \qquad (n \in N, t \in T, n \neq 0) \tag{3.11}$$

Let $R(n,t)$ indicate the number of requests that are served at node *n* from time *0* to time *t*. Then, Equation (3.3) can be reformulated as follows:

$$R(n,t) = \sum_{t \in T} \sum_{v \in V} req_n(v,t) \cdot c_n(v,t) \tag{3.12}$$

where $\sum_{t\in T} req(v,t)$ represents the total number of requests for video v from time *0* to time *t*.

This thesis denotes the reduced video transmission delay by performing caching at node n from time *0* to time *t* as $D(n,t)$ which can be represented as follows:

$$D(n,t) = \sum_{t\in T}\sum_{v\in V}\sum_{m\in N}(d_{0n}(v,t) - d_{mn}(v,t)) \tag{3.13}$$

where $d_{mn}(v,t)$ is the video transmission delay for video v from node *m* to node *n* at time *t*, $d_{0n}(v,t)$ is the video transmission delay for video v from the video sever to node *n*, $\sum_{t\in T} d_{mn}(v,t)$ indicates the total video transmission delay for video v from node m to node *n* during the time period (from time *0* to time *t*). $G(t)$ is used to represent the total reduced video transmission delay, i.e., *gain,* from time *0* to time *t*. Equation (3.14) is used to calculate $G(t)$.

$$G(t) = \sum_{n\in N} R(n,t)\cdot D(n,t) \tag{3.14}$$

Substitute Equation (3.12) and (3.13) to (3.14), we can get Equation (3.15) as follows:

$$G(t) = \sum_{t\in T}\sum_{m\in N}\sum_{n\in N}\sum_{v\in V} a_{mn}(t)\cdot req_n(v,t)\cdot c_n(v,t)\cdot d_{mn}(v,t) \tag{3.15}$$

Finally, the objective function can be reformulated as follows:

$$\max G(t) \tag{3.16}$$

$$s.t.\ \sum_{v\in V} L_v\cdot C_n(v,t) \le Z_n, \qquad (n\in N, t\in T, n\neq 0) \tag{3.17}$$

$$\sum_{m\in N} a_{mn}(t) = 1,\ (n\in N) \tag{3.18}$$

$$a_{mn}(t)\in\{0,\ 1\},\ (m\in M, n\in N) \tag{3.19}$$

where *N* is the set of nodes, $a_{mn}(t)$ indicates if node n retrieves video from node m ($a_{mn}(t) = 1$ indicates yes; otherwise no).

Proposition 1: As the time t increases, the caching state $c_n(v,t)$ becomes more stable than in the previous time period, namely, the difference between $c_n(v,t)$ and $c_n(v,t+1)$ decreases.

Proof: Let us consider the caching state of a cache-enabled node n in a network. We assume there is one video provider, which can provide V videos for users; that the request generation follows a stationary Poisson process with an arrival rate λ; and that the video popularity follows the Zipf distribution [24]. The cache size of node n is Z_n. $Z_n < V \cdot S_v$; hence, node n cannot cache all the videos. At the beginning, node n will cache every requested video until its cache is full. After that, node n begins to cache the most popular video v and since there is only one video provider and the transmission delay for caching each video is the same for node n, caching popular videos will result in higher gain. With the increase of time t, popular videos are being requested more frequently, which leads to $req_n(v_{popular}, t) > req_n(v_{unpopular}, t)$ and this difference continues to increase. In addition, the top 20% of the videos account for 80% of the total network traffic since the video popularity follows the Zipf distribution; hence, $req_n(v_{popular}, t) \gg req_n(v_{unpopular}, t)$. As a result, the rank of videos becomes increasingly stable, which means that the popularity of the videos does not change substantially from time t to time $t+1$. Since caching the most popular video at node n will result in a larger gain for node n, node n will always try to cache the most popular video. Consequently, the difference between $c_n(v,t)$ and $c_n(v,t+1) \rightarrow 0$, i.e., the caching state becomes increasingly stable.

The current optimal caching decision can be solved by calculating the historical data. The achieved optimal caching decision can be used for the video caching at the next time slice Δt. After that, the optimal caching decision for the next time slice Δt can be solved by calculating the historical data and the new request at the next time slice Δt. In this way, the optimal decisions for the real-time network can be found.

However, the reformulated model for real-time scenario is still NP-hard. Even though a solution can be obtained with a general solver, the computational complexity is extremely high, which is not suitable for a real-time caching system. Therefore, a light-weight and practical approach is needed.

3.3 Router Position-based Cooperative Caching

This section describes the proposed router position-based cooperative caching approach. The main idea of RPC is to cache popular videos closer to the users. Along the video delivery path, the proposed RPC allows a router to calculate its own topology level value by adding 1 to the value of its immediate upstream router's topology level value. Routers also track the access count for each video locally, and stores all the access counts information as a key-value structure (video name; access count).

3.3.1 Principle of RPC

The basic principle of RPC is described as follows:

- Each router has a caching threshold and keeps track of an access count for each video (indicating the video's popularity). Each router decides what video should be cached according to the access count of the videos, its own caching threshold, and the available storage space.

- Least Recent Used (LRU) [47] is used for the replacement policy. Obviously, other methods can be used as the replacement policy as well, such as Least Frequently Used (LFU) [93].
- Along the video delivery path, routers need to transmit their topology level value and root routers' caching threshold to their immediate downstream routers. This cooperation only happens once during the procedure of determining the caching threshold, which has low overhead.
- The caching threshold of root routers is pre-configured. The other routers determine their caching threshold using the proposed caching threshold decision policy described in Section 3.2.3.

3.3.2 How RPC Works

A router decides what video to be cached based on its caching threshold and storage space. A video can be cached if one of the following conditions is satisfied: 1) The access count of the video is greater than the caching threshold of the router; 2) The router has enough space to store the video. If a video access count exceeds a router's caching threshold, but the router does not have enough space to cache it, the router will perform the replacement policy to discard videos until the router has enough space to cache the video.

In RPC, root routers are routers which are connected to video servers directly. Routers collaborate with each other through transmitting their topology level value and root routers' caching threshold value. The principle for setting the caching threshold is as follows: 1) Root routers set a minimal caching threshold all over the network; 2) Downstream router's caching threshold should be less than that of its upstream routers, so that the popular videos can be cached in low level routers, e.g., edge routers; 3) Routers should use the same threshold if they are at the same level in the topology.

The procedure for determining the caching threshold works as follows: at the initial state, the root routers set a default value as their caching threshold, then inform this threshold value and their topology level value to their immediate downstream routers. When a downstream router receives this threshold value, it will use the caching threshold decision policy (illustrated in Section 3.2.3) to calculate its caching threshold and advertise its topology level value and the root routers' threshold value to its immediate downstream routers along the video delivery path. The rest of the routers repeat the above procedure until all routers have a caching threshold.

Algorithm 3.1 Router position-based cooperative caching

```
Input: access count = 0 for all contents
       caching threshold = a default value for root routers, or calculated (as
       explained in Section 2.2.3) for other routers

 1: a request for content arrives
 2: content's access count++
 3: checks whether it has this content
 4: if (has this content) then
 5:    forward it
 6: else if (has enough space) then
 7:    cache it once received from other nodes
 8:    forward it
 9: else if (access count > caching threshold) then
10:    while (not enough space) do
11:       delete least requested replica
12:    end while
13:    cache it
14:    forward it
15: else
16:    forward it
17: end if
```

After the caching threshold is set up, routers can perform the RPC as described in Algorithm 3.1.

In RPC, only a router's topology level value and the root routers' caching threshold value are transmitted in the caching threshold determining procedure, and this procedure only happens once. Therefore, the cooperation overhead is significantly lower compared to other cooperative caching decision policies [25] [51] [79] which require transmitting recommendations between upstream routers and downstream routers for each video.

3.3.3 Caching threshold decision policy

This section describes how a router determines its caching threshold. First, it explains how to calculate the topology level value for a router when it receivers multiple topology level values from its immediate upstream routers. Second, it illustrates the caching threshold decision policy based on the principle of RPC.

3.3.3.1 Topology level decision

As mentioned above, routers can calculate their topology level value as follows:

$$l_i = l_{i-1} + 1 \tag{3.20}$$

where l_i is the topology level value of router i, l_{i-1} is the topology level value of the immediate upstream router. However, if router i has multiple immediate upstream routers, it may receive multiple topology level values from these immediate upstream routers. As shown in Fig. 3.1, edge router B has two immediate upstream routers, edge router A and the root router. As the topology level value increases along the video delivery path, the topology level value transmitted from edge router A to edge router B will be higher than the one received from the root router. Hence, there are two options that router B can adopt to calculate its own topology level value: 1) use A's topology level value; or 2) use the root router's topology level value.

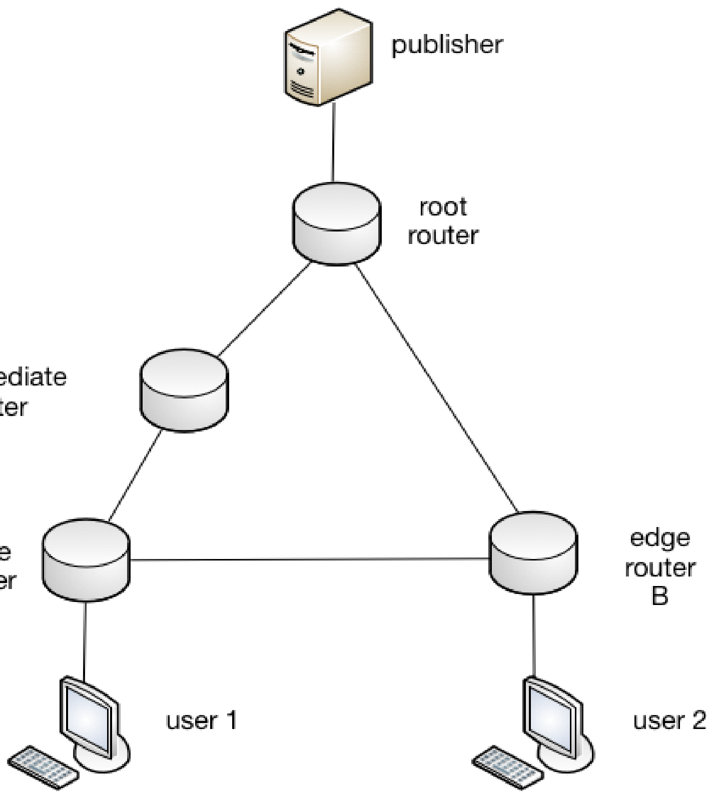


Fig. 3.1: Different topology levels from multiple immediate upstream routers

It is important to note that the topology level and the caching threshold have a positive correlation, i.e., a higher topology level value always leads to a higher caching threshold. However, a higher caching threshold will lower the chance for contents to be cached at the router. As a result, these contents have to be cached in upstream routers which causes the video delivery path to be longer. Moreover, contents also need more time to increase their access count to satisfy the router's caching threshold. Finally, during this period, users have to fetch these contents from the router's upstream routers, or even from the video server. Consequently, the number of hops for retrieving these contents will increase. Hence, for the scenario illustrated in Fig 3.1, router B should use the root router's topology level value to calculate its own topology level value. Therefore, when a router receives multiple topology level values from its immediate upstream routers, it should use the lowest one to calculate its own topology level value. Then Equation (3.20) can be modified as follows:

$$l_i = min\ l_{i-1} + 1 \tag{3.21}$$

where $min\ l_{i-1} \in L$, L is the topology level value set for router i's immediate upstream routers, $min\ l_{i-1}$ is the lowest level value in L. If there are multiple video servers, routers should maintain multiple topology level values correspond to these video servers. Once an interest arrives, router can decide to use which topology level value based on the interest and video server's name prefix.

3.3.3.2 Caching Threshold Calculation

According to the principle of RPC mentioned above, a default caching threshold can be configured for the root routers. Then, other routers' caching thresholds will be determined by two parts: 1) The root router's caching threshold; and 2) Its own topology level. Specifically, the caching threshold determining policy can be described as follows:

$$t_i = \alpha t_r + \beta l_i \tag{3.22}$$

where t_i is router i's threshold, t_r is the root router's threshold, l_i is the topology level value of router i. Both α and β are configurable wrights for t_r and l_i respectively (each is an integer ≥ 0) for the policy.

3.3.4 Performance Evaluation

This section presents the simulation and results. LCE [1] and ABC [62] are chosen as comparisons, because LCE is the default caching strategy in ICN [1] and ABC outperforms other popularity-based algorithms, such as WAVE [25]. A custom-built simulator (written in C++) is used to perform the simulation, since the well-known simulators [18] [26] [66]

[93] do not integrate the caching approach to be compared in this thesis and they are not sufficient for large scale simulation [85]. The simulation is conducted by using a real topology with real data traces. A shortest path routing protocol is applied in the simulation. The reduced video server load ratio and the average number of hops are used as the simulation metrics, which are described as follows.

Reduced video server load ratio: Equation (3.23) is used to calculate the reduced video server load ratio. If there is a video request hit, the video will be transmitted from the router without having to be transmitted from the video server, which causes a reduction of traffic delivery. The reduced video server load ratio is defined as:

$$R = \frac{\sum_{i=1}^{Request\_count} T_i}{F} \tag{3.23}$$

where T_i stands for the traffic reduction contributed by router i and F is the total amount of traffic for all requests transmitted from the video server if no caching is used.

Average number of hops: This thesis uses the average number of hops to indicate the network delay from a general perspective. It is calculated by Equation (3.24).

$$A = \frac{\sum_{i=1}^{Request\_count} h_i}{|Req|} \tag{3.24}$$

where h_i is the number of hops needed to deliver video request i and $|Req|$ is the total number of requests. Since the number of request used in the simulation is high, the average over all requests gives a good approximation of the performance of the various approaches. This thesis uses the average number of hops to indicate the network delay from a general perspective.

3.3.4.1 Topology and Data

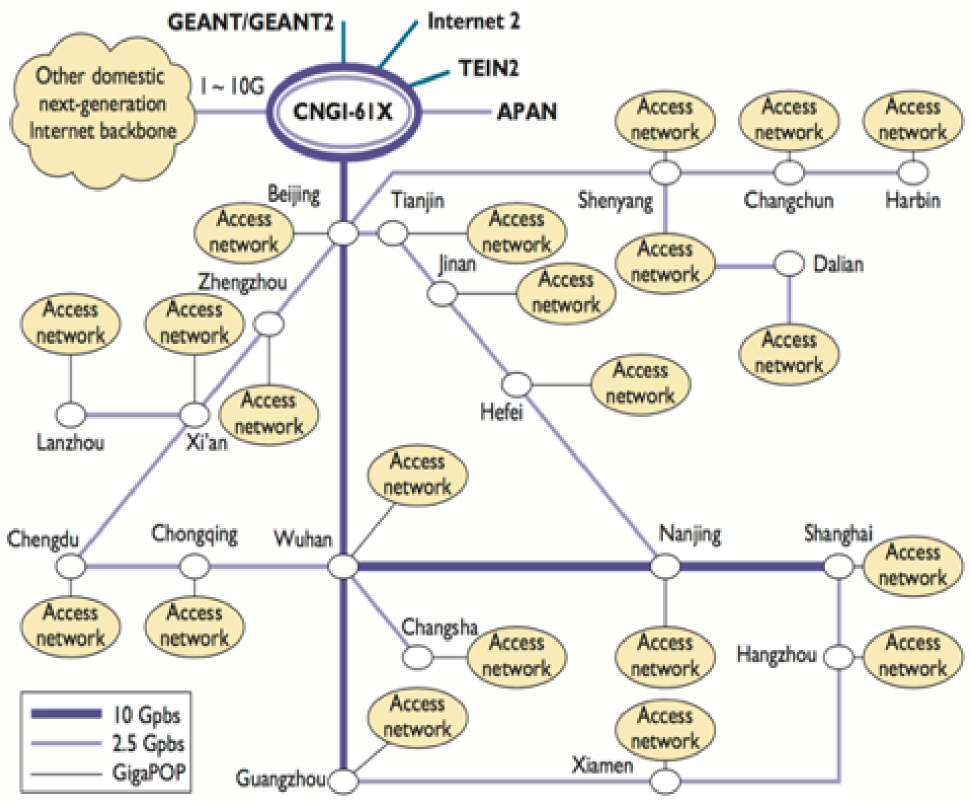


Fig. 3.2: Topology of CERNET2 [40]

CERNET 2 [40] is the largest next-generation Internet backbone of China and it is also the largest native IPv6 backbone network all over the world. As shown in Fig. 3.2, the topology of CERNET 2 is used for the simulation. Each node represents the city's core router, and users request contents from them. This thesis assumes the video servers are deployed in Beijing and Shanghai; therefore, the routers in Beijing and Shanghai are the root routers. Data traces are collected from the video channel of Sina [92]. Sina video attracts more than 80 million users each day. This thesis filters the data traces and gets 278,262 unique requests as the input. And these data are sorted according to the request time. Each video is divided into small chunks. For simplicity, this thesis assumes all chunks have the same size (4,000 bytes), video size ranges from 50 MB up to 3 GB randomly.

3.3.4.2 Simulation Parameters Setting

For the data traces, we evaluate the base age and maximum age for the ABC from 10 seconds to 180 seconds. The simulation results show that when the base age is set to 10 seconds and the maximum age is set to 30 seconds, the approach can achieve the best performance. For RPC, a number of experiments are conducted by using different values for the root router's caching threshold, α and β, and found that when the root router's caching threshold is set to 1, α and β are also set to 1, the best performance can be achieved.

3.3.4.3 Simulation Results

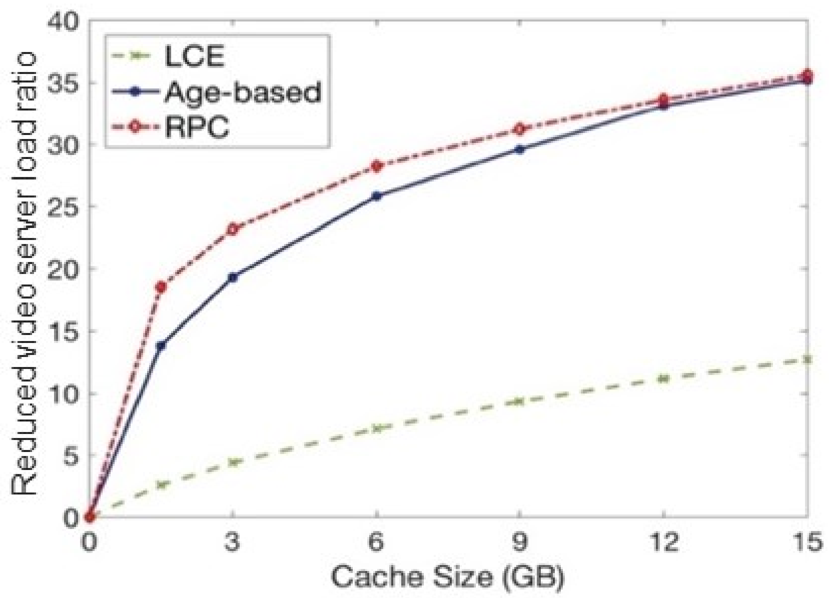


Fig. 3.3: Reduced video server load ratio VS cache size

Fig. 3.3 shows the reduced video server load ratio for different approaches. We can see that RPC outperforms ABC and LCE for the experiments. It is worth noticing that RPC has significant performance gain when the cache size is small. In reality, routers have limited storage and the size of video keeps growing, especially as HD-video (or even 4K videos) becomes more and more popular. Under this circumstance, we can see RPC can achieve evident benefits in reducing the publisher load compared with LCE and ABC.

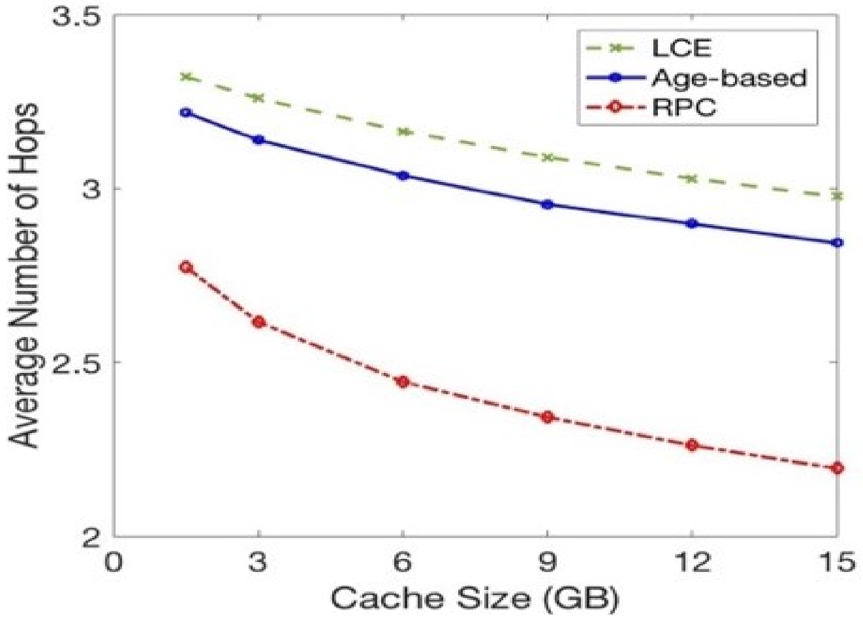


Fig. 3.4: Average number of hops VS cache size

Fig. 3.4 describes the trend for the average number of hops of LCE, ABC and RPC. Obviously, RPC has the smallest average number of hops out of these three algorithms despite the change of cache size. ABC slightly outperforms LCE with a reduction around 3% regardless of the cache size. In comparison RPC can reduce the average number of hops by 26.7% compared to ABC when the cache size is 15 GB. The reason for this phenomenon is that ABC is not sensitive user request rate, i.e., user may request a number of contents in a short time, but routers do not make any change to this burst request, they still replace contents only when they expire, which causes a higher number of hops. But RPC can handle a burst of requests effectively, since it relies on access count, which can adaptively cache new popular videos in time.

3.3.4.4 Parameter Configuration of Caching Threshold Decision Model

This section discusses how to configure parameters from equation (3.22). The caching threshold is determined by root router's pre-configured caching threshold $t_r$and router's topology level l_i, n and w are their weights. The smaller n is, the more important role l_i plays, which leads to a greater difference between t_i and t_{i-1}, i.e., the caching thresholds tend to be hierarchical, which suits contents following the Zipf–distribution. On the other hand, the smaller w is, the bigger impact t_r has, and the difference between t_i and t_{i-1} is smaller, i.e., the caching threshold tends to be uniform, which will perform better when contents are requested uniformly. t_r should be configured large enough to filter most unpopular contents, e.g., $t_r = 1$ for the data traces, as 90% of the videos are requested only once. Obviously, if n and w are set to 0, RPC will become to LCE and its performance will degrade to LCE's performance.

3.4 SDN-based Caching Approach

The proposed SDN-based caching decision policy for dynamic caching is presented in this section.

3.4.1 How SDN Can Improve the Caching Efficiency for ICN

In pure ICN, nodes make their caching decision locally, i.e., every node tries to cache the most popular video, which leads to high caching redundancy and low caching efficiency. Several cooperative caching schemes [3] [114] have been proposed to allow nodes to cooperate with each other to make the caching decision. However, in those schemes cooperation only happens among the node and its neighbors. As a result, the caching redundancy and efficiency can only be improved slightly, because nodes in those proposed schemes still lack the global view to make caching decisions.

Since SDN provides a global view, we can leverage this feature to make caching decisions from a global perspective. Generally, the more information we use, the more efficient the caching decision can be. For instance, nodes in a specific area will always cache the local popular videos if they only have the local information and make the caching decision locally. However, some local popular videos may not be popular in other parts of the whole network. Hence, caching those videos may obtain a lower caching efficiency for the whole network. If these nodes have a global view and are coordinated by the centralized SDN controller to make their caching decision, nodes can achieve better caching performance for the whole network.

In current ICN, the content transmission is inefficient since nodes need to broadcast interest packets to all of their neighbor nodes. This mechanism could cause significant overhead and delay, especially for dynamic networks. By exploiting the SDN capabilities, nodes can send interest packets to the SDN controller directly. The SDN controller is able to forward the interest packets to the best node based on the current status, such as the available bandwidth, the link latency, etc. Hence, the overhead caused by broadcasting interest packets in the current ICN can be significantly reduced.

3.4.2 How the proposed SDN-based caching decision policy works

The term "delivery cost" is used to represent the total delivery delay for delivering a video from the video server to all users at current time, which is denoted as $DC(v), v \in V$. Equation (3.25) is used to calculate $DC(v)$.

$$DC(v) = \sum_{n \in N} req_n(v) \cdot d_{0n} \tag{3.25}$$

The term "benefit", (denoted as $B_n(v)$), is used to indicate how much delay can be reduced by caching video $v \in V$ at node $n \in N$ locally. Since the video is cached locally, d_{mn} becomes to d_{nn}, then we can calculate $B_n(v)$ as follows:

$$B_n(v) = req_n(v) \cdot (d_{0n} - d_{nn}) \tag{3.26}$$

where, $req_n(v)$ represents the number of requests for video v from node n. d_{0n} represents the delivery delay from the video server to node n. Obviously, d_{nn} is 0 since the requested content is cached locally, there is no delay to fetch the content. Therefore, the reduced delay should be d_{0n}, and Equation (3.26) can be re-written as Equation (3.27):

$$B_n(v) = req_n(v) \cdot d_{0n} \tag{3.27}$$

In the proposed SDN-based caching decision policy, the centralized controller tracks both the global popularity and the local popularity of each video through receiving statistic information from all nodes periodically. The local video popularity information and the global video popularity information are stored in the local video popularity rank table (denoted as TL) and the global video popularity rank table (denoted as TG), respectively. Both of these two tables are key-value structured, in which the video name is stored as the key (character), and the popularity is stored as the value (integer). Generally, the average number of characters of video titles is not too big, e.g., it is only 16.7 for on our dataset. Considering one character is 1 byte, one integer variable is 4 bytes, each entry in TL and

TG only accounts 21 bytes in average, i.e., 1 MB memory can support 49000 entries at least. Since the global and local video popularity, the video delivery delay, and $req_n(v)$ are known, the controller can calculate $B_n(v)$ and can make the caching decision for each node in the network based on $req_n(v)$ and d_{0n}.

Algorithm 3.2 SDN-based caching decision policy

```
1: the controller checks the video cost rank table
2: if (the rank of videos in $Tc$ changes) then
3:     update and re-rank $TL$ and $TG$
4:     create $Td$
5:     find the 1st ranked video ($v_{\text{1st}}$) in $Td$
6:     decide which node should cache video $v_1$ in order
       to achieve the maximum benefit ($B_{n_{obj}}(v_{\text{1st}})$)
7:     update the delivery cost of video $v_{\text{1st}}$ in $Tc$ and $Td$
8:     re-rank $Td$
9:     send caching decision to router $n_{obj}$
10:    if $(\exists v \in V, DC(v) \neq 0) || (\exists n \in N,\ Z_n\ is\ not\ full)$
       then
11:        goto step 5
12:    else
13:        update $Tc$
14:    re-rank $Tc$
15: end if
```

The following steps illustrate how the proposed SDN-based caching decision policy works:

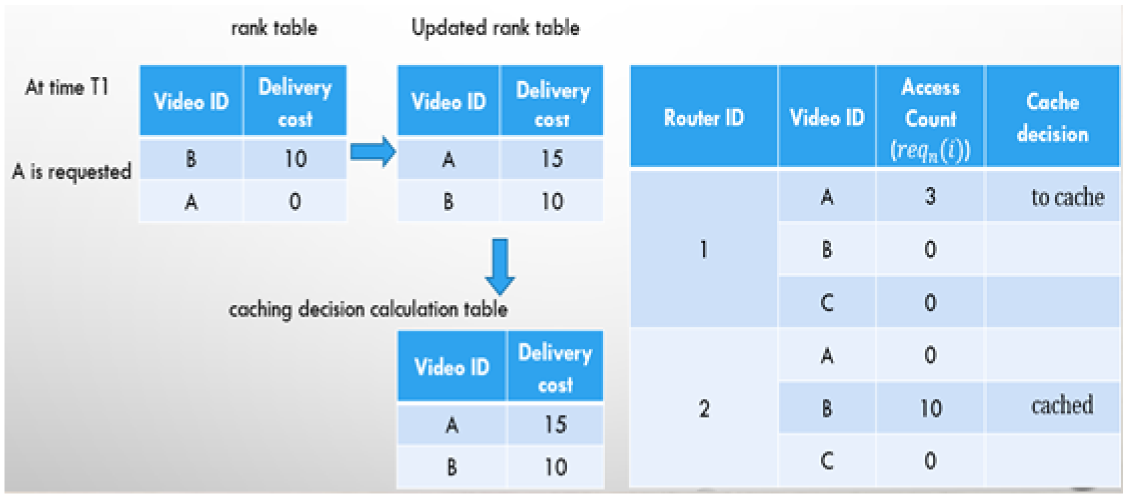
1. The controller maintains a video cost rank table (denoted as Tc) which is used to store the delivery cost ($DC(v)$) for each video from the entire network perspective.
2. The caching decision policy is triggered if the rank of any video cost in Tc changes. Then, a caching decision calculation table (a copy of the modified videos in the video cost rank table, denoted as Td) will be created temporarily to facilitate the caching decision making.
3. The controller always checks the 1st ranked video (the video that changes its rank) in the caching decision calculation table (Td) and marks it as v_{1st}. Then, the

controller determines which node (the objective node is marked as n_{obj}) should cache v_{1st} to realize the maximum benefit, i.e., $B_{n_{obj}}(v_{1st}) \geq B_n(v), \forall n \in N, \forall v \in V$. Then the controller sends the name of the video v_{1st} to node n_{obj} for caching, i.e., node n_{obj} will cache video v_{1st} once v_{1st} goes through n_{obj}.

4. Then, the controller updates $DC(v_{1st})$ in the caching decision calculation table; the updated value is denoted as $DC_{updated}(v_{1st})$, $DC_{updated}(v_{1st}) = DC(v_{1st}) - B_{n_{obj}}(v_{1st})$.
5. The controller re-ranks the caching calculation table, and goes to step 3. The calculation will terminate if (i) the cache size of each node (Z_n) is full; or (ii) in the caching decision calculation table, all $DC(v) = 0, \forall v \in V$.
6. Since the popularity of video changes over time, a former popular video may not be popular in the current time. To cache the recent popular videos, the video cost rank table is calculated periodically.

Via the above steps, the controller can perform the proposed caching decision policy as described in Algorithm 3.2.

Fig. 3.5 presents an example to show how the proposed SDN-based approach works. There are two routers (routers 1 and 2), three videos (A, B and C). For simplicity, this example uses the number of hops to indicate the delivery delay. The delivery delay from the video server to router 1 and router 2 is 5 hops and 1 hop respectively. At time T_1, video A is requested 3 times, therefore the rank table needs to be updated. The entry of delivery cost for video A is updated from 0 to 15, i.e., 3×5 according to Equation (3.27). Since the rank of video A in the rank table changes, the caching decision is triggered. A caching decision calculation table is created based on the updated rank table. The controller checks the 1st video (video A) in the caching decision calculation table, then decides to cache video A at router 1. Video B is already cached at router 2 from a previous iteration.



Video ID	Delivery cost
B	10
A	0

Video ID	Delivery cost
A	15
B	10

Video ID	Delivery cost
A	15
B	10

Router ID	Video ID	Access Count $(req_n(i))$	Cache decision
1	A	3	to cache
	B	0	
	C	0	
2	A	0	
	B	10	cached
	C	0	

Fig. 3.5: An example for proposed SDN-based approach

3.4.3 Responsibilities of the SDN Controller

The controller is the core of the proposed SDN-based caching decision policy. The main responsibilities of the controller are as follows:

- Making caching decision. The SDN-based caching decision policy is implemented in the controller so that it can make caching decision.
- Tracking video popularity from both local and global perspectives. Nodes in the network send their local information on video popularity to the controller periodically. Based on this information, the controller can maintain the local video popularity rank table TL and the global video popularity rank table TG. TL is used to find the objective node n_{obj}, TG is used to obtain the video cost rank table Tc.

- Determining where to forward requests (interest packets). If a node does not have the requested video, it will forward the interest packets to the controller. Then, the controller will decide how to forward the request based on the network status, e.g., the available bandwidth, the link latency, etc.

3.4.4 Computational Complexity

As mentioned in Section 3.2, the problems formulated in Equation (3.7) and (3.16) are 0-1 ILP problems and are NP-hard. Although exhaustive searching methods can be used to find a solution, the computational complexity of an exhaustive searching method is exponential [99], which is unacceptable for dynamic networks. Even though the branch and cut method or dynamic programming can be used to find an optimal solution for experiment purpose, they are not suitable for practical scenarios as they are still computationally intensive.

The computational complexity of the proposed SDN-based caching decision policy mainly depends on the sorting parts in Algorithm 3.2, i.e., steps 3, 6, 8 and 14. Since the controller updates all the rank tables each time the caching decision policy is triggered, all the rank tables are in a nearly sorted initial order, i.e., in each update, only a few videos' ranks are changed; most videos' rank remain unchanged. Therefore, insertion sort can be used to re-rank those tables. Consequently, the computational complexity of steps 3 and 14 is $O(|V|)$, and the computational complexity of step 8 is $O(X)$, where V is the set of videos and X is the number of videos in Td. Step 6 identifies the node n_{obj} that satisfies $B_{n_{obj}}(v_{1st}) \geq B_n(v), \forall n \in N, \forall v \in V$. The benefit can be easily calculated by (3.27), we only need to find the maximum benefit in step 6. Since the computational complexity of finding the maximum value in a given data set is $O(n)$, where n is the total number of the given videos, we can easily know that the computational complexity of step 6 is $O(|N|)$, where $|N|$ is the total number of nodes in the network. Moreover, step 6 is executed until the conditions in step 10 are not satisfied. Hence, the final computational complexity of step 6 should be $O(Y|N|)$, where Y is the number of iterations of step 10. Y could be $O(\max\{|V|, |N|\})$ in the worst case; therefore, the computational complexity of step 6 could be $O(\max\{|V||N|, |N|^2\})$. Obviously, the computational complexity of the proposed SDN-based caching decision policy is constituted by the computational complexity of steps 3, 6, 8 and 14, it is $O(|V| + \max\{|V||N|, |N|^2\} + X + |V|)$. Notably, the computational complexity of step 6 ($O(\max\{|V||N|, |N|^2\})$) is the dominant part, because it is exponential. Since the controller keeps track of table Tc, even a slight change can be detected by the controller, the value of Y would not be too big. What's more, with an increase in time, the video popularity tends to be stable and the cache of nodes becomes full eventually; therefore, the value of Y will become much smaller than the earlier time period. In fact, the practical computational complexity of the proposed scheme is much lower than $O(|V| + \max\{|V||N|, |N|^2\} + X + |V|)$; it could be $O(|V| + X + |N| + |V|) = O(2|V| + X + |N|)$ in the best case (when $Y = 1$, the computational complexity of step 6 is $O(|N|)$). Notably, the proposed scheme reduces the computational complexity significantly compared to the exhaustive searching method and branch and cut method, and it can be implemented in a real network to make dynamic caching decisions.

3.4.5 Performance Evaluation

In this section, the evaluation metrics are described firstly, then the simulation setting and results are presented.

As mentioned in Section 3.2.1, the ILP problem in Equation (3.17) is NP-hard, even the computational complexity of solving it by CPLEX is exponential, namely, $O(2^{2|V|\cdot 3|Req|\cdot 2|N|})$ [91], which can incur an incredibly long execution time when the topology is large. Hence, firstly, a small topology is used to establish a baseline comparison between the optimal solution using Equation (3.17) and the proposed SDN-based caching decision policy. Then, this thesis uses a real topology and chooses several practical caching decision policies, such as LCE [1], MPC [14] and RPC, to compare and evaluate the proposed SDN-based caching decision policy.

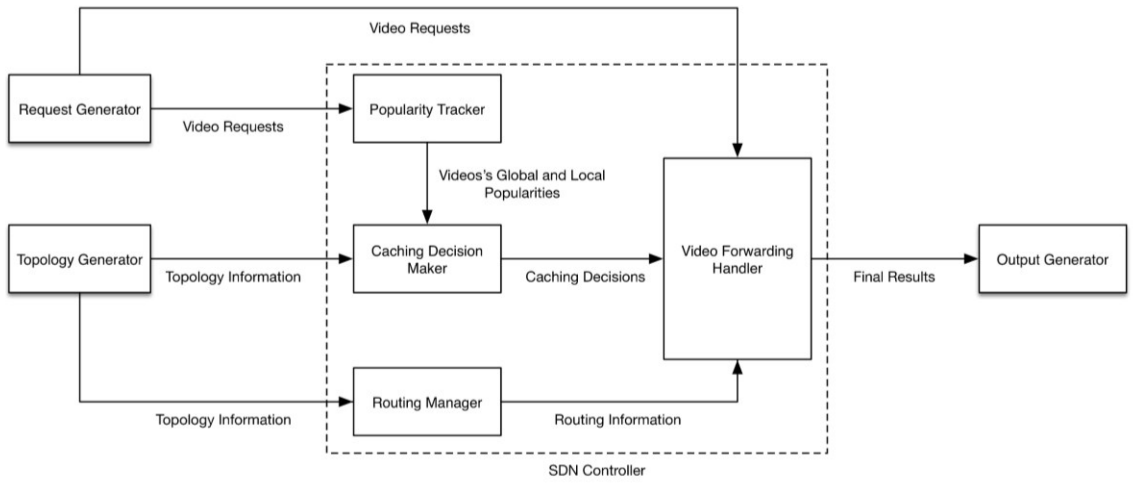


Fig. 3.6: Overview of the simulator

Since well-known simulators such as ndnSIM [66] and ccnSIM [26] do not support recent caching decision policies, including MPC and RPC, a simulator (written in C++) is developed to evaluate the performance of the policies used in this thesis. Fig. 3.6 shows an overview of our simulator. More specifically, we emulate the *SDN Controller* with a *Routing Manager*, a *Popularity Tracker*, a *Caching Decision Maker* and a *Video Forwarding Handler*. The *Request Generator* and the *Topology Generator* produce the video requests and the topology information, respectively. The *Routing Manager* offers a global view of the network status by maintaining a global table that contains the network status information (such as the shortest path for any given source-destination pair and the delivery cost for each delivery path). The *Popularity Tracker* can track each video's global and local popularities. Based on the caching decisions and the routing information, the *Video Forwarding Handler* can forward the video requests to the selected nearest node that has cached the requested video or to the content provider if the video is not cached in the network. Either the corresponding selected node or the remote video server will forward the requested video to the user or the destination node.

The *Caching Decision Maker* decides what video should be cached at which node based on the videos' popularity, and the topology information. In addition, we set a configurable parameter as the caching threshold for each node in the topology. By configuring the caching threshold and disabling the *Popularity Tracker*'s global popularity tracking function (i.e., it only tracks videos' local popularity), we can easily perform LCE, RPC and MPC policies. For instance, LCE can be realized by only setting the caching threshold to 0; RPC can be emulated by setting different threshold values to different nodes based on their topological information relative to the video server; and MPC can be conducted by setting the same threshold value (could be greater than 0) to each node.

If we set the caching threshold value to 0 and enable the *Popularity Tracker*'s global popularity tracking function (i.e. global and local popularities will be tracked), we can perform the proposed SDN-based approach. Finally, the *Output Generator* formulates and writes the simulation performance results to a file, which can be used to evaluate the caching performance. The simulations are executed on an 8-processor (Intel i7-4770) X86 desktop with 16 GB of RAM using Microsoft Windows 7 Enterprise edition.

3.4.5.1 Evaluation Metrics

Hit Ratio: The cache hit ratio is a key metric to evaluate the efficiency of a caching decision policy. Equation (3.28) shows how we calculate the hit ratio.

$$Hit\ Ratio = \frac{\sum_{t\in T}\sum_{v\in V}[req_n(v,t)\cdot c_n(v,t)]}{\sum_{t\in T}\sum_{v\in V} req_n((v,t)} \tag{3.28}$$

Average Number of Hops: This thesis uses the average number of hops (denoted as A) to indicate the average video retrieval latency. It can be calculated as follows:

$$A = \frac{\sum_{t\in T}\sum_{v\in V}\sum_{m\in N}\sum_{n\in N}[req_n(v,t)\cdot c_n(v,t)\cdot d_{mn}(v,t)]}{\sum_{t\in T}\sum_{v\in V} req_n((v,t)} \tag{3.29}$$

Number of Interest Packets: The number of interest packets (denoted as I) that are generated or broadcasted in the network is used to indicate the overhead. It can be calculated by Equation (3.30).

$$I = \sum_{t\in T}\sum_{v\in V}\sum_{n\in N} req_n((v,t) \tag{3.30}$$

3.4.5.2 Comparisons of the Optimal Solution and the Proposed SDN-Based Caching Decision Policy

This subsection evaluates the performance of the optimal solution and the proposed SDN-based caching decision policy in terms of the cache hit ratio and the average number of hops.

3.4.5.2.1 Simulation Setting

This thesis uses the topology illustrated in Fig. 3.7 for the simulation. There are five switches in the network, three of them at the bottom layer are edge switches which are directly connected to users. The other two switches in the middle layer are connected to the video server. A centralized controller is deployed to manage the network and perform the proposed SDN-based caching decision policy. In total, 120 videos are requested 1,000 times by the users, and their popularity distribution follows the Zipf–distribution [24], which is widely used in Video-on-Demand (VoD) systems [50] [58] [117]. Users retrieve videos through sending interest packets to the edge switches.

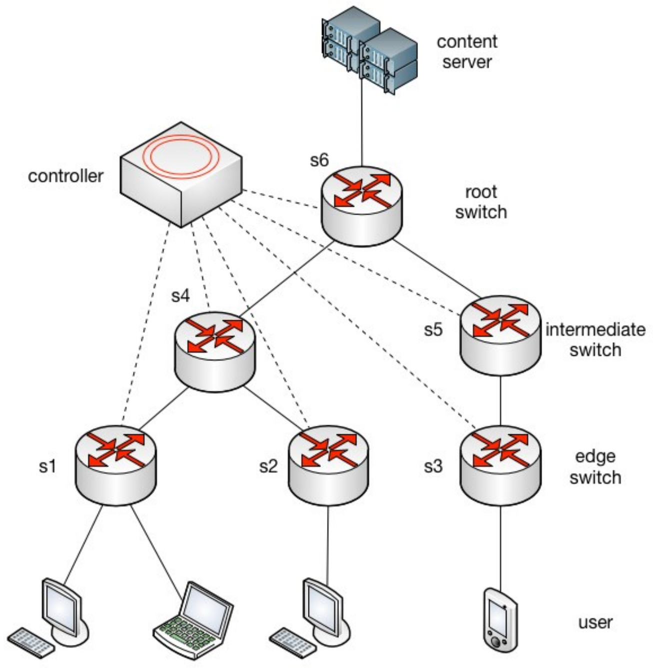


Fig. 3.7: System architecture

3.4.5.2.2 Simulation Results

Results shown in Fig. 3.8 compare the average number of hops between the optimal solution and the proposed SDN-based caching decision policy. As can be seen, the average number of hops decreases with the increase of the network cache size (the total size of all nodes in the network) for both of the optimal solution and the proposed SDN-based caching decision policy. The proposed SDN-based caching decision policy has a similar performance with the optimal solution. Moreover, the gap between the optimal solution and the SDN-based caching decision policy is much smaller when the network cache size ranges from 10% to 20%. In a real network, nodes have limited caching storage, which means that the network can only cache a small portion of the total videos [115]. Hence, the proposed SDN-based caching decision policy can achieve a high efficiency in terms of cache hit ratio when it is implemented in dynamic networks in which the requests and network states are changed frequently.

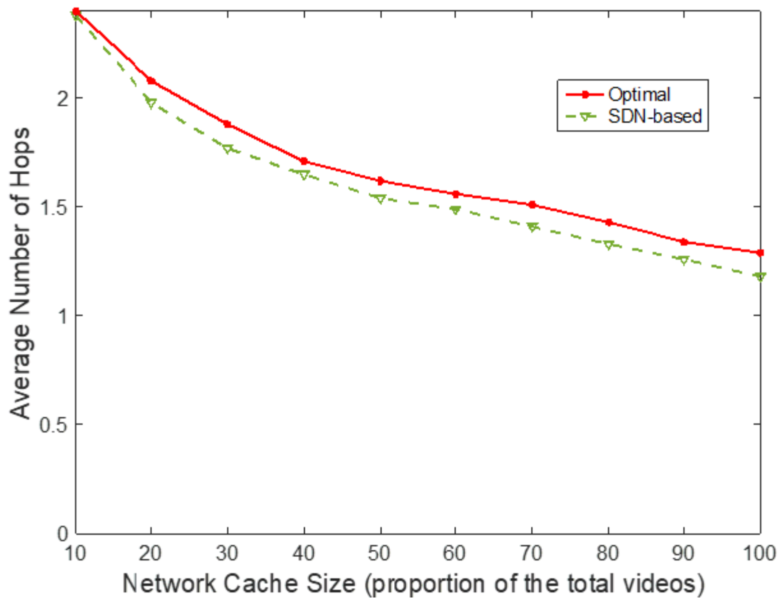


Fig. 3.8: Comparison between the optimal solution and the SDN-based caching decision policy in terms of the average number of hops

Fig. 3.9 describes the influence of the network cache size on the hit ratio. It is observed that the optimal solution and the proposed SDN-based caching decision policy have comparable performance in terms of hit ratio. Moreover, the proposed SDN-based caching decision policy even outperforms the optimal solution in terms of hit ratio at some particular cache size. The reason is that the objective of the optimal solution is to find the maximum gain which is mainly affected by the average number of hops. Moreover, the maximum gain only guarantees a minimum average number of hops instead of the hit ratio.

For example, if a user sends a request to switch s3, as depicted in Fig. 3.7, for a particular video which is cached at switch s1 only, it will travel 5 hops for this user to fetch the video from switch s1. But it only traverses 4 hops if the user fetches the video from the video server, but it will result in a miss which means the hit ratio is decreased.

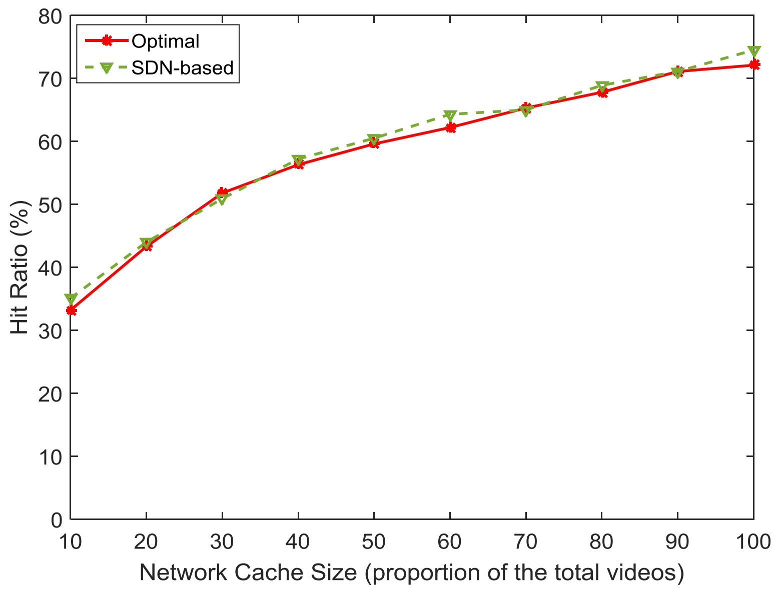


Fig. 3.9: Hit ratio of the optimal solution and the SDN-based caching decision policy for different network cache size

Because the computational complexity of finding the optimal solution is NP-hard, the execution time for realistic cache sizes is another important performance metric that needs to be evaluated. Fig. 3.10 shows the comparison between the actual execution time of the optimal solution and the proposed SDN-based caching decision policy versus the network cache size. Based on the curves in Fig. 3.10, the execution time of the proposed SDN-based policy is about 1 second, whereas the execution time required to find the optimal solution is more than 7 minutes even for a simple topology which is presented in Fig. 3.7. The computational complexity of the optimal solution is significantly higher than the computational complexity of the proposed SDN-based caching decision policy, and it is highly affected by the network cache size. Evidently, finding the optimal solution is not suitable for real networks, especially for large networks, whereas the proposed SDN-based caching decision policy is lightweight and can be implemented for dynamic scenarios for real time calculation.

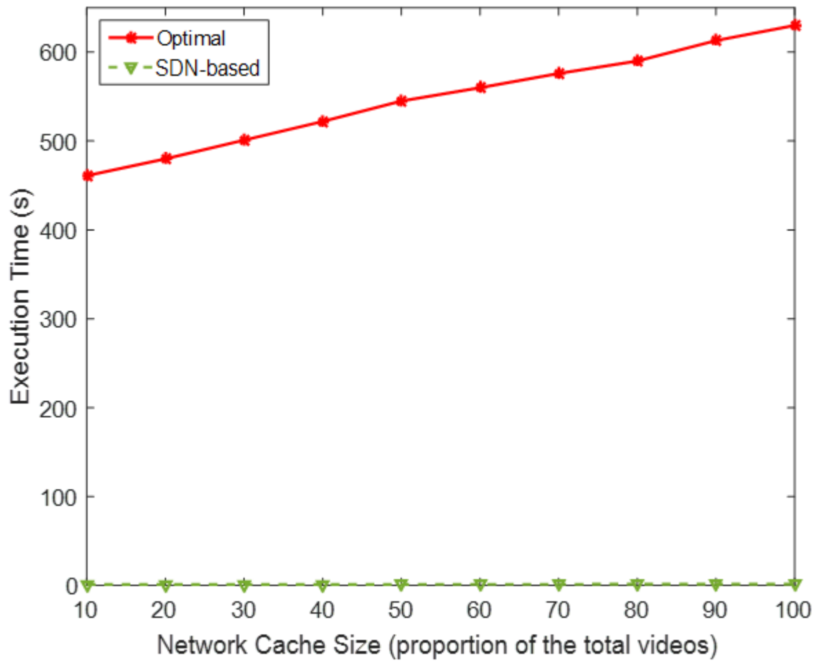


Fig. 3.10: Execution time of the optimal solution and the SDN-based caching decision policy for different network cache size

Based on the comparisons conducted above, we can see that the performance of the proposed SDN-based caching decision policy can approximate the performance of the optimal solution in terms of the hit ratio and the average number of hops, whereas the execution time of the proposed algorithm is significantly shorter.

3.4.5.3 Comparisons of the Proposed SDN-Based Caching Decision Policy and the Existing Caching Decision Policies

In this subsection, evaluations of the proposed SDN-based caching decision policy versus existing caching decision policies (i.e., LCE, MPC and RPC) are presented in terms of the hit ratio and the average number of hops.

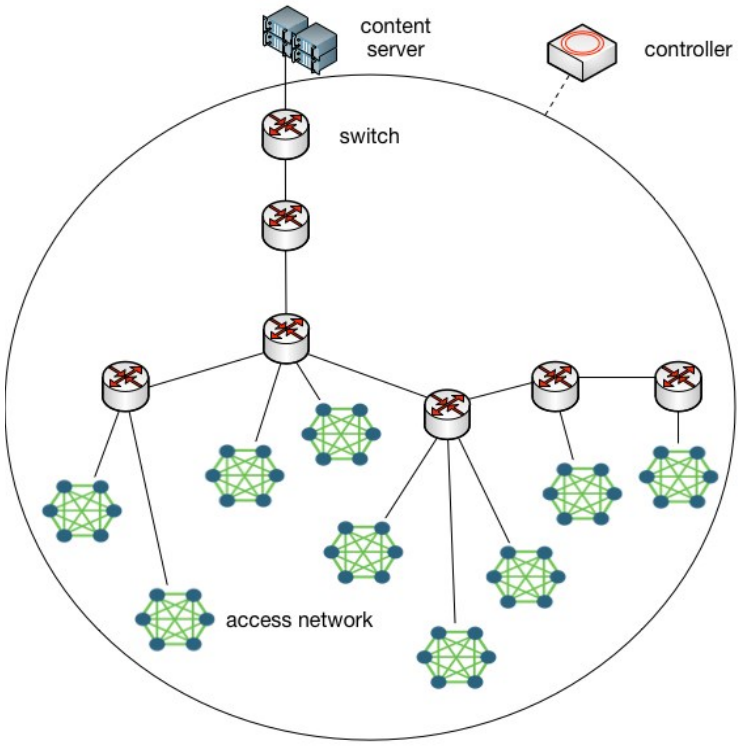


Fig. 3.11: Simulation topology

3.4.5.3.1 Simulation Setting

The Northern and the Eastern China regions of CERNET 2 [40], as shown in Fig. 3.11, are used as the simulation topology. This thesis assumes that there is a video server which is connected to the switch which is located in Beijing to provide video service for all users. Each switch represents an aggregate network device of its associated city. Users are connected to the access network directly and send requests to them to retrieve videos. As a result, Fig. 3.11 illustrates the logical view the network from the video server to other switches. There are 1,000 identical videos that are requested 10,000 times by all 2,500 users, and the popularity of videos follows the Zipf–distribution [24]. Since there is only one video server which sends video contents to all the users through the real topology, the video server becomes the root, and the network becomes a logical tree topology. In other words, you could have any physical topology, but when there is only one video server and it is responsible for sending contents to users, the topology for content delivery will become a tree topology. The concept is the same if there are multiple video servers.

Since the proposed SDN-based caching decision policy can be implemented in a real network for real time caching, this thesis choses the following practical caching decision policies as comparisons: LCE, MPC and RPC. LCE is the default caching decision policy of ICN. In LCE, videos are cached at each node along the video delivery path. Also, in comparison, MPC is selected as the non-cooperative caching decision policy, whereas RPC is selected as the cooperative caching decision policy. Both MPC and RPC use a caching threshold to filter the unpopular videos, i.e., a video can be cached at a node only if the number of requests of the video exceeds the caching threshold of the node. The only difference is that RPC takes the topology position of a node into consideration and assigns a pre-configured caching threshold value to each node based on their topology position, which leads RPC to be more efficient in caching.

3.4.5.3.2 Simulation Results

a) Description of the existing caching decision policies

Table 3.2: Optimal Parameter Settings of the Existing Caching Decision Policies

Caching Decision Policy	Key Parameters	Values
RPC	Threshold for the root router (t_r)	2
	Weight of the root router threshold (α)	1
	Weight of the topology level (β)	5
MPC	Caching threshold (TH)	20

Based on the performance results from Section 3.4.5.2.2, the proposed SDN-based caching decision policy has the potential to be implemented in a real network for real-time caching, this thesis compares it with the following practical caching decision policies: LCE, MPC and RPC. LCE is the default caching decision policy of ICN. In LCE, videos are cached at each node along the video delivery path as long as the cache size is large enough. For comparison, MPC is selected as the non-cooperative caching decision policy, whereas RPC is selected as the cooperative caching decision policy. Both MPC and RPC use a caching threshold to filter unpopular videos, namely, a video can be cached at a node only if the number of requests for the video exceeds the pre-configured caching threshold value of the node. The only difference is that RPC considers the topological position of a node and assigns a caching threshold value to each node based on its topological position (more details can be found in Section 3.3.3), which renders RPC more efficient in caching. For MPC, since all the nodes share the same caching threshold, the only key parameter is the caching threshold (TH). In contrast, RPC has three key parameters: the threshold for the root router (TH_r), the weight of the root router threshold (α), and the weight of the topology level (β). Through extensive experiments, the optimal parameter settings in terms of hit ratio and the average number of hops for RPC and MPC in the simulation topology are identified and listed in Table 3.2.

b) Impact of cache size

This part evaluates how cache size influences the performance of the proposed SDN-based caching decision policy and the compared policies.

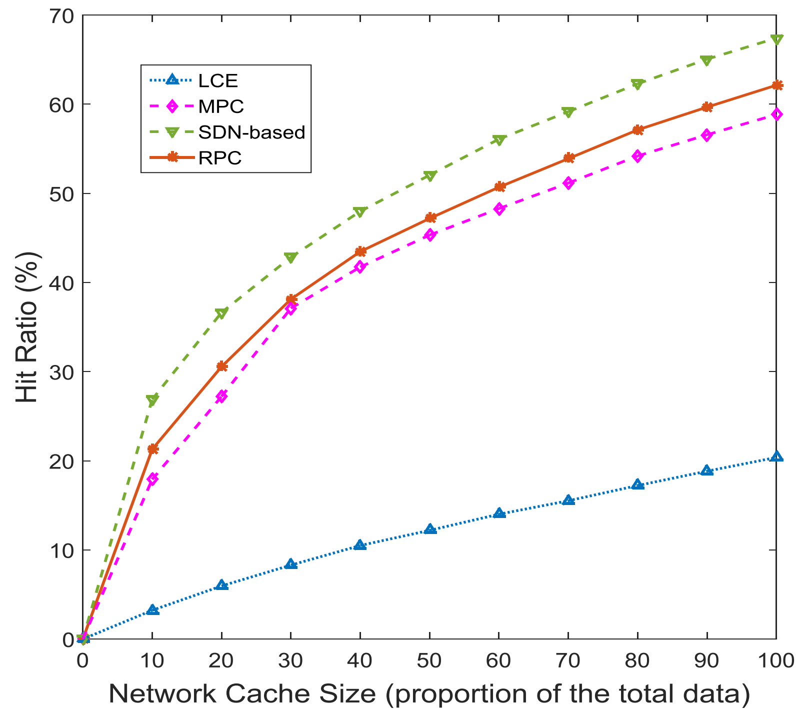


Fig. 3.12: Network cache size VS hit ratio

Fig. 3.12 demonstrates the impact of the cache size on the hit ratio. Notably, all these four policies can increase the hit ratio with the increase of the network cache size. We can see that the proposed SDN-based caching decision policy performs best among these policies regardless of the change of the network cache size. The poor performance of LCE is caused by its high caching redundancy, i.e., videos are cached at each node along the video delivery path. RPC and MPC are all popularity based, therefore they have a close performance. Since RPC considers the relative topology location of each node, it achieves a slightly better performance than MPC.

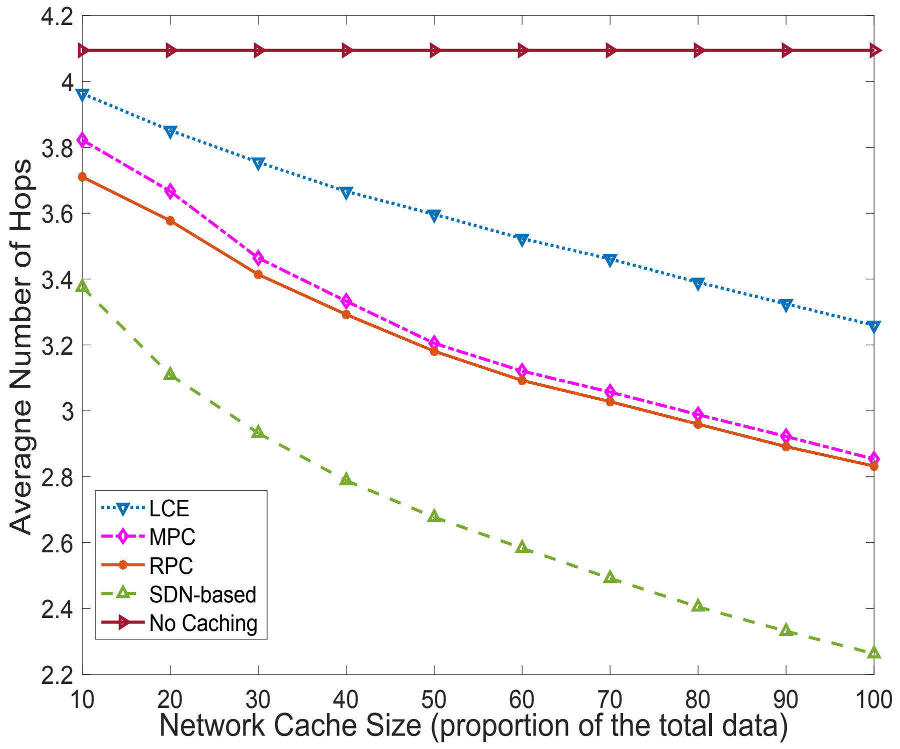


Fig. 3.13: Network cache size VS average number of hops

Fig. 3.13 shows the trend of the average number of hops when the network cache size ranges from 10% to 100% of the total video size. If no in-network caching is adopted, i.e., represented by the red top line, the average number of hops is about 4.1. As expected, all policies can reduce the number of hops with the increase of the network cache size. Specifically, the proposed SDN-based caching decision policy outperforms other policies regardless of the network cache size; it can achieve 17.3% and about 45% reduction in the average number of hops when the network cache size is 10% and 100%, respectively. Although the network cache size cannot be 100% in a real network, the proposed algorithm can still obtain a significant reduction in the average number of hops when the network cache size is less than 100%. It is also observed that RPC and MPC have similar performance in terms of the average number of hops, whereas LCE performs poorly compared to the other three caching decision policies.

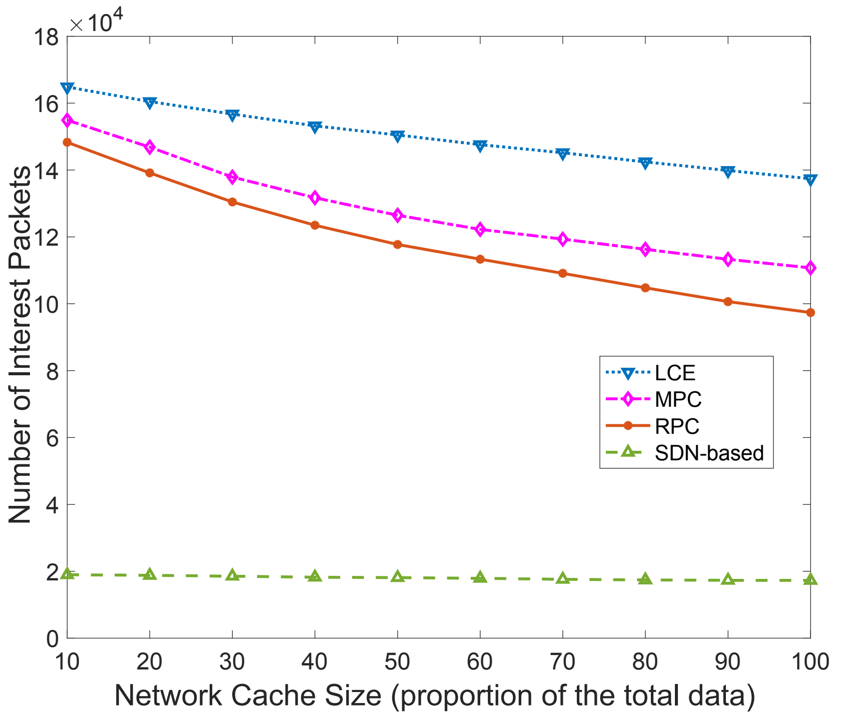


Fig. 3.14: Network cache size VS number of requests

The impacts of cache size on the number of interest packets are illustrated in Fig. 3.14. Obviously, the proposed SDN-based caching decision policy can significantly reduce the number of interest packets (from 18,987 to 17,279 when network cache size ranges from 10% to 100%), while other three policies need to generate or broadcast at least 97,378 interest packets. The reason behind this phenomenon is that in the current ICN, routers need to broadcast interest packets to their neighbors if the requested video is not cached locally. With the help of SDN, the forwarding policy of current ICN can be easily changed to let routers send requests to the SDN controller directly if the video is not cached locally. Since the SDN controller has all the information of the network, it can find the nearest node which has the requested video, and forward the interest packet to that node directly.

Therefore, the proposed SDN-based caching decision policy can significantly reduce the number of interest packets, i.e., the overhead. More specifically, the proposed SDN-based caching decision policy can reduce up to 954% the number of interest packets (from 162,148 to 18,987) compared to these three ICN caching decision policies.

c) Impact of the Exponent Parameter α of the Zipf-Distribution

The probability mass function (PMF) of the Zipf-distribution is described as follows:

$$f(\mathrm{x}) = \frac{1}{x^{\alpha} \sum_{i=1}^{n} (1/_{i})^{\alpha}} \qquad x = 1, 2, \dots, \mathrm{n}, \tag{3.31}$$

where α is the exponent parameter of the Zipf-distribution that reflects how skewed the popularity distribution is [67], and x is the rank of a video.

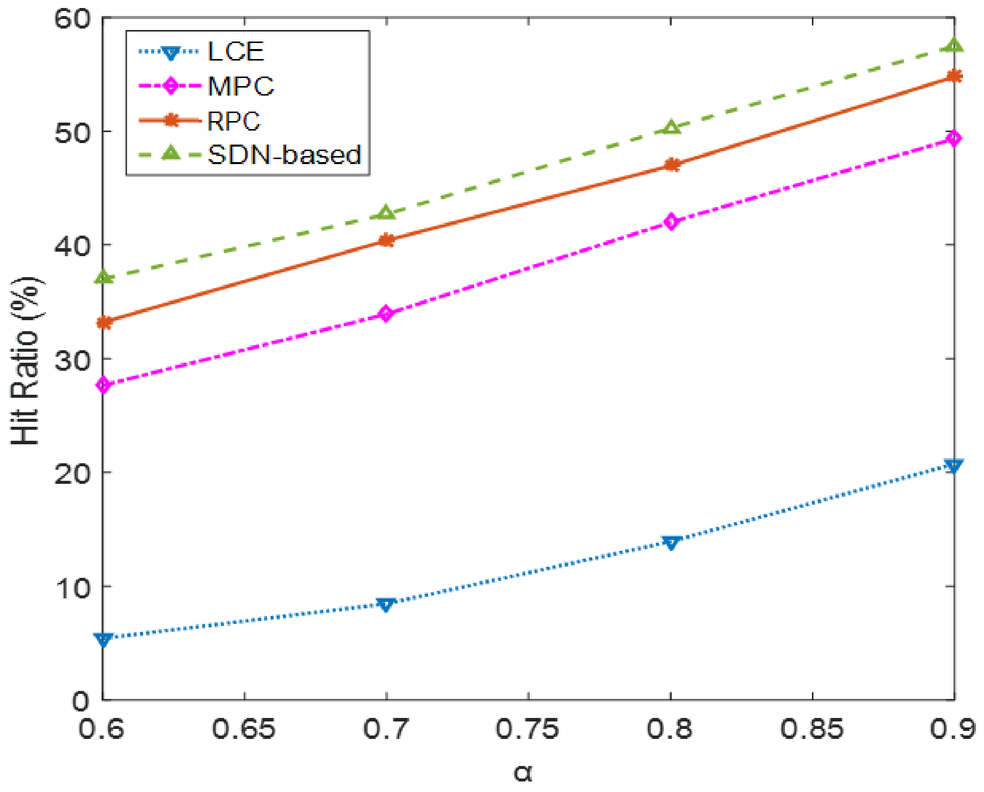


Fig. 3.15: Exponent parameter α VS hit ratio

As shown in Fig. 3.15, the proposed SDN-based caching decision policy has the best performance in terms of the hit ratio regardless of the value of α. It is also observed that the higher the value of α, the higher the hit ratio can be achieved for all these four caching decision policies. Because a higher α means a higher skewness of the distribution, i.e., a higher probability of requesting the top rank videos, which means that a higher popularity of these top rank videos. Hence, caching these top ranked videos can improve the hit ratio.

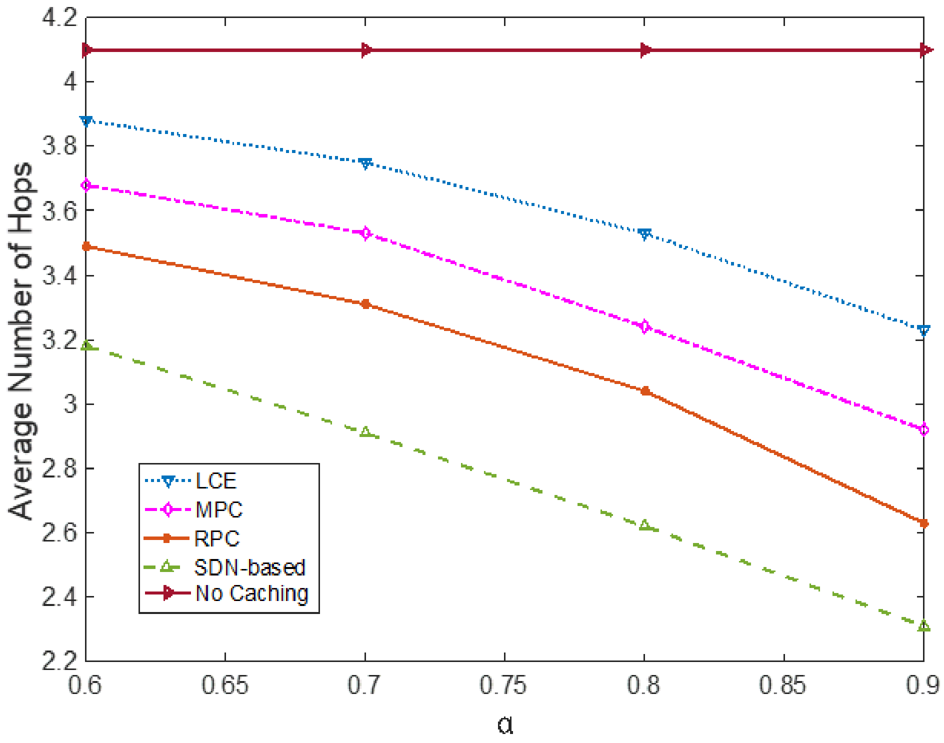


Fig. 3.16: Exponent parameter α VS the average number of hops

The results demonstrated in Fig. 3.16 illustrate how the exponent parameter α affect the average number of hops for these four caching decision policies and without caching. Similarly, the proposed SDN-based caching performs the best among all caching decision policies. Furthermore, with the increase of α, all caching decision policies can further reduce the average number of hops. The reason is the same as the one described above.

Through the curves, we can see that the exponent parameter α has a significant influence on the average number of hops for all these caching decision policies. For example, the proposed SDN-based caching decision can improve the reduction from 24% to 45% (from 4.2 hops to 3.19 hops and 2.31 hops respectively) compared to no caching when α changes from 0.6 to 0.9. Therefore, the proposed SDN-based caching decision policy can reduce the video delivery delay significantly, especially when the popular videos account for the majority of the traffic, i.e., a skewed Zipf–distribution.

d) Comparisons in Terms of Execution Time

The execution time per caching decision, denoted as $\overline{T_e}$, can be calculated as follows:

$$\overline{T_e} = \frac{Total\ execution\ time}{Total\ number\ of\ caching\ decisions} \tag{3.32}$$

Table 3.3: Comparisons in Terms of Execution Time

Caching Decision Policy	$\overline{\boldsymbol{T_e}}$
SDN-based	0.015 s
RPC	0.004 s
MPC	0.003 s
LCE	0.001 s

The execution time per caching decision for these four approaches is shown in Table 3.3. We can see that all existing practical approaches can make faster caching decisions compared to the proposed SDN-based caching decision policy. The reason is that in these three light-weight approaches (RPC, MPC and LCE), nodes make their own caching decision locally with light-weight computation, or even no computation. For example, there is no caching computation in LCE as nodes just cache everything that goes through them. The reason that the proposed SDN-based caching decision policy performs the worst in terms of $\overline{T_e}$ is because the SDN controller needs to calculate $B_n(v)$ (which is used to indicate how much delay can be reduced by caching video $v \in V$ at node $n \in N$ locally) based on the current popularity of videos, which involves more computation compared to the three light-weight approaches. However, the proposed caching decision will be triggered only if the rank of any video cost in Tc changes. Based on *Proposition 1*, the video popularity tends to be stable with the increase of time; hence, the caching decision will not be triggered frequently. Even if it is triggered, it only costs 0.15 s which does not make a big impact for caching compared to the length of a video which could be hours. Finally, considering that real networks always use powerful servers, $\overline{T_e}$ can be significantly reduced.

d) Comparisons in Terms of Simulation Time

In this part, the complexity of performing each caching decision policy is evaluated by comparing the number of caching decisions with respect to the simulation time.

According to Fig. 3.17, the proposed SDN-based caching decision policy makes fewer caching decisions as the simulation time increases. The reason is that the popularity of videos stabilizes over time. Hence, the caching decision policy is triggered less frequently with the proposed approach. However, RPC and MPC share a similar constant trend with respect to the simulation time. Since these two policies are threshold-based, the caching decision will be made once a video's popularity exceeds the caching threshold. LCE exhibits the worst performance across the four policies since it stores every requested video and the policy is triggered often due to the limited cache size.

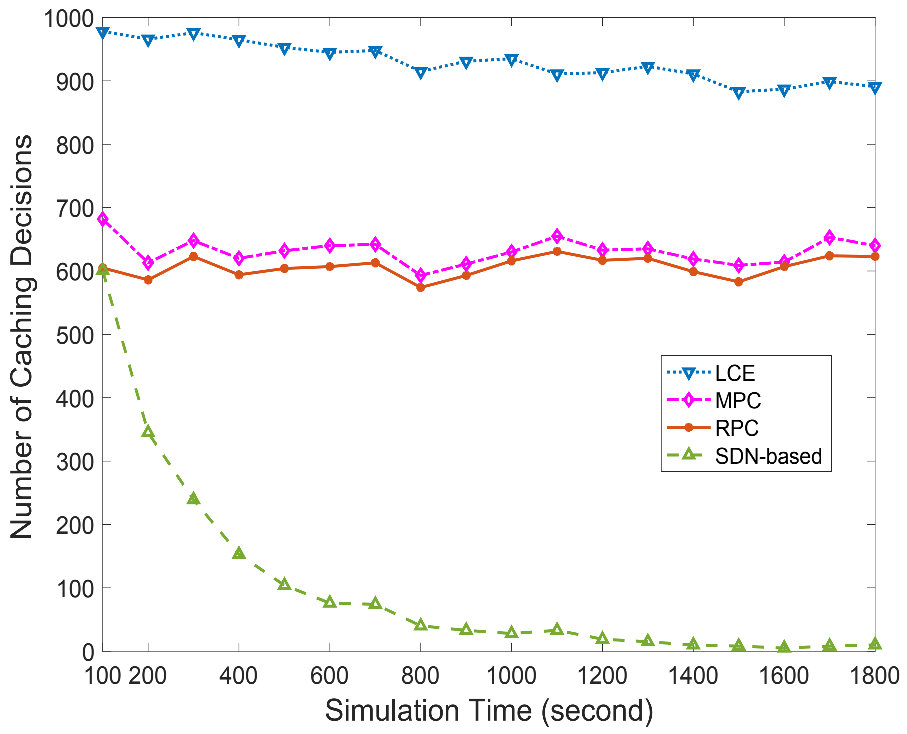


Fig. 3.17: Impacts of the simulation time on the number of caching decisions made

3.5 Summary

In this chapter, the caching problem was first formulated as a 0-1 ILP problem. Then, it was turned from a static ILP problem to a dynamic ILP problem by introducing the notion of time. However, the formulated ILP problem is NP-hard, which means it is impractical to find the optimal solution for a dynamic network. Hence, a light-weight router position-based cooperative caching (RPC) decision policy was proposed for the pure ICN. The proposed RPC approach works from a local perspective which may lead to a low efficiency of ICN in-network caching, hence SDN is a promising technique that can be leveraged to make the caching decision from a global perspective. This thesis leveraged the centralized control and the global view of SDN to design an SDN-based caching approach which considers both the video popularity and delivery delay. With the help of the centralized controller, the video popularity and delivery delay can be easily recorded and calculated. Based on that information, network nodes are coordinated by the controller to make their caching decision from a global perspective. Moreover, with the help of the controller, nodes do not need to broadcast interest packets any more for video transmissions, which can reduce the overhead significantly.

Chapter 4: Caching Approach for ICN-5G Networks

4.1 Introduction

The dramatic increasing demand for video from mobile users (MU) has imposed huge pressure on cellular networks. According to Cisco's VNI report [27], an estimated sevenfold increase in mobile data traffic will be reached by 2021. Among all forms of data traffic, the mobile video will account for 78 percent of the total mobile traffic by 2021. The mobile video has become a fundamental service for the wireless networks.

Millimeter wave (mmWave) is the key technique for 5G networks to overcome the bandwidth limitations of current wireless networks [32]. As the spectrum of mmWave is between 30GHz and 300GHz, it can allocate a huge amount of bandwidth to satisfy the dramatically increasing demands of mobile videos and multimedia services.

However, the limitation of mmWave is the short transmission range. If we take the propagation degradation into consideration, the transmission distance of mmWave is only 100 meters [32]. Therefore, mmWave BSs have to be deployed in small-cells. Consequently, high mobility users will suffer more frequent handoffs and shorter connection durations in 5G networks. Users with high mobility have to reconnect to the original content provider once a handoff happens [74], which induces heavy overheads and high retrieval delay. As a result, the QoE for mobile video users will be notably affected, and choppy playback might be caused. How to satisfy the QoE requirements for mobile video users with frequent handoffs in 5G networks has become a huge challenge.

This chapter combines ICN and 5G networks to propose the ICN-based caching approach for mobile videos in ICN-5G networks. Compared to existing IP-based approaches [42] [63] [84] [108] [114], videos can be cached at every node along the video delivery path in ICN. If a video is requested by many high mobility users, this video will be cached at the CR which is directly connected to the BS according to the proposed approach. In this way, when a mobile user enters a new cell, the mobile user does not need to reconnect to the original content provider to retrieve the video, he/she can fetch the video from the nearby CR by the name of the video. Hence, the retrieval delay can be significantly reduced and the QoE for users can be noticeably improved. However, which video should be cached at a CR is an important issue to improve the QoE for mobile video users and to reduce network traffic for 5G networks.

The main contributions of this chapter are as follows:

- This thesis proposes to integrate the features of ICN, such as in-network caching and name-based routing, into 5G networks to facilitate video delivery for MUs.
- The proposed caching approach takes both the mobility of users and the popularity of videos into consideration, which results in reduced retrieval delay and cache miss ratio.
- This thesis assumes the CRs which are directly connected to the BSs have the capability to cache videos. By caching popular videos for high mobility users at CRs, high mobility users can fetch the videos from CRs directly when a handoff occurs. As a consequence, QoE has been notably improved based on the simulation results.

The reminder of this chapter is organized as follows: Section 4.2 presents the proposed caching approach and the system model. Section 4.3 shows the experimental settings and simulation results. Finally, Section 4.4 summarizes this chapter.

4.2 System Model

In this section, the system model and the proposed caching approach for mobile videos in ICN-5G networks is described. The approach consists of a user mobility calculation model, a content popularity calculation model, and a caching decision model. Videos are cached at the chunk level. The architecture of the ICN-based caching approach is illustrated in Fig. 4.1. This thesis first uses a simple scenario to illustrate how the ICN-based caching approach works; then it details all those aforementioned models.

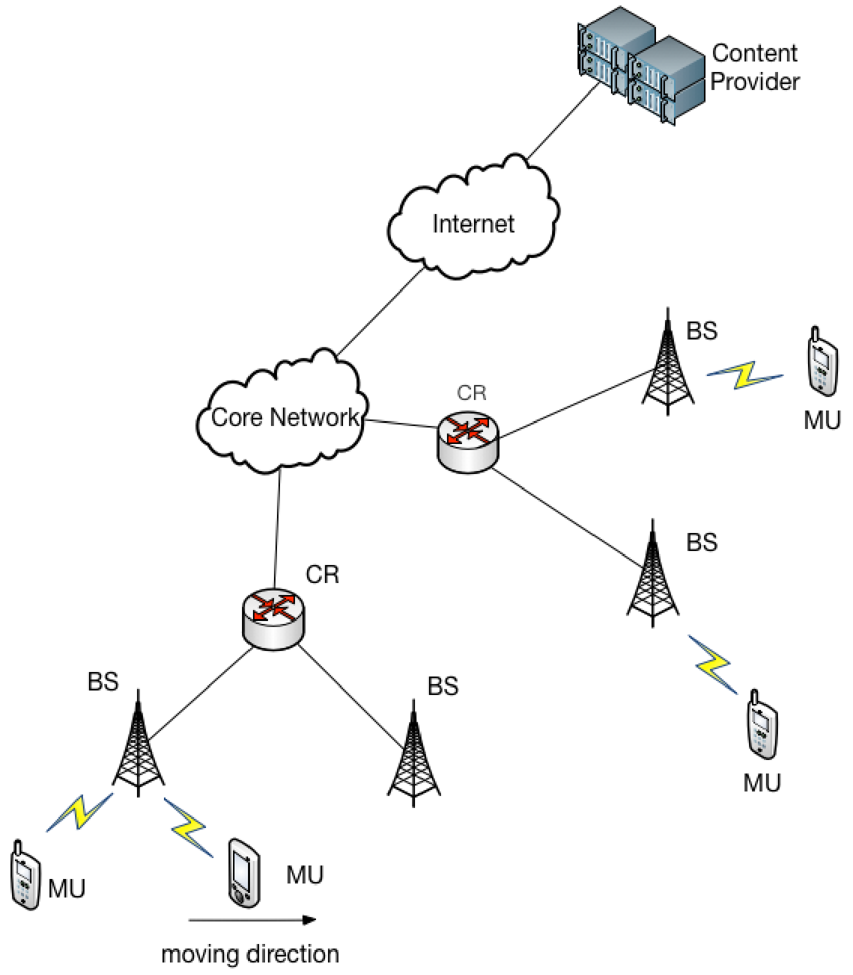


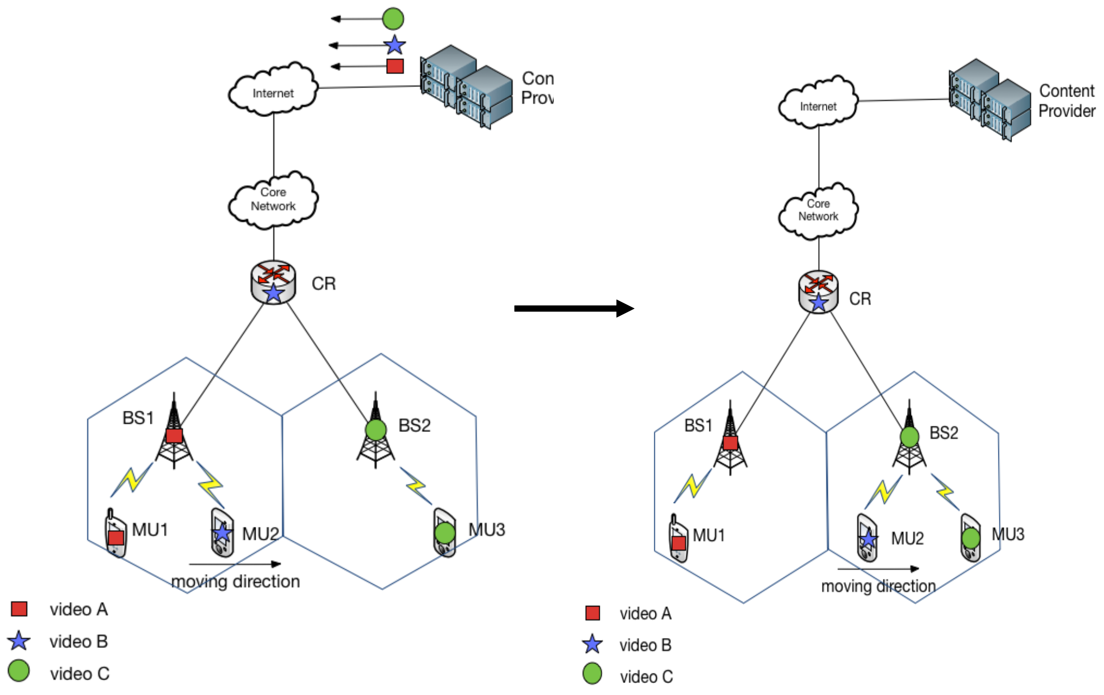
Fig. 4.1: System model

The main idea of the ICN-based caching approach is to cache videos requested by high mobility users at CRs in the RAN, and cache videos requested by low mobility user at BSs. Hence, low mobility users can fetch the popular video from a BS directly. If a handoff occurs, and the newly connected BS does not have the requested video, the high mobility users can fetch the video from a CR directly. By this means, there is no need to download the video from the remote content provider. Hence, the retrieval delay can be reduced. And the requirements of high QoE can be satisfied for both high mobility and low mobility users.

The proposed ICN-based caching approach works as follows: A mobile user sends an interest packet to the BS first. The BS retrieves the video name from the interest packet, and checks if it has the video. If the requested video is cached locally, the BS will send the video back to the mobile user directly; otherwise, it will forward the interest packet to the connected CR. Similarly, the CR checks its local memory first. If it has the requested video, it will send the video back to the user in reverse route of the interest packet deliver path; otherwise, it will forward this interest packet to its neighboring CRs. Its neighboring CRs repeat the same process. If no CR has this video, the interest packet will be forwarded to the content provider via the Internet.

4.2.1 A Simple Scenario

Fig. 4. 2 shows how the proposed ICN-based caching approach works with the help of a simple scenario. There are two BSs and both are connected to the same CR. Three users are requesting three different videos. Further, one user, MU2, is moving while the other two users are not (e.g., waiting for a bus). For simplicity, this thesis assumes the cache size of all BSs and CR is one, i.e., only one video can be cached for this simple scenario.



(a) (b)

Fig. 4.2: A simple scenario

As shown in Fig. 4.2 (a), video A is cached at BS1, video B is cached at CR and video C is cached at BS2. MU1 issues an interest packet for video A to BS1. BS1 checks if it has video A once it receives the interest packet. As BS1 has video A, BS1 sends video A back to MU1 directly. During this period, no traffic for video A is imposed at the core network, and the retrieval delay is low as the video is fetched from BS1 directly. Similarly, MU3 retrieves video C directly from BS2.

Assuming that MU2 is moving from BS1 to BS2 and the speed of MU2 is fast enough that he/she cannot finish watching video B before the handoff occurs. At the initial phase, MU2 sends an interest packet to BS1 for requesting video B. After receiving this interest packet, BS1 searches its cache memory and finds there is no video B. Then BS1 forwards this interest packet to CR. As video B is assumed to be cached at CR, MU2 can retrieve video B from CR. As MU2 keeps moving, he/she leaves the coverage area of BS1 and enters the coverage area of BS2, as shown in Fig. 4.2 (b). Hence, a handoff occurs for MU2. After the handoff is finished, MU2 sends another interest packet to BS2 for fetching video B. BS2 checks its local cache memory and cannot find video B; therefore, BS2 forwards the interest packet to CR. After the interest packet is received by CR, it checks its cache memory by video's name and finds out that video B is cached in its cache memory. Then, MU2 can retrieve video B from CR by video's name.

In the traditional IP network, if a handoff occurs, the user's device has to reconnect to the content provider, because the user only knows the IP address of content provider and has to rebuild a connection to the content provider first before recovering the video delivery. Even if a neighboring CR has the content, the user has no choice to fetch the video from them due to the principle of the host-centric IP network. However, in the proposed ICN-based approach, the user does not need to find out where the content is, the only thing the user needs to do is send an interest packet to nearby CRs, and then the nearest CR which has the content will send the content back to the user. In this way, the retrieval delay is reduced, and the QoE for mobile users is improved.

4.2.2 User Mobility Calculation Model

The user mobility calculation model is used to calculate the mobility of a mobile user and predict whether a handoff will occur during the playback period based on the location change information of the mobile user. As BSs in 5G networks can provide accurate location information with a deviation of one meter [30], the displacement of a mobile user can be precisely calculated by the change in position. The user mobility calculation model can benefit from the positioning technologies in 5G networks to calculate the speed of the user accurately.

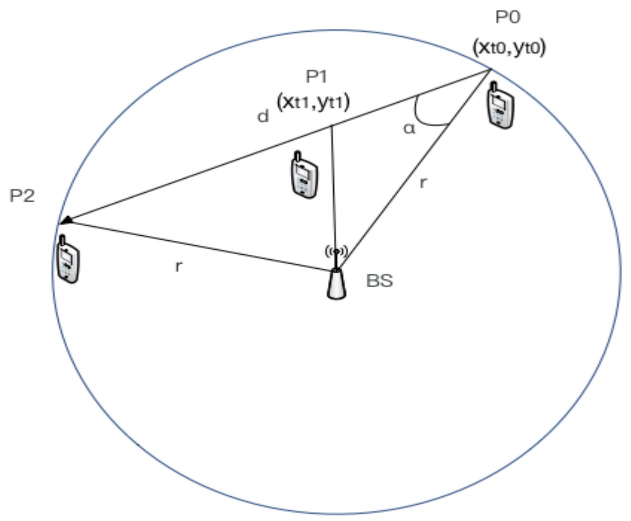


Fig. 4.3: An illustration for user mobility calculation

Table 4.1: Symbols Used for the Mobility Calculation Model

Symbol	**Definition**
S_i	Size of video i
R_i	Bit rate of video i
α	Angle between user moving direction and the BS
r	Radius of the BS coverage
d	Distance that a user moves in the coverage area of the BS
d_{t1-t0}	Distance moved by a mobile user between time $t0$ and $t1$
d_{t0-bs}	Distance between a mobile user and the BS at time $t0$
d_{t1-bs}	Distance between a mobile user and the BS at time $t1$

This thesis assumes the coverage area of the BS is a circle, P_0 is the position when user enters the coverage area of BS, P_1 is the position of user at time t_1, P_2 is the predicted position when user is about to leave the coverage area of BS, as shown in Fig. 4.3. Table 4.1 presents the important symbols and their definition. As the transmission distance of BS is around 100 meters [32], this thesis assumes users do not change their moving direction in a cell. Therefore, by calculating the change in position within a period of time, the user's speed can be calculated. Taking the duration of the video and the user's speed into consideration, a handoff can be predicted by (4.1):

$$H = \frac{\frac{S_i}{R_i} \times \frac{d_{t1-t0}}{t1-t0}}{d} \tag{4.1}$$

$$cos\alpha = \frac{d_{t1-t0}^2 + d_{t0-bs}^2 - d_{t-bs}^2}{d_{t1-t0} \times d_{t0-bs}} \tag{4.2}$$

$$d = 2r \cdot cos\alpha \tag{4.3}$$

$$d_{t1-t0} = \sqrt{(x_{t1} - x_{t0})^2 + (y_{t1} - y_{t0})^2} \tag{4.4}$$

$$d_{t0-bs} = \sqrt{(x_{t0} - x_{bs})^2 + (y_{t0} - y_{bs})^2} \tag{4.5}$$

$$d_{1t-bs} = \sqrt{(x_{t1} - x_{bs})^2 + (y_{t1} - y_{bs})^2} \tag{4.6}$$

where the first part of the numerator (S_i/R_i) in equation (1) depicts the time it takes to watch a particular video and the second part ($d_{t1-t0}/(t1 - t0)$) is the mobile user speed. If $H \geq 1$, a handoff will occur; otherwise, no handoff will occur during the playback time. The calculation result is an input of the caching decision model. Based on the distance between user and the BS, and the change of user position, the user moving direction can be calculated by equation (4.2). Then the distance (d) that the user will move until the user leaves the coverage area of the BS can be calculated by equation (4.3). Equation (4.4), (4.5) and (4.6) are used to calculate d_{t1-t0}, d_{t0-bs} and d_{1t-bs} respectively. The position of a mobile user at time $t_0 + t_1$is denoted by (x_{t0+t1}, y_{t0+t1}), where (x_{t0}, y_{t0}) is the position of the mobile user at time t_0, and (x_{bs}, y_{bs})is the coordinate of the BS.

4.2.3 Content Popularity Calculation Model

In this chapter, each node tracks the number of requests of each video locally by the video's name and stores them as a key-value structure (key: video name; value: access counts) into a popular video table. An example is shown in Table 4.2. The number of requests for each video indicates the popularity of the video. A popularity threshold is assigned to each node. Once a video's access count reaches the popularity threshold, this video is tagged as a popular video in the popular video table, 1 indicates popular, 0 indicates unpopular. This information of the popular video table is another input of the caching decision model.

Table 4.2: Popular Video Table

Video name	**Number of requests**	**Popular or not**
Video A	100	1
Video B	5	0
…	…	…

As the popularity of video generally decreases over time, a former popular video may not be popular at the current time. If a node still regards it as a popular video and stores it at the local cache memory, the limited cache memory cannot store recent popular videos; hence, the cache space is wasted. In order to prevent this phenomenon from happening, a reset value is configured to reinitialize the number of requests in the popular video table. In other words, if the reset value is reached, all the information will be reinitialized in the popular video table.

4.2.4 Caching Decision Model

The principle of the caching decision model is described as follows:

- CRs cache videos which are requested by high mobility users only.
- BSs cache local popular videos for users with low mobility.
- Least Recently Used (LRU) is used in this chapter for the replacement policy. However, LRU can be replaced by other replacement policies, such as Least Frequently Used (LFU).

```
Algorithm 4.1: Caching decision model
Input: 1) The number of requests for a video;
2) The mobility of a user
popularity threshold = a pre-configured value
1:  a video request arrives at the node
2:  increment the number of requests of the video
3:  checks if it has this video
4:  if (content cached) then
5:      send the video back to user
6:  else if (enough free space) then
7:      cache the video
8:      forward the video
9:  else if (the node is a CR) then
10:     if (high mobility user) then
12:         if (No. of requests ≥ popularity threshold)
13:            then while (not enough space) do
14:                delete least requested replica
15:             end while
16:            cache the video
17:         else
18:            forward the video only
19:     else
20:         forward the video only
21:  else if (the node is a BS) then
22:     if (No. of requests ≥ popularity threshold)
23:         then while (not enough space) do
24:             delete least requested replica
25:         end while
26:         cache the video
27:     else
28:         forward the video only
29:  end if
```

The caching decision model is used to decide which video should be cached, and the replacement policy is used to decide which video should be replaced if there is not enough space to cache new video. The models is designed as follows:

1. Each node (BS or CR) first checks its cache storage. If there is enough space to cache the video, it caches the video directly.
2. If there is not enough space, the information of the user mobility is needed. If the user moves with a high speed, i.e. a handoff will happen during the session, the video will be cached at the CR when the number of requests for the video (stored in the popular video table) reaches the popularity threshold of the CR.
3. Otherwise (i.e. no handoff happens), the video will be cached at the BS if the number of requests of the video exceeds the BS's popularity threshold. In this way, the delay induced by frequent handoffs and the core network traffic can be reduced. Therefore, the user can have better QoE for watching video in 5G networks. The whole procedure to make caching decision is described in Algorithm 4.1.

4.3 Performance Evaluation

This section presents the simulation environment and results. As the proposed caching approach is ICN-based, this thesis choses IP address-based RAN caching [100] (cache contents at BSs) for a comparison, and apply LRU as replacement policy for the ICN-based caching approach and RAN caching. Since the well-known simulators such as ccnSim [26] and ndnSIM [66] do not support 5G networks, a custom-built simulator (written in C++) is used to evaluate the performance of the proposed ICN-based caching approach. This thesis performed the simulation 10 times for each experiment and calculated the average of those 10 runs.

4.3.1 Evaluation Metrics

Average retrieval delay: The average retrieval delay is used to evaluate how fast a user can fetch the requested video. If the average retrieval delay is high, QoE will be significantly affected and a lot of packets may be lost and retransmissions will be triggered, which leads to choppy playback. The average retrieval delay can be calculated by equation (4.7).

$$\overline{D} = \frac{\sum d_i}{|Req|} \tag{4.7}$$

where the average retrieval delay is denoted as $\overline{D}$, d_i is the retrieval delay of video i, and $|Req|$ is the total number of requests.

Average miss ratio: This thesis evaluates the efficiency of the proposed ICN-based caching approach by measuring the miss ratio which can be calculated by equation (4.8). The average miss ratio shows the efficiency of the ICN-based caching approach from the macroscopic viewpoint.

$$\overline{M} = \frac{\sum m_i}{|Req|} \tag{4.8}$$

$$m_i = \begin{cases} 1, & if\ video\ i\ is\ not\ cached \\ 0, & if\ video\ i\ is\ cached \end{cases} \tag{4.9}$$

where the average miss ratio is denoted by $\overline{M}$, m_i is a binary value, $m_i = 0$ if video i is cached at a BS or a CR locally for a request; otherwise, $m_i = 1$.

Average number of choppy playback: The average number of choppy playback is used to indicate the QoE for mobile users. The number of choppy playbacks is recorded each time a handoff occurs. Each handoff will lead to a retransmission due to packets lost if the newly connected BS or CR does not cache the requested video. The average number of choppy playback can be calculated by equation (4.10)

$$\bar{C} = \frac{\sum c_i}{|Req|} \tag{4.10}$$

where $\bar{C}$ represents the average number of choppy playback, c_i is the number of choppy playback for video i.

4.3.2 Evaluation Settings

This section presents the evaluation settings for the proposed approach.

4.3.2.1 Topology and Input Data

This thesis implements 20 BSs and 6 CRs in a $500\times500\ m^2$ regular grid. Each BS is connected to a CR. As shown in Fig. 4.4, a CR connects at least two BSs. The max number of connected BSs for a CR is 5. CRs connect to their neighbor CRs via optical fiber.

There are 1,000 users who keep requesting videos until they walk out of the grid. A random mobility module is used for user mobility, each user moves with a random speed ranging from $0\ m/s$ to $20\ m/s$ and in a random direction. It indicates that the user is stationary when the speed is $0\ m/s$, while the user may be driving when the speed is $20\ m/s$. The total number of video requests is 10,000, and there are 2,598 identical videos. The popularity of videos follows the Zipf-distribution with $\alpha = 0.7$, which has been widely used in video streaming systems [19].

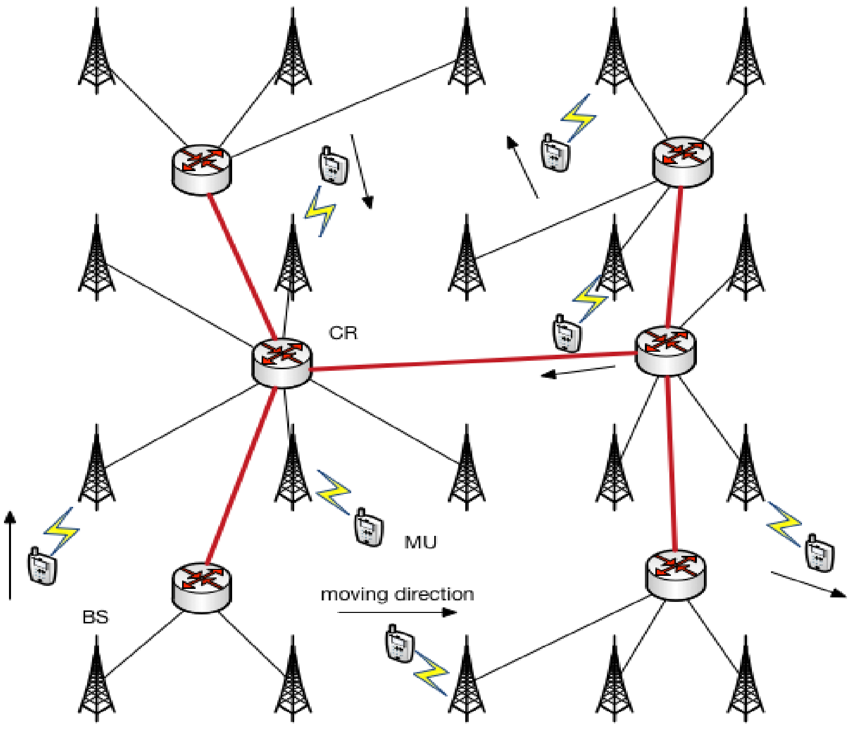


Fig. 4.4: Topology for evaluation

4.3.2.2 Parameter Setup

This thesis assumes the wireless link capacities are equally shared among mobile users. As the bandwidth provided by mmWave technology is significantly high, the latency from a BS to a mobile user is set to 5 ms [13]. The latency from a CR to a BS is assigned to be 10 ms [96], while the latency from the content provider to a CR is assigned to be 50 ms [6]. Videos are divided into chunks. For simplicity, this thesis assumes all chunks have the same size (4,000 bytes), and the size of videos ranges from 30 MB to 3 GB. In addition, CRs are assumed that can be implemented with larger storage memory than BSs. The shortest path routing protocol is adopted for the simulation. The parameter setup is summarized in Table 4.3.

Table 4.3: Parameters Setup

Description	**Value**
BS layout	Regular grid $500 \times 500\ m^2$[73]
Radius of BS coverage	100 m
BS – Mobile user latency	5 ms
CR – BS latency	10 ms
CR – Content provider latency	50 ms
Video chunk size	4000 bytes
Number of BSs	20
Number of CRs	6
Range of video size	30 MB~3 GB
Video popularity distribution	Zipf
Range of BS cache size	1, 2, 3, …, 10, 15, 20, 25, 30 GB
Range of CR cache size	1, 15, 30, 60, 100 GB
Reset value of popular video table	30 $mins$

4.3.2.3 Simulation Results

This part evaluates how cache size influences the performance of the ICN-based caching approach. RAN caching is selected as a comparison.

Fig. 4.5 shows the trend of the average retrieval delay when the BS cache size ranges from 1 GB to 30 GB and the CR cache size is set as shown in Table 4.3. Retrieval delay is a key factor for QoE. We can see both RAN caching and the ICN-based caching approach can reduce the average retrieval delay with the increase of the BS cache size. However, the ICN-based caching approach outperforms RAN caching by more than 13 ms deduction in average retrieval delay (i.e., the minimal gain is from 44 ms to 31 ms with 1 GB BS and

CR cache size). As videos can only be cached at BSs not at routers in RAN caching, therefore, the change of router cache size has no influence on the performance of RAN caching in terms of the average retrieval delay. For the ICN-based caching, the average retrieval delay can be reduced significantly with the increase of the CR cache size.

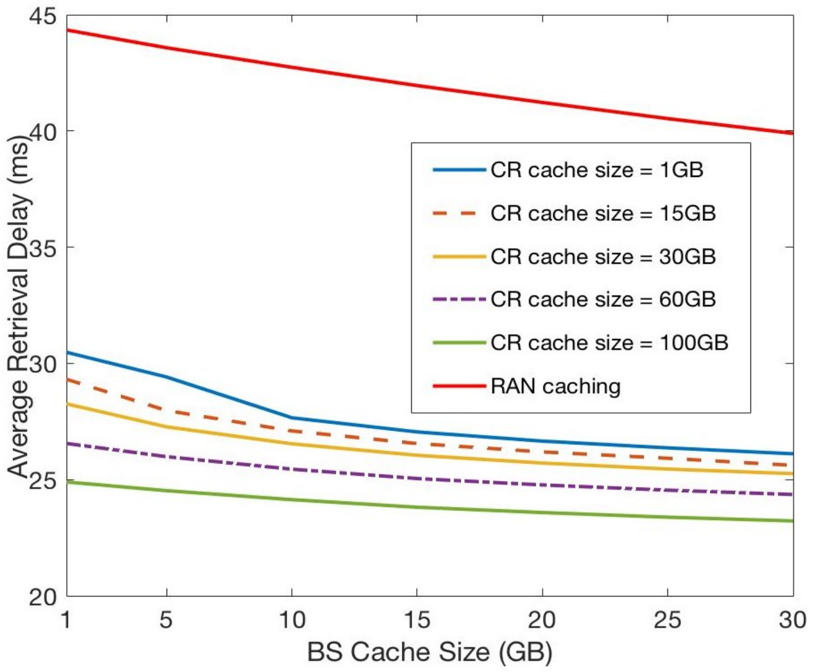


Fig. 4.5: Impact of cache size on average retrieval delay

However, the benefits from the increase of the BS cache size decrease when the CR cache size increases. The reason is that a user still needs to reconnect to the remote content provider once a handoff occurs for RAN caching, while a user can fetch the video from the CR by name in the proposed ICN-based caching. On the other hand, increasing the BS cache size may not reduce the retrieval delay caused by handoffs considerably. As shown in Fig. 4.5, when the CR cache size is set to 100 GB, the average retrieval delay decreases from 25 ms to 23 ms with the increase of the BS cache size from 1GB to 30 GB. When the CR cache size is set to 1 GB, the average retrieval delay decreases from 31 ms to 26 ms with the increase of BS cache size from 1GB to 30 GB.

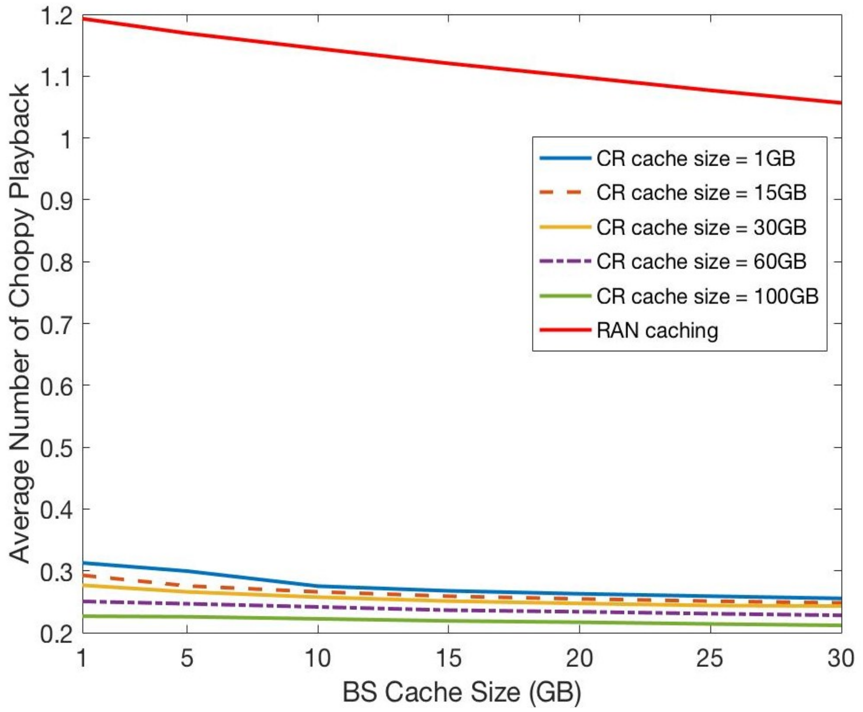


Fig. 4.6: Impact of cache size on average number of choppy playback

Fig. 4.6 illustrates the impact of the cache size on the average number of choppy playback. The ICN-based caching approach reduces the average number of choppy playback significantly compared to RAN caching. Specifically, the average number of choppy playback is below 0.31 for the proposed caching approach for various cache sizes, while the average number of choppy playback of RAN caching is greater than 1. For the proposed caching approach, the BS cash size does not have much impact on the average choppy playback, especially when the CR has adequate cache size. The reason behind this phenomenon is that the CR plays a vital role in reducing the retrieval delay and packet loss for frequent handoffs. Hence, no significant advantage for the average number of choppy playback is achieved by increasing the BS cache size, especially when the CR has adequate cache memory. Therefore, the proposed caching approach can improve the QoE significantly for mobile users even with a small cache size.

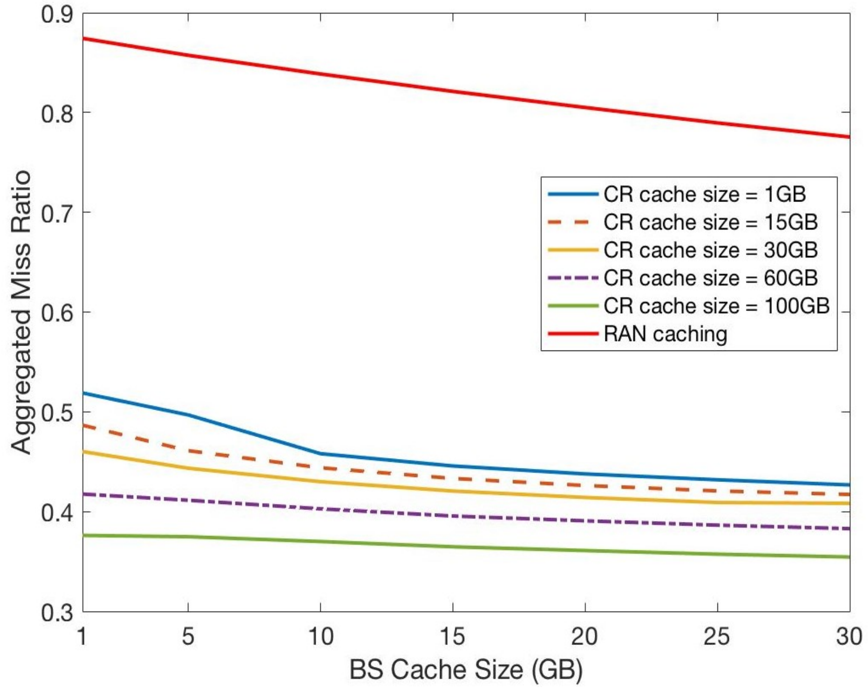


Fig. 4.7: Impact of cache size on average miss ratio

The impact of the cache size on the average miss ratio is shown in Fig. 4.7. Similar to Fig. 4.5, the average miss ratio decreases when the BS cache size increases. It is worth noticing that the ICN-based caching approach reduces the average miss ratio significantly compared to RAN caching. When the CR cache size is set to 1 GB, the average miss ratio of ICN-based caching ranges from 42% to 52%, whereas the average miss ratio of RAN caching ranges from 77% to 87% when the BS cache size ranges from 1 GB to 30 GB.

In terms of the effect of the cache size for the ICN-based approach, it can be seen that the ICN-based caching achieves noticeable benefits from the increase of the CR cache size, especially when the BS cache size is small. The average miss ratio decreases from 52% to

37% for various CR cache sizes when the BS cache size is 1 GB. Even if the gain decreases with the increase of BS cache size, but about 10% improvement can still be achieved for various BS cache sizes.

In conclusion, the proposed ICN-based caching approach outperforms RAN caching in terms of the average retrieval delay, the average number of choppy playback and the average miss ratio. The BS cache size has limited impact on the performance of the ICN-based caching approach when the CR has adequate cache memory. Slight benefits can be achieved in reducing the average number of choppy playback by increasing the CR cache size, while noticeable benefits can be achieved in reducing the average retrieval delay and average miss ratio by increasing the CR cache size.

4.4 Summary

This chapter proposed an ICN-based caching approach for videos in 5G networks. Both the video popularity and user mobility are considered to reduce the retrieval delay and core network traffic. Caching videos that are requested by high mobility users can significantly reduce the retrieval delay which is caused due to the frequent handoffs in 5G networks. This thesis performed extensive evaluations, and the simulation results show that the proposed ICN-based caching approach is more efficient in reducing the average retrieval delay, the average number of choppy playback and average miss ratio significantly compared to RAN caching which is widely used in 5G networks.

Chapter 5: Caching Approach for ICN-IoT Networks

5.1 Introduction

The continuous development of networking technologies and smart devices has led Internet of things (IoT) to be growing at an unprecedented pace. It is reported that billions of devices will be connected to the Internet over the next 5 years, which will lead the current IP-based Internet to facing tremendous challenges, such as the limited expressiveness of IP addressing, multicast, complex mobility support and the energy efficiency requirement for IoT resource-constrained devices.

In order to solve those problems, ICN is considered as the replacement of IP-based network architecture since ICN supports mobility, name-based routing and in-network caching. Some pioneer use cases (smart grid, smart home, etc.) for ICN-IoT networks have been investigated and applied by the ICN Research Group (IGNRG) of the Internet Research Task Force (IRTF) [76].

Generally, IoT devices like sensor nodes, actuators, etc., are battery-powered and consume energy when they process and transmit data [65]. To save energy, IoT devices spend the majority of their lifetime in sleep mode. They are only awake when they need to process and transmit data. In integrated ICN-IoT networks, IoT devices, e.g., monitoring sensors, are the content producers and the user applications are the content consumers. Content consumers send interest packets to content producers to retrieve data. If the data item is cached at an intermediate node (i.e. is between the users and the IoT devices), the content consumers can retrieve the data directly from that node instead of the content producers. Therefore, the content consumers can get the data without activating the IoT devices, which leads to low energy consumption. Hence, if the IoT data items are cached properly at the intermediate nodes (such as content routers, BSs, etc.), the IoT network can gain a great benefit in terms of energy efficiency and users can get the data faster.

A great deal of research has been conducted on the traditional Internet data caching [1] [17] [62] [90] [103]. Unlike the traditional Internet data items, e.g., video data, IoT data items often expire within a certain time period after being generated by the content producer [90]. This thesis defines this time period as data lifetime. The term freshness is used to express how recent an IoT data item is, after being generated. Caching IoT data is more challenging than caching traditional Internet data, since the IoT data lifetime and its freshness need to be considered to make caching decisions. The lifetime of IoT data varies based on the type of data. For instance, traffic monitoring data has a shorter lifetime compared to temperature monitoring data. Besides, different applications may have different freshness requirements for the same type of data. For example, some applications may require the current temperature data, while other applications can be satisfied with the temperature data that was generated 5 minutes ago or even earlier.

In this chapter, IoT data lifetime-based cooperative caching decision (LCC) approach which considers both the IoT data lifetime and the request rate in a certain time period is proposed. The aim is to reduce the massive access to the IoT devices so that they can stay in sleep mode for most of the time, while the data would still be available at the intermediate nodes (content routers, BSs, etc.). Hence, the energy consumption of IoT devices and the data retrieval delay can be greatly reduced. The intermediate nodes can cooperate with each other by configuring a caching threshold based on their topology location information and the request rate.

This thesis evaluated the proposed approach by comparing with existing approaches, [1], [7] and [90]. Simulation results show that the proposed approach outperforms existing approaches in terms of the total energy consumption and the average number of hops.

The main contributions of this chapter are summarized as follows:

- Unlike other approaches that use energy efficient hardware or particularly designed protocols [107] to improve energy efficiency for IoT, this thesis integrates ICN and IoT to save IoT devices' energy.
- Through configuring a caching threshold based on each intermediate node's topology location information and request rate, the intermediate nodes can cooperate with one another to perform cooperative caching.
- A sliding time window is introduced to measure the change of the request rate. This thesis designs and implement an auto-configuration mechanism that allows each intermediate node to adjust its caching threshold dynamically based on the current request rate.
- This thesis develops a simulator (written in C++) to evaluate the proposed caching approach. The simulation results show that the proposed caching approach outperforms existing approaches, in terms of total energy consumption and the average number of hops.

The remainder of this chapter is organized as follows: Section 5.2 first explains the basic concepts used in this chapter, such as "data lifetime", "freshness" and "intermediate nodes". Then, the proposed LCC approach and detail the auto-configuration mechanism of the caching threshold are presented. Section 5.3 presents the simulation setup and the simulation results. Finally, Section 5.4 summaries this chapter.

5.2 IoT Data Lifetime-based Cooperative Caching

In this section, the concepts of “data lifetime”, “freshness” and “intermediate nodes” are explained firstly. Then, this thesis describes the specifications of LCC and the auto-configuration mechanism of the caching threshold.

5.2.1 Basic Concepts

1) IoT data lifetime

Data lifetime can be defined as the length of time between which a data item is generated by the content producer and the time it is no longer valid, i.e., it expires.

In ICN, there are two types of packet: *interest packet* and *data packet* [1]. There is a signed information field which contains information about publisher ID, key locator, stale time, and timestamp, etc. By checking the timestamp, a node can know when the data was generated by the content producer.

2) IoT data freshness

Freshness is defined as the time difference between the time at which the data was generated (T_g) by the content producer and the current time. It can be defined as:

$$Freshness = Current\ time - T_g \tag{5.1}$$

The data freshness is 0 when the current time is T_g. When the data freshness equals to the data’s lifetime, the data item is expired, and it should be discarded.

Different applications may have different freshness requirements for the same data. The data can only be sent back to the application when the data freshness value is less than the application’s freshness requirements.

3) ***Intermediate nodes***

In this chapter, the IoT devices, e.g., IoT sensors, are the content producers and the user applications are the content consumers. All the nodes between the content producers and the consumers are referred to intermediate nodes, including the gateway node, the content routers and the BS.

5.2.2 IoT Data Lifetime-based Cooperative Caching Approach

This section describes the proposed IoT data lifetime-based cooperative caching (LCC) approach.

5.2.2.1 System Model

This thesis considers an ICN-IoT scenario as shown in Fig. 5.1. This thesis selects the IoT sensor nodes as the IoT devices in this scenario since they can generate IoT data items, i.e., they can be regarded as the content producers. The IoT sensor nodes stay in sleep mode for most of the time until a request comes or an event occurs, e.g., a timer expires for periodic sensing. Then the IoT sensor nodes are activated to perform the sensing task (e.g., sensing the local pollution level) and transmit the data items to the content consumers. All the data items that can be sensed by the IoT sensor nodes are denoted as $D = \{d_1, d_2, d_3, \ldots, d_l, \ldots, d_L\}$, where $|D|$ is the total number of data items. Because the packet size of IoT data items is usually small, this thesis assumes all the IoT data items have the same packet size. This thesis assumes each IoT sensor node can only sense one kind of data for the sake of simplicity.

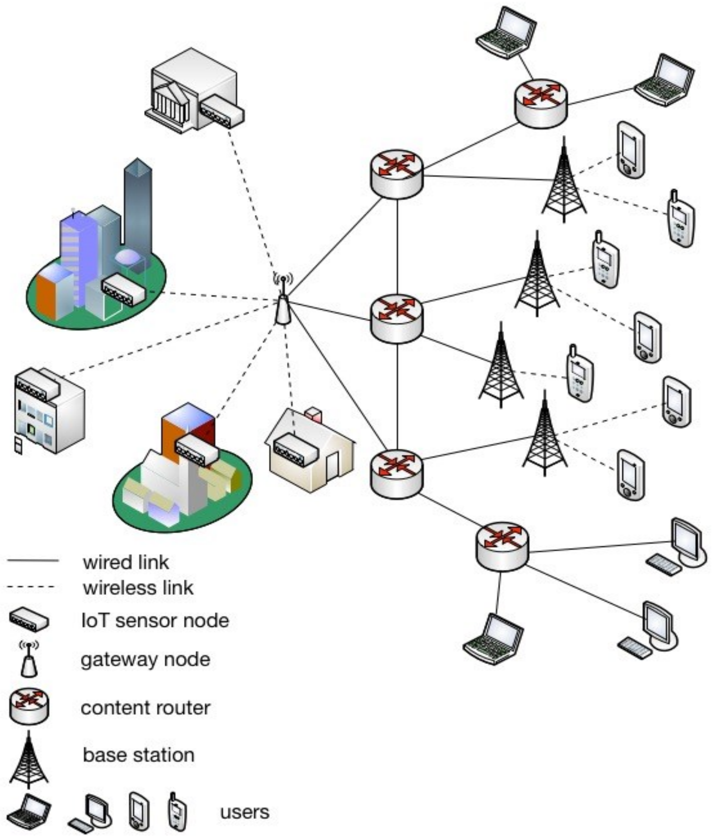


Fig. 5.1: System model

User applications send interest packets to retrieve data items. This thesis denotes the requests for the IoT data items at time t by $Req(t) = \{req_1(t), req_2(t), req_3(t), \ldots, req_n(t), \ldots, req_N(t)\}$. Each request in this set is represented by $req_n(t) = < d_l, f, t>$, where $d_l \in D$, f is the freshness requirement for data d_l, t is the time that the request arrives.

A wireless node (e.g., access point) is deployed here to act as a gateway node to provide Internet access to those IoT sensor nodes. The gateway node has storage to cache the data items that go through it. The nodes in the ICN core network are the content routers, while nodes in the ICN edge network could be content routers or BSs. All the nodes in the ICN network, including the content routers and BSs, have caching capability to store data items that go through them. A binary array $C(i, t) = \{c_{d_1}(i, t), c_{d_2}(i, t), c_{d_3}(i, t), \ldots, c_{d_l}(i, t), \ldots, c_{d_L}(i, t)\}$ is used to denote intermediate node i's caching status at time t. If a data item d_l is cached at intermediate node i at time t, $c_{d_l}(i, t) = 1$; otherwise, $c_{d_l}(i, t) = 0$. $\sum_1^L P \cdot c_{d_l}(i,t) = s$, where P is the packet size, s is the cache size of intermediate node i. Furthermore, $R(Req(t), \sum_{i=1}^{|I|} C(i,t)) = \{r_1(t), r_2(t), r_3(t), \ldots, r_n(t), \ldots, r_N(t)\}$ is used to represent how many requests in the request set $Req(t)$ that can be obtained from intermediate node i, where $i \in I$, I is the set of the intermediate nodes, $|I|$ is the total number of the intermediate nodes. $r_n(t) \in \{0,1\}$ indicates if the nth request can be served from the intermediate nodes at time t.

By performing a caching decision policy A, $C(i, t) \xrightarrow{A} C(i, t+1)$, the new caching status of intermediate node i at time $t+1$ can be achieved. The hit ratio is the basic metric to evaluate the efficiency of the caching decision policy, which is defined as the percentage of requests can be satisfied by the cache system. Therefore, the hit ratio can be denoted as follows:

$$H(A) = \sum_{t=1}^{T} \frac{1}{|Req(t)|} \cdot R\left(Req(t), \sum_{i=1}^{|I|} C(i,t)\right) \tag{5.2}$$

where T is the total time, $|Req(t)|$ is the total number of requests at time t. Theoretically, an optimal caching decision policy can be found if all the prior information is known [54]. However, it is not implementable as the future information cannot be known in advance. Hence, the aim is to find a practical caching decision policy to achieve better caching efficiency than existing policies.

5.2.2.2 Caching Decision Policy

A caching approach includes two parts: caching decision policy and caching replacement policy. The proposed LCC approach uses the Least Recently Used (LRU) first as the replacement policy. The principle of the caching decision is as follows:

- Each intermediate node has a caching threshold. The data can be cached at a node if one of the following conditions is satisfied: 1) The data lifetime is longer than the node's caching threshold; 2) the node has enough space to store the data.
- All intermediate nodes can be classified into three types: edge node, middle-level node, and root node. This classification is based on the topology information which could be obtained by calculating the number of hops from the content consumers (or content producers) to the node.
- The edge nodes are directly connected to the content consumers, e.g., content routers, BSs, while the root nodes are directly connected to the content producers, such as the gateway node. The rest of the nodes are the middle-level nodes.
- Different types of nodes have different caching thresholds, and they can adjust their caching threshold dynamically based on the current request rate by applying the proposed auto-configuration mechanism. The auto-configuration mechanism is discussed in Section III.C.
- The edge nodes have the smallest caching threshold so that they could cache more data to reduce the retrieval delay. The root nodes have the highest caching threshold, meaning that only data with a long lifetime can be cached. The

caching threshold of middle-level nodes is between that for the edge nodes and the root nodes.

Algorithm 5.1 IoT Data Lifetime-based Cooperative Caching

```
 1: a data item arrives at intermediate node i
 2: the node checks whether it has this content
 3: if (the data item is in cache) then
 4:   if (its freshness < cached data's freshness) then
 5:     cache (refresh) the item
 6:     forward the item to the next hop's node
 7:   else
 8:     forward the item to the next hop's node
 9:   end if
10: else if (has enough space) then
11:   cache the item
12:   forward the item to the next hop's node
13: else if (data lifetime ≥ caching threshold) then
14:   while (not enough space) do
15:     perform LRU replacement policy
16:   end while
17:   cache the item
18:   forward the item to the next hop's node
19: else
20:   forward the item to the next hop's node
21: end if
```

This approach exploits the caching capability of the intermediate nodes to avoid frequently activating IoT devices to reduce the energy consumption. The retrieval delay can be reduced by leveraging the in-network caching of ICN. The intermediate nodes can perform the caching decision policy as described in Algorithm 5.1.

5.2.2.3 Auto-configuration Mechanism

Since the proposed LCC is threshold-based, the configuration of the caching threshold can make a great impact on its performance. Unlike traditional Internet data, IoT data items are usually transient and small. The cached IoT data items can expire very quickly even before the next request comes. Therefore, the request rate should be considered when configuring the caching threshold of the intermediate nodes.

As discussed earlier, since there are three different types of nodes (root, middle-level, and edge), the threshold of node i (TH_i) can be denoted as follows:

$$TH_i = \alpha \cdot f_r(r_{it}) + \beta \cdot f_m(r_{it}) + \gamma \cdot f_e(r_{ir}) \tag{5.3}$$

where $\alpha + \beta + \gamma = 1$, $\alpha, \beta, \gamma \in \{0,1\}$. In other words, the type of node i can only be one of the three types at a given time. r_{it} is the request rate of node i at time t, $f_r(\,)$ is the threshold decision function of the root nodes, $f_m(\,)$ is the threshold decision function of the middle-level nodes, $f_e(\,)$ is the threshold decision function of the edge nodes.

When the request rate r_{it} increases, TH_i should be de creased so that more data items can be cached at intermediate node i to reduce the retrieval delay. On the other side, if r_{it} decreases, TH_i should be increased to avoid caching short lifetime data at intermediate node i, because data with short lifetime may expire before it can be served for the upcoming request when r_{it} is small. Therefore, the auto-configuration mechanism can be described as follows:

If $r_{it+\Delta t} > r_{it}$ (i.e., request rate is increasing), then:

$$\begin{cases} f_e(r_{it}) = age(i)_{min} \\ f_m(r_{it}) = \dfrac{1}{\sigma} \cdot (age(i)_{min} + age(i)_{max}),\ \ 1 < \theta < \sigma \\ f_r(r_{it}) = (1 - \dfrac{1}{\sigma\theta}) \cdot age(i)_{max},\ \ \sigma \le \dfrac{age(i)_{max}}{age(i)_{min}} \end{cases} \tag{5.4}$$

Else if $r_{it+\Delta t} \le r_{it}$ (i.e., request rate is decreasing), then:

$$\begin{cases} f_e(r_{it}) = \theta \cdot age(i)_{min}, \quad \theta > 1 \\ f_m(r_{it}) = \frac{1}{\theta} \cdot (age(i)_{min} + age(i)_{max}), \ 1 < \theta < \sigma \\ f_r(r_{it}) = \left(1 + \frac{1}{\sigma\theta}\right) \cdot age(i)_{max}, \sigma \le \frac{age(i)_{max}}{age(i)_{min}} \end{cases} \tag{5.5}$$

where $age(i)$ is the set of data lifetime for intermediate node i in a sliding time window (from time t to time $t + \Delta t$). θ and σ are configurable weights which are used to decide the increment (or decrement) amount of the caching threshold.

With (5.4) and (5.5), the proposed LCC can dynamically adjust the intermediate nodes' caching threshold based on the recent request rate.

5.3 Performance Evaluation

This section first introduces the evaluation metrics, then presents the simulation setup and results.

5.3.1 Evaluation Metrics

Total energy consumption: The total energy consumption is the total amount of energy consumed by all IoT devices during the simulation. This thesis uses the total energy consumption to evaluate the efficiency of the caching decision approach in reducing energy consumption. A better caching decision approach can save IoT devices' more energy by avoiding activating them too frequently.

The energy consumed by IoT device j (denoted as e_j) for transmitting one bit to the gateway node can be calculated by the following equation [34]:

$$e_j = e_t + b \cdot D_j^a \tag{5.6}$$

where J is the set of IoT devices ($j \in J$), e_t is the energy consumed by a transmitter for transmitting one bit, b is the energy cost of the transmitter amplifier, D_j is the Euclidean distance between IoT device j and the gateway node, and a stands for path loss factor.

This thesis uses $e_{sensing}$ to denote the energy consumed by IoT devices for sensing one bit. Since IoT device also consumes energy when transferring from the sleep mode to the active mode [64], denoted by e_{awake}, the total energy consumption E_{total} can be calculated as:

$$E_{total} = \sum_{j=1}^{|J|} n_j \cdot [P \cdot (e_j + e_{sensing}) + e_{awake}] \tag{5.7}$$

where P is the packet size, $|J|$ is the total number of IoT devices, n_j represents for how many times IoT device j is activated, which is affected by the hit ratio of caching decision policy A, it can be calculated as:

$$n_j = \sum_{t=1}^{T} \left[|Req(t)| - R\left(Req(t), \sum_{i=1}^{|I|} C(i,t) \right) \right] \tag{5.8}$$

where $|I|$ is the total number of the intermediate nodes, T is simulation time, $C(i,t)$ is used to denote node i's caching status at time t, $|Req(t)|$ is the total number of requests at time t.

Average number of hops: The data retrieval delay is measured by the average number of hops which is calculated by (5.9).

$$\overline{Hop} = \sum_{t=1}^{T} \frac{1}{|Req(t)|} \cdot Hop\left(Req(t), \sum_{i=1}^{|I|} C(i,t) \right) \tag{5.9}$$

where $Hop(Req(t), \sum_{i=1}^{|I|} C(i,t)) = \{hop_1(t), hop_2(t), hop_3(t), \ldots, hop_n(t), \ldots, hop_N(t)\}$ represents how many hops are used for satisfying each request at time t. If $req_n(t)$ can be satisfied from the intermediate node *i*, *hop*$_n$(*t*) equals to the number of hops from intermediate node *i* to the user; otherwise, *hop*$_n$(*t*) equals to the number of hops from the content producer to the user.

5.3.2 Simulation Setup

Table 5.1: Parameter Settings

Description	**Value**
Simulation Time (*T*)	3,000 s
Number of intermediate nodes (*I*)	10
Number of IoT sensor nodes (*J*)	30
Number of IoT data types (*D*)	30
Transmission energy consumption (e_t)	50 nJ/bit [34]
Sensing energy consumption ($e_{sensing}$)	150 nJ/bit [34]
Transmit amplifier (b)	100 pJ/bit/m^2 [34]
Path loss factor (a)	2 [34]
Awake energy consumption (e_{awake})	$7.34 \cdot 10^4$ nJ [64]
Packet size (P)	500 bytes [34]

This thesis chooses LCE [1], *Prob caching* [7] and *caching transient data* [90] approaches as comparisons. LCE is the default caching approach in ICN where each data item is cached at every node along the data delivery path. Evidently, the content redundancy of LCE is extremely high. For the sake of reducing the content redundancy, *Prob caching* was proposed by caching a content with a certain probability. [90] is a more recent work, the authors present another probability-based caching approach where they exploit in-network caching, the key feature of ICN, to cache transient data for IoT. The *caching transient data* approach takes both the data freshness and the multi-hop communication cost into consideration, which is more efficient than *Prob caching*. LRU is applied as the replacement policy for LCE, *Prob caching* and LCC, while LFF (Least Fresh First) is used as the replacement policy for *caching transient data* approach based on [90].

The scenario illustrated in Fig. 5.1 is used for the simulation. There are 30 kinds of IoT data items, and their lifetime is uniformly distributed between 1 second and 1 minute. User applications request those data items with a random freshness requirement (less than the data's lifetime) from the edge nodes randomly. The request generation follows a stationary Poisson process. For simplicity, this thesis assumes there are 30 IoT sensor nodes ($|J| = 30$) to sense those data items. A shortest path routing protocol is applied in this scenario. A simulator (written in C++) is developed and be use to evaluate the proposed LCC approach. The parameter settings are summarized in Table 5.1.

5.3.3 Simulation Results

This thesis performed 30 different runs for each caching approach and calculated the average over these 30 runs to plot results.

5.3.3.1 Impact of Cache Size

There are 6 edge nodes as shown in Fig. 5.1. Hence, the request rate is set to $\lambda = 6/s$, so that there is one request per second at each edge node on average. Fig. 5.2 illustrates the impact of cache size in terms of total energy consumption E_{total}. This thesis uses the proportion of the total data items that can be cached at a node to represent the cache size. We can see that LCC outperforms other caching approaches in the experiments. As expected, all these caching approaches reduce E_{total} with an increasing of the cache size, and they can achieve a great reduction of E_{total} when the cache size ranges from 0% to 30%. LCC can reduce about 46% energy consumption compared to no caching used. Also, even though LCE has a high content redundancy, it has significantly less energy consumption compared to both *Prob caching* and *caching transient data* approaches. When the intermediate nodes' cache size keeps increasing until they can cache all the data, i.e., cache size = 100%, LCC and LCE have the same performance with a reduction of around 48% of the total energy consumption compared to no caching used. The *caching transient data* approach outperforms *Prob caching* when the cache size is less than 90%. The reason behind this phenomenon is that the caching probability is too small for a data item with a short lifetime in the *caching transient data* approach even if the node has enough space to cache all the data. We can see that LCC can achieve a significant reduction in energy consumption with a small cache size, e.g., 45% reduction in total energy consumption when the cache size is 30%. Therefore, LCC is more efficient than other approaches in terms of total energy consumption, especially when the cache size is small, e.g., 30%.

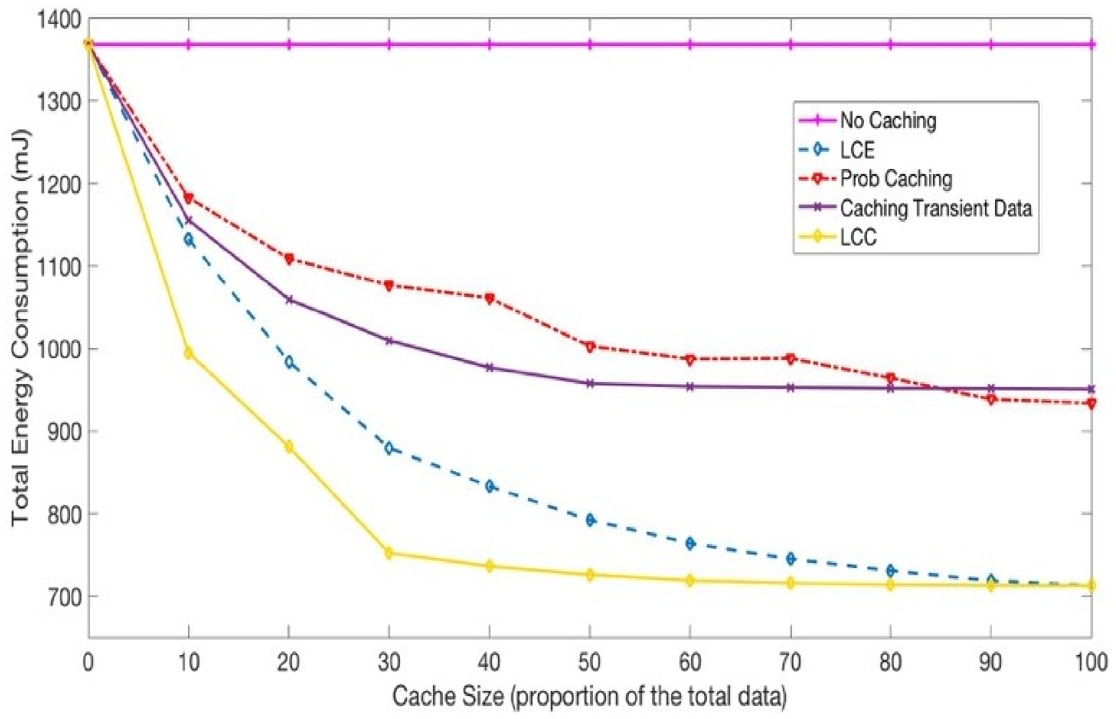


Fig. 5.2: Total energy consumption VS cache size

Fig. 5.3 demonstrates how the cache size influences the performance of the average number of hops. Similar to Fig. 5.2, all these caching approaches achieve a great reduction in the average number of hops when the cache size ranges from 0% to 30%. What's more, LCC outperforms other caching approaches, and it can reduce about 28% in the average number of hops compared to no caching used. Because the more efficient the caching approach is, the more requests can be served from the intermediate nodes which are closer to the users. Notably, the performance of the *caching transient data* approach is better than the *Prob caching* approach regardless the change of cache size. This is due to the fact that the *caching transient data* approach considers the trade-off between the data freshness and the multi-hop communication cost.

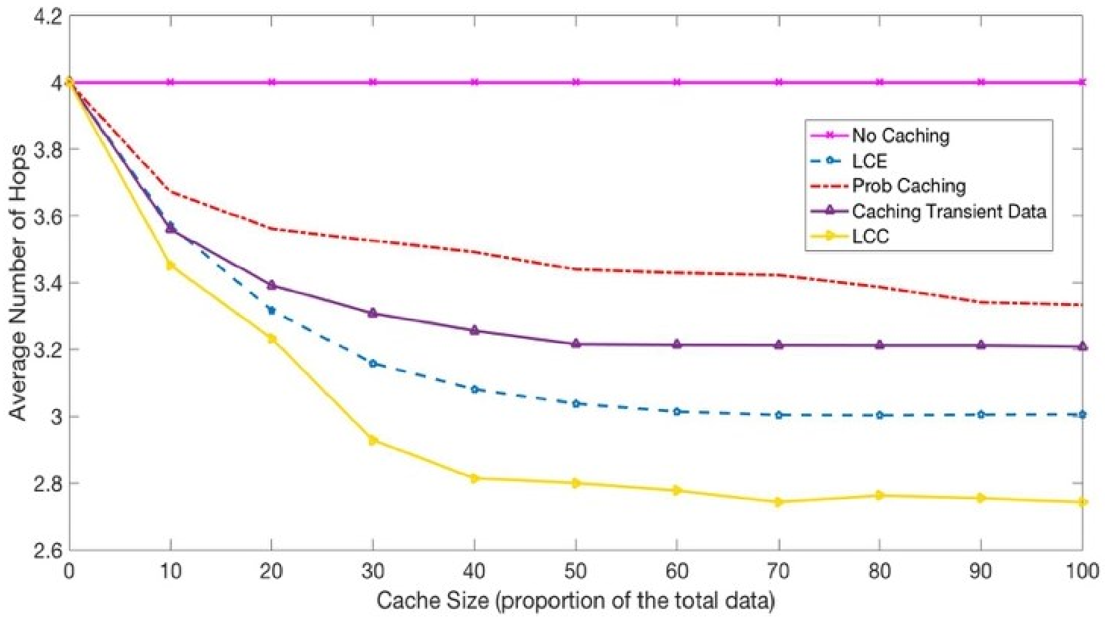


Fig. 5.3: Average number of hops VS cache size

5.3.3.2 Impact of Request Rate

Based on the simulation results of the impact of the cache, as described in the previous sub-section, this thesis sets the cache size to 30%. In addition, the request rate is varied (λ = 1/s, 5/s, 10/s, 15/s, 20/s, 25/s, 30/s) to explore the impact of total energy consumption and the average number of hops.

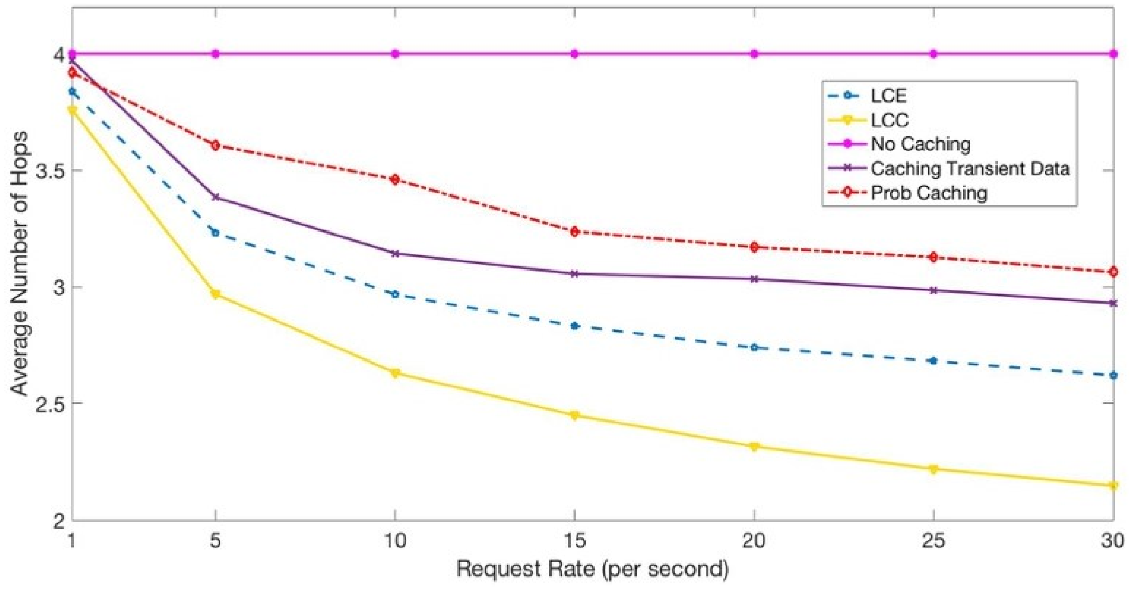


Fig. 5.4: Average number of hops VS request rate

Fig. 5.4 shows the effect of the request rate λ on the total energy consumption. Obviously, the total energy consumption keeps decreasing when the request rate increases. In fact, a higher λ indicates that the data items could be requested multiple times before they expire. Hence, the caching approach can reduce more energy consumption with higher request rates. Fig. 5.4 demonstrates that the performance in total energy consumption of the proposed LCC approach is the best for all request rates. When $\lambda = 30/s$, LCC provides reduction of about 70% and 40% in terms of total energy consumption compared to no caching used and LCE, respectively.

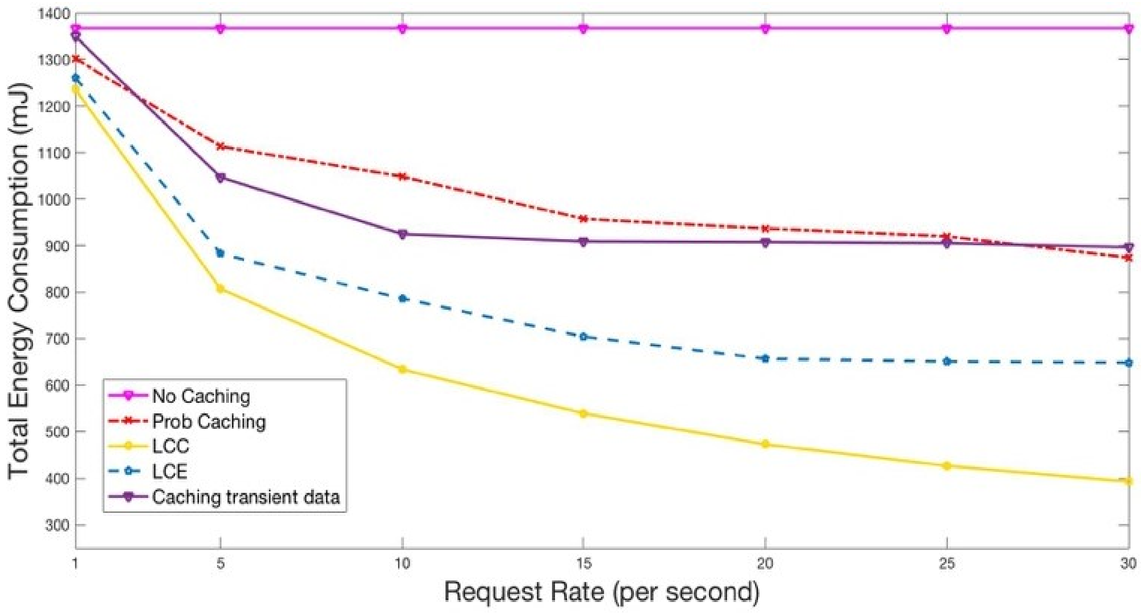


Fig. 5.5: Total energy consumption VS request rate

Fig. 5.5 describes the trend for the average number of hops with different λ. As expected, all these caching approaches have a poor performance when $\lambda = 1/s$, the reason of this phenomenon is that data may expire before the next request arrives when λ is small. When the request rate is increased, all the caching approaches achieve a notable reduction in the average number of hops. The proposed LCC approach outperforms other caching approaches for all different request rates; it can reduce around 47% in terms of the average number of hops compared to no caching used, and about 20% compared to LCE when $\lambda = 30/s$.

5.4 Summary

This chapter proposes an IoT data lifetime-based cooperative caching (LCC) approach for ICN-IoT networks. Both the IoT data lifetime and the request rate are taken into consideration to reduce the IoT devices' energy consumption and the data retrieval delay.

An auto-configuration mechanism was proposed to adjust the caching threshold dynamically so that the LCC approach can perform well under varying request rates. The evaluation results show that the proposed LCC approach is significantly more efficient compared to existing caching approaches in reducing the total energy consumption of IoT devices and the data retrieval delay.

Chapter 6: Proactive Caching for Autonomous Vehicle Users

6.1 Introduction

Video streaming over vehicular networks will play an increasing role in the near future. It will become a new way of entertainment for users in vehicular networks. Hence, how to distribute videos efficiently for vehicular networks will become a huge challenge.

AVs are equipped with smart sensors and intellectual analytic tools and are expected to drive themselves safely with little or no human input. Recently, the rapid development of AVs has boosted its testing and deployment within a much shorter time than previously expected. Companies like Google, Tesla, Uber, Baidu, etc., have brought self-driving vehicles closer to reality than ever. With the help of AVs, drivers do not need to focus on the road all the time. Instead, they can relax for a while and enjoy the scene of the trip, especially when the full self-driving vehicles (which is the highest level of AVs [112]) become a reality.

Recent advances in wireless networking technologies, such as 5G cellar networks, have reshaped the ways of entertainment for users in vehicular networks to browse the web, listen to the radio, play online games and watch videos. In the near future, AVs will become new entertainment places for mobile users. However, due to the short transmission range of RSUs and BSs in 5G networks [77], users will incur frequent handoffs and shorter connection durations. Therefore, users may have to reconnect to the original video content provider for some applications once a handoff occurs, which induces heavy overheads and high video retrieval delay. Compared to pedestrians, this situation will be worst for AV users due to their higher velocity. Furthermore, the high demand for entertainment services (such as video streaming services) also creates a huge pressure for the vehicular networks.

Hence, how to improve users' QoE and reduce the vehicular networks backhaul load are becoming crucial challenges.

To cope with these challenges, caching videos at the edge nodes (e.g., BSs, RSUs) has been proposed. In this way, users (drivers and/or passengers) can retrieve videos from the edge nodes, which can reduce the video retrieval delay and the backhaul load of 5G networks. However, the existing IP-based Internet paradigm is unsuitable for vehicular networks, since it was designed for host-to-host communications, not for mobile content delivery. More specifically, it cannot support in-network caching and mobility without additional techniques, e.g., DNS for supporting in-network caching lookup, mobile IP for supporting mobility. Information-Centric Networking (ICN) [1] was proposed to cope with the issues of the current Internet. Unlike traditional IP-based networks, ICN supports name-based routing, in-network caching and mobility by nature, which makes ICN more suitable for vehicular networks to support video streaming services for AV users [57]. With the help of the in-network caching feature of ICN, proactive caching can be implemented directly in ICN-based vehicular networks. Therefore, 5G-ICN is a promising paradigm for providing video streaming services for AV users.

Generally, caching can be categorized into reactive caching and proactive caching [61]. In reactive caching, videos can only be cached at a node when they are transmitted through that particular node. In other words, if a video has never been requested via a specific node, then there is no cached copy of this video at this node. Unlike reactive caching, proactive caching can fetch videos in advance from the content provider or cloud servers before users' requests arrive. This means that although a video has never been requested from a RSU, the RSU can still proactively cache the particular video and send it back to the AV users when requested. Due to the limited storage space available in RSUs, they are only able to cache a limited number of videos. Further, the high diversity of users' preferences makes it extremely difficult for the next user to request the exact same videos as the last few users. Hence, proactive caching is more suitable for AV users compared to reactive caching.

Existing research efforts on proactive caching are at the video-level [12] [36] [116], which is not suitable for AV users. In fact, the high velocity of AVs and the short transmission range of 5G BSs/RSUs lead to short connection durations which means that AV users can only retrieve a small portion of a video from a BS/RSU. As a result, the storage of BSs/RSUs is wasted for caching the rest of the same video. Hence, to improve the efficiency of proactive caching for mobile users, some recent works (such as [118]) propose caching videos at the chunk-level. This chapter also follows this idea and proposes a chunk-level caching approach for AVs users.

For proactive caching, two sub-problems need to be solved: the "What" problem deals with what to cache, and the "Where" problem addresses where to cache. Hence, future demands (the "what" problem) and user mobility (the "where" problem) are two major factors that need to be considered in proactive caching for AV users in vehicular networks. For future demands prediction, the most recent research trend is to use machine learning techniques [12] [36] [116].

On the other hand, matrix factorization (MF), an advanced machine learning technique [43], is another popular approach to predict users' future demands. As users' preferences are the primary reason that makes videos have different levels of popularity, future user demands can be predicted by predicting the ratings of videos that have not been watched.

Since MF-based approaches consider users preferences, they can achieve a notable improvement in terms of the user satisfaction, average video retrieval delay and hit ratio compared to traditional proactive caching approaches [12] [37] [95]. Recently, some MF-based proactive caching approaches [12] [37] [95] are proposed from the users' preferences perspective. The authors of [12] propose a singular value decomposition (SVD) based proactive approach that outperforms the reactive caching approach. But SVD may generate negative ratings as predictions for low rated videos which is considered impractical for real life networks [37]. To overcome this shortcoming, non-negative matrix factorization (NMF) based proactive caching schemes are proposed. The recent work in [37] predicts the users' future demands by using NMF and making caching decision based on the predicted user demands. However, they model the proactive caching as a mixed-integer linear programming (MILP) problem which is a typical an NP-hard problem [29]. Hence, their proposed approach is impractical due to the high computational complexity.

Before the rapid development of networking technologies, e.g., 5G, the user mobility information, such as moving direction, velocity, destination, route, etc., was really hard to obtain in real time in traditional vehicular networks. Therefore, Markov-based predictors [84] were used in the past to predict the user mobility. However, nowadays, all the aforementioned information can be easily retrieved from the self-driving system of AVs. As a result, the mobility information can be calculated based on the information. Therefore, a more accurate proactive caching approach can be achieved.

Although some excellent research efforts have been reported on proactive caching, most of the existing works only consider either the future demand or the user mobility, i.e., only one problem (either "what" or "where") is answered. This chapter considers both the future demand and the user mobility to propose a novel hierarchical proactive caching approach. Compared to the statistic model and the neural network model, NMF has less number of parameters. Furthermore, NMF also considers users' preferences, which makes NMF more suitable than other techniques for caching videos for AV users. Therefore, this chapter uses the NMF technique to predict the user future demands. The distinct features of this research are as follows:

- Unlike the existing proactive caching approaches [12] [37] [36] [95] [110] [116] that use traditional IP networks as the basic infrastructure for vehicular networks, the chapter proposes to use the ICN paradigm, as it supports in-network caching and mobility by nature.
- This chapter proposes a hierarchical approach which caches videos at both edge nodes (BSs and RSUs) and core network nodes (routers).
- As AV users have a very short connection duration with edge nodes, they may only retrieve a few chunks of a video. As a result, the proposed hierarchical approach works at the chunk-level to improve the caching efficiency. On the other hand, AV users will keep a long connection duration with the core networks for watching the entire video. Hence, the proposed hierarchical approach works at the video-level for nodes in the core networks.
- Traditional MF-based approaches (NMF, SVD) can predict future demands by predicting the user future ratings on videos. However, they may generate inaccurate predictions for unpopular but highly rated videos. To resolve this issue, not only the predicted ratings are considered, but also the previous popularity of videos are used to predict the users' future demands.

- A user mobility prediction module for vehicular networks is proposed to calculate the user future position and to decide which chunks should be cached at the future BSs/RSUs that the user will be connected to.

The remainder of this chapter is organized as follows: Section 6.2 describes the system model. In Section 6.3, the details of the NMF algorithm are presented. Section 6.4 illustrates the details of our proposed proactive caching approach. Simulation setups and numerical results are shown in Section 6.5. Finally, Section 6.6 concludes this chapter and outlines the future research directions.

Table 6.1: Notations

Symbol	**Definition**
V	Set of videos
U	Set of AVs users
I	Set of nodes
M	Total number of videos
N	Total number of AVs users
v_m	Video m
u_n	AVs user n
s_{v_m}	Size of video v_m
$K(v_m)$	Number of chunks that a video v_m can be divided
ξ	Size of a video chunk
$Req(i,t)$	Set of requests for all videos from node i at time t
$req_{v_m^k}(i,t)$	Total number of requests for the k^{th} chunk of video v_m of node i at time t
λ	Arrival rate
$C(i,t)$	Caching status of node i at time t
$c_{v_m^k}(i,t)$	If video chunk v_m^k is cached at node i at time t
cs_i	Cache size of node i
A	Caching decision
R	Rating matrix
$r(u_n, v_m)$	Rating of user u_n on video v_m

6.2 System Model

This section presents the system model for the proposed proactive caching approach, including the network architecture, the caching model and the rating model. The symbols (and their definition) that will be used in this section are summarized in Table 6.1.

6.2.1 Network Architecture

This chapter considers an ICN-based vehicular network for AVs users. More specifically, 5G technologies, including mmWave, massive MIMO, beamforming, etc., are used for the wireless communications at the physical layer, while ICN is used as the basic network architecture at network and transport layer.

AV users (both drivers and passengers) can send requests to retrieve videos from the video provider. This chapter denotes all AV users as $U = \{u_1, u_2, \dots u_N\}$ where N is the total number of AVs users. The set of nodes, including BSs, RSUs, and routers, is denoted as I, where any node in this network can be represented as $i \in \mathrm{I}$. We use $V = \{v_1, v_2, \dots, v_M\}$ to indicate the set of videos that can be retrieved from the video provider, where M is the total number of videos. s_{v_m} represents the size of video v_m. Any video $v_m \in V$ can be divided into multiple chunks, e.g., $v_m = \{v_m^1, v_m^2, \dots, v_m^k, \dots, v_m^{K(v_m)}\}$. For simplicity, all videos chunks are assumed to have the same size which is denoted as ξ. Consequently, the number of chunks that a video v_m can be divided into is $K(v_m) = \frac{s_{v_m}}{\xi}$. As mentioned before, AV users in the ICN-based vehicular networks can send interest packets to retrieve videos. The requests for videos at time t are denoted as $Req(i,t) = \{req_{v_1^1}(i,t), req_{v_1^2}(i,t), \dots, req_{v_1^{K(v_1)}}(i,t), req_{v_2^1}(i,t), \dots, req_{v_m^k}(i,t), \dots, req_{v_M^{K(v_M)}}(i,t)\}$ where $req_{v_m^k}(i,t)$ represents the total number of requests for the kth chunk of video v_m of node i at time t. Since Poisson distribution is widely used to represent users' request pattern, the arrival of video requests is assumed to follow a Poisson distribution with arrival rate λ.

6.2.2 Caching Model

All nodes (BSs, RSUs, and routers) in this ICN-based vehicular network have the capability to cache videos. The caching status of node i at time t can be represented by a binary array $C(i,t) = \{c_{v_1^1}(i,t), c_{v_1^2}(i,t), \dots, c_{v_1^k}(i,t), \dots, c_{v_2^1}(i,t), \dots, c_{v_m^k}(i,t), \dots, c_{v_M^{K(v_M)}}(i,t)\}$, where $c_{v_m^k}(i,t)$ represents if video chunk v_m^k is cached at node i at time t. More specifically, $c_{v_m^k}(i,t) = 1$ means that video chunk $c_{v_m^k}(i,t)$ is cached at node i at time t, while $c_{v_m^k}(i,t) = 0$ indicates that v_m^k is not cached at node i at time t. Since each node in the network can only cache a limited number of video chunks, the total size of all cached videos should be smaller than the node's storage capacity, hence we can have $\sum_{m=1}^{M} \sum_{k \in K(v_m)} [\xi \cdot c_{v_m^k}(i,t)] \le cs_i$, where ξ is the unit size of a video chunk, cs_i is the cache size of node i.

A centralized server is deployed to make caching decisions for all nodes. With a caching decision A, the caching status of node i at time t can be changed to the new caching status at time $t + \Delta t$, i.e., $C(i,t) \overset{A}{\rightarrow} C(i, t + \Delta t)$.

6.2.3 Rating Model

All AV users will rate the videos that they have watched to express their degree of preference. The ratings of AV users in U on videos in V can be presented as a matrix which is denoted as R. Each user-video pair in R is denoted as $r(u_n, v_m)$ which presents the rating of user u_n on video v_m. The rating matrix R is stored in the centralized server and it is used to predict users' future ratings on all videos by using the NMF technique.

6.3 Problem Formulation

This section presents how the caching decision problem is formulated. Then, the NMF technique, which will be used to predict the users' future ratings on all videos, and the application of NMF to the target problem are illustrated.

6.3.1 Caching Decision Problem Formulation

$H(i,t)$ is used to represent the number of requests that can be served from node i at time t, i.e., it indicates the number of cache hits. $H(i,t)$ can be calculated as:

$$\begin{aligned} H(i,t) &= |Req(i,t) \cdot \ C(i,t)| \\ &= \sum_{m=1}^{M} \sum_{k \in K(v_m)} req_{v_m^k}(i,t) \cdot c_{v_m^k}(i,t) \end{aligned} \tag{6.1}$$

Obviously, $H(i,t)$ can be used to represent the efficiency of caching. Since a caching decision A can update the caching status of a node, namely, $C(i,t)$, the efficiency of caching is highly dependent on the caching decision. In order to achieve the maximum efficiency of caching, a proper caching decision policy which can make effective caching decisions needs to be found. To find a proper caching decision policy, two sub-problems need to be solved: the "What" and the "Where" problems.

6.3.1.1 The "What" Problem

Since the cache size of a node is limited, only a limited number of videos can be cached at the node at a time. However, caching different videos will lead to different caching status, ($C(i,t)$) which may have an impact on the efficiency of caching, ($H(i,t)$). Therefore, it is essential to find out what videos should be cached to solve the first sub-problem. More specifically, the goal of the "What" problem is to find a binary array $\{c_{v_1^1}(i,t), c_{v_1^2}(i,t), \dots, c_{v_m^k}(i,t), \dots, c_{v_M^{K(v_M)}}(i,t)\}$ that can achieve the maximum $H(i,t)$ under the condition of cs_i.

6.3.1.2 The "Where" Problem

Once we know what video chunks should be cached, the next problem consists to find where these video chunks should be cached. $D_{ij}(v_m^k)$ is used to denote the delivery cost for video chunk v_m^k from node i to node j, and it can be calculated as follows:

$$D_{ij}(v_m^k) = \frac{d_{ij}}{c_{v_m^k}(j,t)} \tag{6.2}$$

where d_{ij} is the end to end delay from node i to node j. If $c_{v_m^k}(j,t) = 1$, it means that video chunk v_m^k is cached at node i, and therefore $D_{ij}(v_m^k) = d_{ij}$; otherwise, $D_{ij}(v_m^k) = \infty$. Moreover, d_{ij} can be calculated as follows:

$$d_{ij} = \sum_{h \in H} \left(\frac{\xi}{BW_h} + \frac{Dist_h}{3 \times 10^8} + Q_h \right) \tag{6.3}$$

where H is the set of hops that a video needs to be delivered where h is the h^{th} hop in H. BW_h is the available bandwidth of the link for the h^{th} hop, $Dist_h$ is the physical link length for the h^{th} hop, Q_h is the queueing delay for the h^{th} hop, and 3×10^8 (in meters) is the approximate speed of light in a vacuum, which can be considered as the speed of electronic signals that travel in the physical cable (or electromagnetic waves that travel through the air). Clearly, $\frac{\xi}{BW_h}$ is the transmission delay and $\frac{Dist_h}{3\times 10^8}$ is the propagation delay for transmitting video chunk v_m^k at the hth hop.

Therefore, the goal of the "Where" problem is to find the proper node i for the binary array $C(i,t)$ to achieve the minimum delivery cost for transmitting video v_m to the user.

6.3.2 Non-negative Matrix Factorization Technique

NMF, as stated earlier, is one of the MF techniques [43] classified as an advanced machine learning technique. NMF has been widely used in recommender systems to predict users' ratings on never watched videos. Compared to other MF techniques such as SVD, NMF does not generate negative predictions, which is considered more suitable for video ratings in real life.

The idea of NMF is that there are W latent features that have impacts on the rating conducted by a user on a video. NMF tries to explain the ratings by characterizing both users and videos [43]. For example, features of a video could be actions, adventure, science fiction, etc. Similarly, features of a user could measure how much the user likes a movie on the corresponding movie features. By factorizing the original R (a $N \times M$ matrix) into user feature matrix P and video feature matrix Q where P is a $N \times W$ matrix which measures the extent of the association between users and user features, and Q is a $M \times W$ matrix which denotes the extent of the relations between videos and video features. All elements in P and Q are non-negative, namely, $\forall p_{nw} \in P, p_{nw} \geq 0$, $\forall q_{mw} \in Q, q_{mw} \geq 0$. By calculating the dot product of P and Q, the estimated rating matrix $\tilde{R}$, which is the approximation of the original matrix R, can be found and all missing ratings are filled with the estimated ratings denoted as $r(\widetilde{u_n, v_m})$. According to the descriptions of [43], $r(u_n, v_m)$ can be calculated as follows:

$$R \approx \tilde{R} = PQ^T \tag{6.4}$$

or

$$\begin{bmatrix} r_{11} & r_{12} \cdots & r_{1M} \\ r_{11} & r_{22} \cdots & r_{2M} \\ \vdots & \vdots \ \ddots & \vdots \\ r_{N1} & r_{N2} \cdots & r_{NM} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \cdots & p_{1W} \\ p_{21} & p_{22} \cdots & p_{1W} \\ \vdots & \vdots \ \ddots & \vdots \\ p_{N1} & p_{N2} \cdots & p_{NW} \end{bmatrix} \times \begin{bmatrix} q_{11} & q_{21} \cdots & q_{M1} \\ q_{12} & q_{22} \cdots & q_{M2} \\ \vdots & \vdots \ \ddots & \vdots \\ q_{1W} & q_{2W} \cdots & q_{MW} \end{bmatrix} \tag{6.5}$$

To predict how an AV user would rate a video, the dot product of user u_n and video v_m vector will result in a single number as:

$$r(u_n, v_m) \approx r(\widetilde{u_n, v_m}) = p_n q_m^T = \sum_{w=1}^{W} p_{nw} q_{wm} \tag{6.6}$$

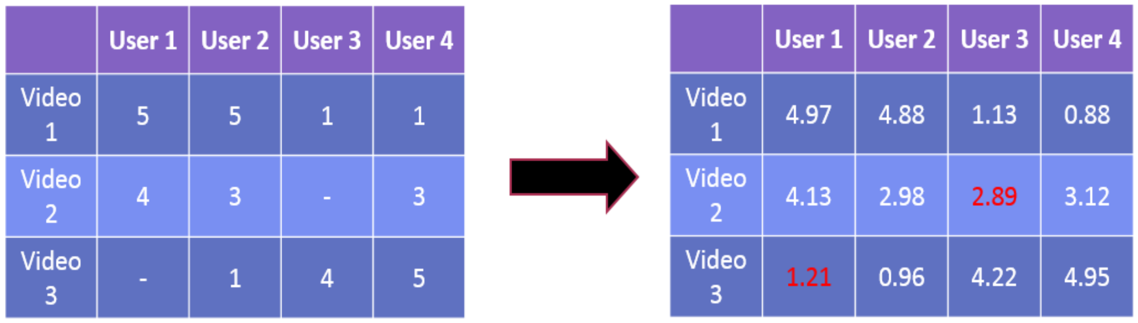
where $p_{nw} \in P, q_{wm} \in Q^T$, p_n is a n^{th} row vector, q_m^T is a m^{th} column vector.

The goal of the model is to generalize those previous ratings in a way that can predict future (i.e., unknown) ratings. Hence, the model should avoid overfitting the observed data by regularizing the learned parameters. Consequently, a regularization parameter, $\lambda > 0$ is involved to control the weight of the regularization term. Thus, to learn the latent features (P and Q), the most common approach is to minimize the regularized squared error on the set of know ratings [43]:

$$min \left[\sum_{m \in M, n \in N} (r(u_n, v_m) - p_n q_m^T)^2 + \lambda(\|p_n\|^2 + \|q_m\|^2) \right] \tag{6.7}$$

where $p_n > 0$, $q_m > 0$, and $\|\cdot\|^2$ represents the Euclidean norm of the vector.

Fig. 6.1 presents an example to show how the NMF technique works. The left table shows the original ratings (out of 5) of users on videos and the right table illustrates the predicted ratings of users on videos by performing the NMF technique. From the left table, we can see that user 3 has no rating on video 2 and user 1 has no rating on video 3. By performing the NMF technique based on Equation (6.4), the predicted rating of user 3 on video 2 is 2.89 and the predicted rating of user 1 on video 3 is 1.21. Based on these two predicted ratings, we can see that user 1 may not like video 3, while user 3 may prefer video 2 over video 1.



	User 1	User 2	User 3	User 4
Video 1	5	5	1	1
Video 2	4	3	-	3
Video 3	-	1	4	5

	User 1	User 2	User 3	User 4
Video 1	4.97	4.88	1.13	0.88
Video 2	4.13	2.98	2.89	3.12
Video 3	1.21	0.96	4.22	4.95

Fig. 6.1: An example for NMF technique

6.3.3 Alternating Least Squares Algorithm

Stochastic gradient descent (SGD) [43] and alternating least squares (ALS) [43] are two common algorithms to minimize Equation (6.7). Compared to stochastic gradient descent algorithm, ALS can be executed in parallel, which makes ALS much faster and more suitable for distributed systems such as cloud servers.

Taking the derivative of Equation (6.7) with respect to p_n (holding q_m constant), Equation (6.8) can be obtained as follows:

$$\frac{d}{d(p_n)}\left[\sum_{m\in M}(r(u_n,v_m)-p_nq_m^T)^2+\lambda(\|p_n\|^2+\|q_m\|^2)\right]$$

$$=\sum_{m\in M}2(r(u_n,v_m)-p_nq_m^T)(-q_m^T)+2\lambda p_n)$$

$$=2\sum_{m\in M}[(q_m^Tq_m^T+\lambda)p_n-r(u_n,v_m)q_m^T] \tag{6.8}$$

Let Equation (6.8) equal 0. Based on Equation (6.6), we have:

$$\sum_{m\in M}(q_m^Tq_m^T+\lambda)p_n=\sum_{m\in M}r(u_n,v_m)q_m^T \tag{6.9}$$

$$\Rightarrow p_n=r(u_n)Q(QQ^T+\lambda E)^{-1} \tag{6.10}$$

where E is the unit matrix.

Similarly, taking the derivative of Equation (6.7) with respect to q_m (holding p_n constant) yields Equation (6.11) as follows:

$$q_m=r(v_m)P(PP^T+\lambda E)^{-1} \tag{6.11}$$

To learn the suitable user feature matrix (P) and video feature matrix (Q), ALS first assigns random values to one matrix, e.g., Q. Then, since only one variable is unknown, the optimization problem for Equation (6.10) becomes quadratic which can be solved optimally. Similarly, Equation (6.11) can be solved optimally by using the previously solved P. Thus, the ALS technique keeps switching between these two steps until Equation (6.7) has converged.

6.4 Proposed Proactive Caching Approach

In this section, the proposed proactive caching approach is presented. The approach consists of three components: user future ratings prediction module, user mobility prediction module and caching decision module. All these modules are implemented in the same centralized server.

6.4.1 User Future Ratings Prediction Module

The user future ratings prediction module (UFRPM) predicts the user future ratings by using the NMF technique. The historical user watching information can be retrieved from the content provider and stored at the centralized server. Since NMF requires high performance computing capability, the UFRPM is implemented at the centralized server as well. As both the historical user watching information and the UFRPM are located at the centralized server, the UFRPM can use this data directly to predict user future ratings without extra data transmission cost. Since new videos may be released and user may watch videos that they haven't watched before, the historical user watching information should be updated periodically, and the UFRPM should be triggered periodically as well. The periodical update time is denoted as T_{update} which may vary for different data set, because different content providers may have different updating cycle or users in different places share different watching behavior.

6.4.2 User Mobility Prediction Module

The user mobility prediction module (UMPM) predicts the future position of AV users based on the velocity and position information of the AVs. The reason to predict the position is to find out in advance the video chunks that could be proactively cached before the AV reaches the next RSU. For AVs, the destination is set before the trip, and the route is planned accordingly. During the trip, the current position of the AVs and the velocity information can be easily obtained from the AVs' GPS module and relevant sensors. Based on the destination, route, current position and velocity information, the UMPM can easily predict the next position of the AVs. Therefore, video chunks can be proactively cached at the node which can serve the predicted position.

Compared to the existing methods to predict user mobility (such as Markov-based predictors [84]), the proposed UMPM can predict a more accurate future position of an autonomous vehicle since all mobility information can be obtained from the self-driving system of the vehicle.

The current velocity vector of an autonomous vehicle is denoted as $\overrightarrow{v(t)}$, t_b represents the time that an AV user needs to finish watching the video chunks buffered at the AV before fetching new chunks from the next RSU, and t_c is the current time. Therefore, the total distance (denoted as D) that an AV user can travel without fetching new video chunks can be calculated as follows:

$$D = \int_{t_c}^{t_b} \overrightarrow{v(t)}\, dt \tag{6.12}$$

Let $f(x, y)$ represent the planned route for the vehicle, (x_c, y_c) is the current position of the vehicle, and (x_p, y_p) is the predicted position of the vehicle. To calculate the predicted position (x_p, y_p) from the current location (x_c, y_c), we have:

$$D = \int_{x_c, y_c}^{x_p, y_p} f(x, y) ds \tag{6.13}$$

where $ds = \sqrt{(dx)^2 + (dy)^2}$ is the distance along the route.

Since D can be calculated by Equation (6.12), the predicted position (x_p, y_p) can be calculated as follows:

$$F(x_p, y_p) = F(x_c, y_c) + D \tag{6.14}$$

where $F(x, y)$ is the integral of $f(x, y)$. Hence, the predicted position, (x_p, y_p) can be easily calculated based on $F(x_p, y_p)$.

6.4.3 Caching Decision Module

The caching decision module (CDM) makes the final hierarchical caching decisions for core network nodes (e.g., routers) and edge nodes (e.g., RSUs) based on the predicted user future ratings, node cache size, the videos' previous popularity and the AVs mobility information. More specifically, the CDM makes caching decisions for core network nodes first based on the predicted ratings, core network cache size and the videos' previous popularity. Then, the CDM selects video chunks from videos that are already cached at the core network and proactively caches them at the edge nodes based on the edge nodes' cache size and the AVs mobility information. If a requested video is not previously cached either at the edge nodes or the core network nodes, the next several chunks of this video will be fetched directly from the video server and proactively cached at the edge nodes based on the outputs of UMPM. Moreover, nodes in the core network cache videos at the entire video level, while the edge nodes, e.g., RSUs, cache videos at the chunk level.

6.4.3.1 Caching for Core Network Nodes

Since all video chunks need to go through the core network, caching videos at the entire video level is more efficient compared to caching videos at the chunk level. Therefore, the CDM considers the videos' ratings and popularity to make caching decisions for core network nodes.

In addition, the predicted user future ratings are used to predict if a video will be liked by users, these predicted ratings have a strong impact on the CDM to make the final caching decisions. Hence, one thing we cannot ignore is that the NMF technique may generate inaccurate predictions for high rated but unpopular videos, e.g., cult films which have a strong attraction on their fans. Unfortunately, this issue has not been addressed in the existing NMF-based proactive caching approaches [12]. To describe this issue, this research assumes that a video v_m is only requested by e users, where $e \ll N$ (i.e., video v_m is not popular). Further, this research assumes that each of the e users gives a high rating for video v_m. When performing the NMF technique to learn the video feature matrix Q, v_m will get high values for its features in Q, namely q_{mw} ($w \in W$, W is the number of features) have high values, due to its high rating in the historical data set. Therefore, the predicted ratings for each user on video v_m would be higher than real ratings as the product of $q_m^T p_n$ is the estimated rating for user u_n on video v_m. Thus, the predicted user future ratings for video v_m will be overestimated, which will lead to biased caching decisions.

To solve this issue, we take the previous video popularity into consideration to make the final caching decision. As each node has a limited caching storage to cache videos, the cache size of the node is another important factor that needs to be considered. Thus, $b_{v_m} \in B$ is denoted as the benefit of caching video v_m, where B is the total benefit that can be achieved for the entire network by performing proactive caching with the node cache condition. Therefore, the aim of the CDM for node i is to achieve the maximum B:

$$Obj: \qquad \max B = \sum_{m \in M} b_m \tag{6.15}$$

$$s.t. \quad \sum_{m \in M} \left[C(i,t) \cdot s_{v_m} \right] \le cs_i \tag{6.16}$$

where s_{v_m} is the size of video v_m, and cs_i is the cache size of node i. Equation (6.16) states that node i can only cache a limited number of videos at time t.

The maximum B can be found by calculating b_m as follows:

$$b_m = \left[Pred(v_m) \cdot \frac{Pop(v_m)}{ns_{v_m}} \right] \tag{6.17}$$

where $Pred(v_m)$ is the normalization of the predicted ratings for video v_m, $Pop(v_m)$ is the normalization of the historical popularity of video v_m and $ns_{v_m}$is the normalized size of video v_m.

$Pred(v_m)$ is calculated as follows:

$$Pred(v_m) = \frac{\overline{Rate(v_m)}}{MaxRate} \tag{6.18}$$

where $\overline{Rate(v_m)}$ is the average predicted rating of video v_m, and $MaxRate$ is the maximum rating in video set V.

$Pop(v_m)$ is calculated as follows:

$$Pop(v_m) = \sum_{t \in T} \sum_{i \in I} \frac{max\left\{req_{v_m^k}(i,t)\right\}_{k=1}^{K(v_m)}}{MaxReq} \tag{6.19}$$

where $max\left\{req_{v_m^k}(i,t)\right\}_{k=1}^{K(v_m)}$ is maximum number of requests for the most requested chunk in video v_m, it can be regarded as the maximum number of requests for video v_m at time t. $MaxReq$ is the maximum number of requests in the video set V .

Equation (6.20) is used to normalize video size:

$$ns_{v_m} = \frac{s_{v_m}}{MaxSize} \tag{6.20}$$

where s_{v_m} is the real size of video v_m, $MaxSize$ is the maximum size of video in V.

The CDM works as follows to make caching decisions for nodes in the core network: By calculating b_m with Equation (6.17), the benefit for caching each video can be obtained. Then, the CDM ranks all videos based on the calculated b_m, and proactively caches videos from the top ranked video until the cache size of node i is full. For example, there are two videos (A and B) that are considered for caching, b_A is 20, b_B is 10, and one RSU which can only cache one video. To perform the proposed proactive caching approach, the CDM first ranks videos A and B based on b_A and b_B, and finds out video A is the 1st ranked video, hence, the CDM decides to cache video A at the RSU. However, the RSU can only cache one video, therefore CDM will terminate the process without selecting video B.

6.4.3.2 Caching for Edge Nodes

To reduce the retrieval delay and the backhaul traffic, videos should be proactively cached at the edge nodes. Different from caching videos at the video level for core network nodes, edge nodes cache videos at the chunk level due to the fact that AV users are moving fast, and the short range of RSUs only allows AV users to fetch a small number of chunks from the edge nodes. Therefore, caching entire videos at edge nodes will waste their cache storage, i.e., the efficiency of caching will be degraded.

Also, the arrival time $t_a(u_n, i)$ and departure time $t_d(u_n, i)$ of an AV user u_n at an edge node i can be calculated, since the UMPM can predict AV users' future position based on the planned route, velocity, and current position information. Let τ represent the duration that a video chunk can be played, the number of video chunks (denoted as $nc(u_n, i, v_m)$) that will be played during the period $t_d(u_n, i)$ and $t_a(u_n, i)$ can be calculated as follows:

$$nc(u_n, i, v_m) = \frac{\tau}{[t_d(u_n, i) - t_a(u_n, i)] \cdot BR(v_m)} \tag{6.21}$$

where $BR(v_m)$ is the bitrate of video v_m.

If the last chunk that the AV user watches before arriving at node i is denoted as k_l, then the video chunks that need to be cached for AV user u_n at node i during time $t_d(u_n, i)$ and $t_a(u_n, i)$ are from k_{l+1} to $k_{l+nc(u_n,i,v_m)}$, where $k_{l+nc(u_n,i,v_m)} \leq K(v_m)$,.

6.5 Performance Evaluation

In this section, the evaluation metrics are illustrated first. The two metrics used are the hit ratio and the average number of hops. Following that, the simulation settings and results are presented. The proposed proactive caching approach is evaluated in two scenarios: a highway scenario and a grid street scenario.

6.5.1 Evaluation Metrics

Hit ratio is a common metric for evaluating the efficiency of a caching decision policy. Although the definition of hit ratio is mentioned in Chapter 3, the calculation of hit ratio used in this chapter is different due to the different nature of the problem. The hit ratio is calculated as:

$$Hit\ Ratio = \sum_{t=1}^{T} \sum_{i \in I} \frac{H(i,t)}{|Req(i,t)|} \tag{6.22}$$

where $|Req(i,t)|$ is the total number of requests of node i at time t and $H(i,t)$ (mentioned in Section 6.3) is the number of requests that can be served from node i at time t.

Average number of hops: The average number of hops (denoted as $\overline{Hops}$) can be used to measure the QoE of AVs users, it can be calculated as follows:

$$\overline{Hops} = \sum_{t=1}^{T} \sum_{i \in I} \frac{Hops(Req(i,t), C(i,t))}{|Req(i,t)|} \tag{6.23}$$

where $Hops(Req(i,t), C(i,t))$ is the total number of hops of node i at time t under caching status $C(i,t)$.

6.5.2 General Settings

In this chapter, the proposed proactive caching approach is compared with an SVD-based approach and LCE. LCE is the default caching decision policy in ICN where each content is cached at every node along the content delivery path. In [12], the user future ratings are predicted with SVD (one of the matrix factorization techniques), and the predicted ratings are regarded as the user future demands. The only difference between SVD and NMF is that NMF does not generate negative values in the feature matrices, hence, these two techniques should have the same performance. Then a caching decision is made based on the predicted user future ratings.

Since the proposed proactive caching approach considers a hierarchy of cache storage (cache storage at the edge and core network) and has two key modules: UFRPM and UMPM, this chapter evaluates these two modules separately and together. More specifically, if UFRPM is turned off, the core network nodes only adopt the SVD-based approach and LCE. On the other hand, if UMPM is turned off, the edge nodes only adopt LCE due to the fact that LCE can support the video check level, but the SVD-based approach works at the entire video level. Therefore, we can have 6 different combinations for core network nodes and edge nodes respectively:

- UFRPM + UMPM which is the proposed approach
- UFRPM + LCE
- SVD-based + UMPM

- SVD-based + LCE (because the SVD-based approach only works at the video level, LCE is used for the edge nodes so that they can work at chunk level)
- LCE + UMPM
- LCE + LCE (the default approach in ICN)

The request data used for the evaluation comes from the latest public MovieLens dataset [33] in which 610 users request 9,742 videos 100,836 times. This chapter sorts all request entries by time stamp and uses the first 80% of the request entries as the training set for the UFRPM. The remaining 20% is used to evaluate the various approaches based on [12]. The parallel computing toolbox of MATLAB [60] is used to perform the ALS algorithm and to evaluate the proposed proactive caching approach. The video sizes range from 500 MB to 5 GB, and the size of each chunk is fixed at 64 KB.

6.5.3 Performance Evaluations

In this subsection, the simulation results of a highway scenario and a grid street scenario are presented to illustrate different situations. But these two scenarios could be combined for a planned route that covers both.

6.5.3.1 Highway Scenario

In real life, traveling on a highway is a very common scenario for AVs, especially for a long-distance trip. For AV users, entertainment services are more attractive for them during the trip. Therefore, it is essential to evaluate the proposed proactive caching scheme for the highway scenario. The important features of the highway scenario are:

- All AVs are moving in the same direction, and cannot change their moving direction.

- The velocity of AVs is much faster (between 80 to 100 km/hr) than the grid street scenario.

6.5.3.1.1 Simulation Setting

This chapter assumes that the arrival of requests follows a stationary Poisson process, and sets the arrival rate as 30/min to simulate the traffic for highway scenario. As shown in Fig. 6.2, all vehicles are moving in the same direction. AVs are connected to RSUs directly to fetch videos. RSUs are then connected to BSs which are then connected (using optical cables) to the core network. Videos are sent from the content provider via the Internet to the core network. The velocity of all vehicles ranges from 80 to 100 km/hour. The total length of the simulated highway is 5 km. The default RSU cache size is set to 300 video chunks, while the default core network cache size is set as 20% of the total videos.

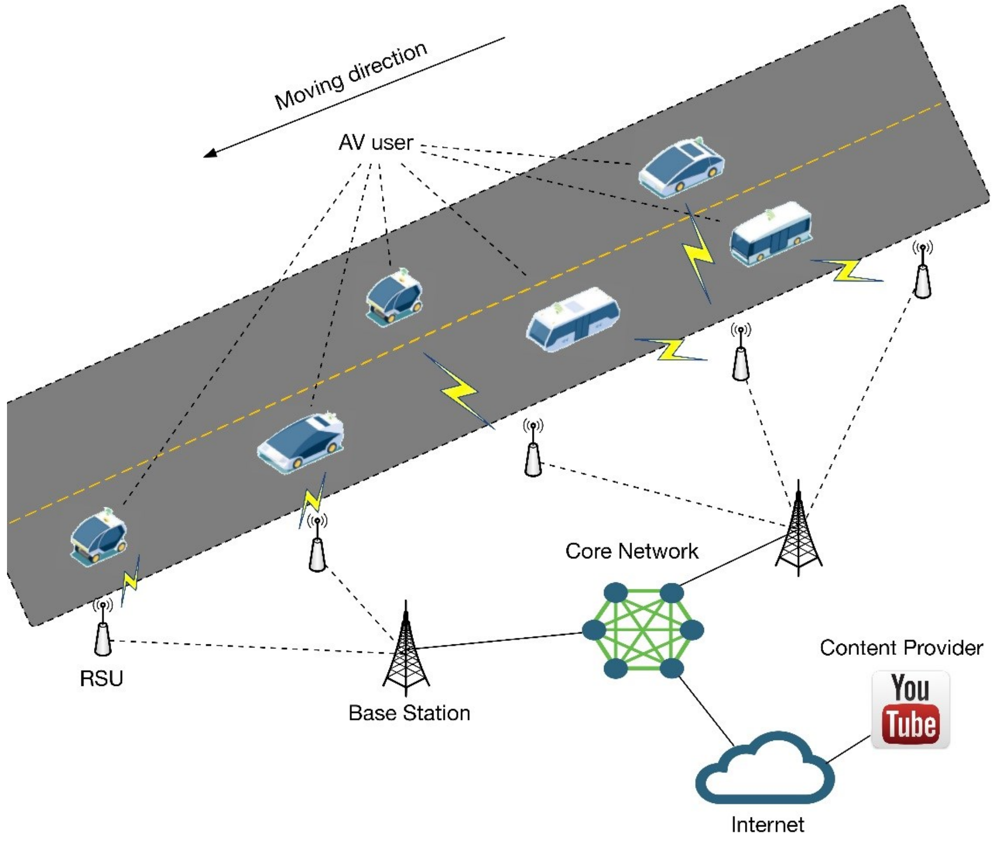


Fig. 6.2: Highway scenario

6.5.3.1.2 Performance Results of Highway Scenario

Fig. 6.3 demonstrates how RSU cache size influences the performance of the hit ratio when the core network cache size is set at 20%. The number of chunks that can be cached at the RSU is used to represent the cache size of RSU. Notably, with the increase of RSU cache size, approaches that use the UMPM can significantly improve the hit ratio, because bigger RSU cache size means that edge nodes can store more chunks, which can increase the chance of hit. We can see that the blue line (i.e. UFRPM + UMPM) has the best performance among the 6 combinations. Comparing the blue line, the yellow dash line and the black line, we can see that the proposed proactive caching approach can achieve higher hit ratio compared to the SVD-based approach and the default approach in ICN. The increasing rate of the hit ratio slows down when the RSU cache size is greater than 1,700 chunks. The reason behind this phenomenon is that most AV users' requests can be satisfied under the given arrival rate, namely 30/min. No matter how the RSU cache size increases, the hit ratio cannot be improved if the requested videos are not cached at the core network, which means that the requested videos have to be fetched from the content provider. Similarly, comparing the red dash line, the purple dash line and the black line,

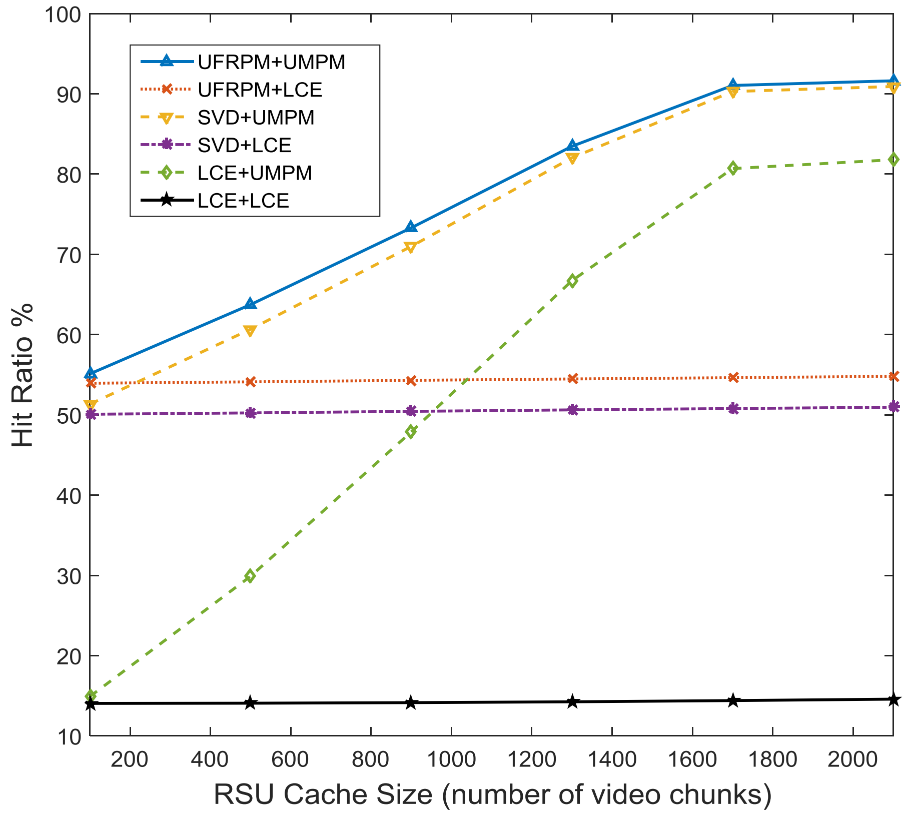


Fig. 6.3: Impact of RSU cache size on hit ratio

we can see that applying the UFRPM can achieve the best performance in terms of hit ratio compared to using the SVD-based approach and LCE for core network nodes.

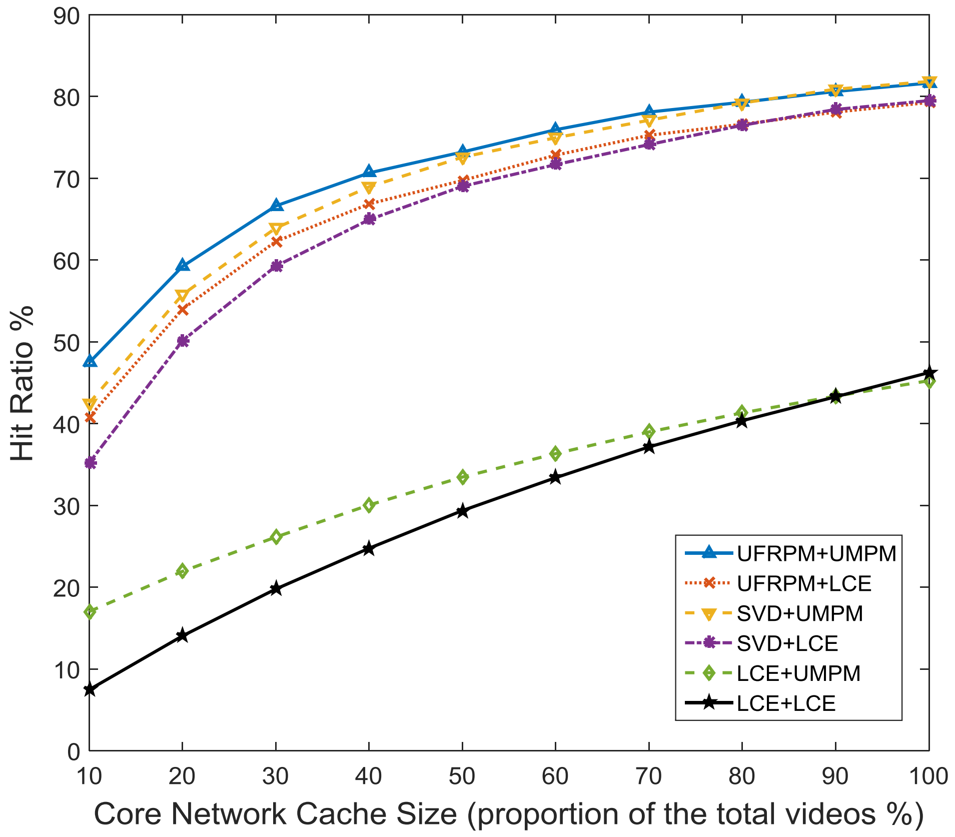


Fig. 6.4: Impact of core network cache size on hit ratio

Fig. 6.4 shows the impact of core network cache size on the hit ratio. The proportion of the total video's size that can be cached at the core network is represented as the core network cache size. Notably, with the increase of core network cache size, all approaches can increase the hit ratio. More specifically, UFRPM + UMPM has the best performance in terms of hit ratio among all combinations. Similar to Fig. 6.3, applying UFRPM outperforms the approach that uses the SVD-based approach and LCE for core network nodes whether the UMPM is applied or not for edge nodes. Another important point is that a higher hit ratio can be obtained for the same approach that is applied to core network nodes if the UMPM is turned on, i.e., the blue line (UFRPM + UMPM) is better than the red dash line (UFRPM + LCE).

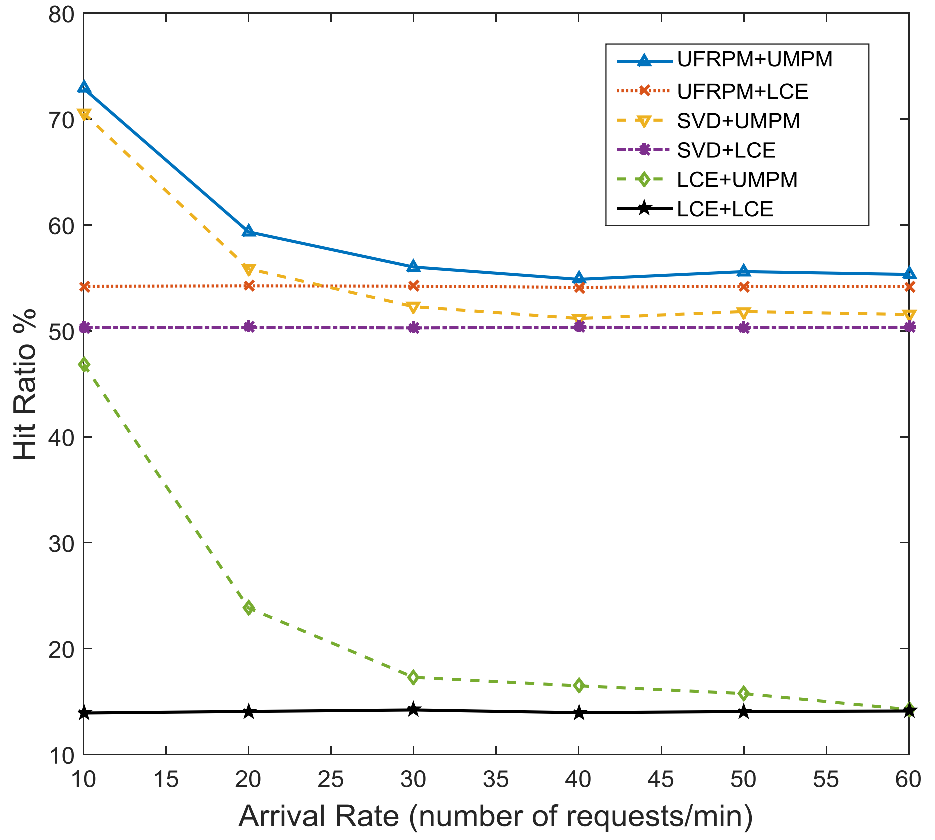


Fig. 6.5: Impact of arrival rate on hit ratio

To evaluate the impact of arrival rate on hit ratio, the cache size of RSUs and core network nodes should be fixed first. Consider that both RSUs and core network nodes have limited storage, the cache size of RSUs is set to 300 (the number of video chunks that can be stored at a RSU), the cache size of core network is set to 20% (proportion of the total videos). As shown in Fig. 6.5, the hit ratio decreases with the increase of the arrival rate for approaches where UMPM is turned on. The reason is that higher arrival rate means more requests from the AV users and therefore, the proportion of unsatisfied requests will increase due to the limited cache size of edge nodes. The arrival rate has significantly less impact for approaches that apply LCE as the caching decision policy for edge nodes, because the arrival rate does not affect the performance of LCE, as LCE always caches everything that goes through the node, while the UMPM will degrade to first-in first-out (FIFO) if the cache size of edge nodes is full and the number of AV requests keeps increasing. For example, the only difference between the green dash line and the black line is if the UMPM is turned on. With the increase of the arrival rate, the improved hit ratio that is generated by the UMPM is decreasing. When the arrival rate reaches 60/min, the green dash line and the black line achieve the same hit ratio, which means the UMPM has the same performance in terms of the improved hit ratio as LCE. Although the performance in terms of hit ratio decreases with the increase of the arrival rate, the proposed proactive caching approach for both core network nodes and edge nodes is always the best among all these 6 combinations.

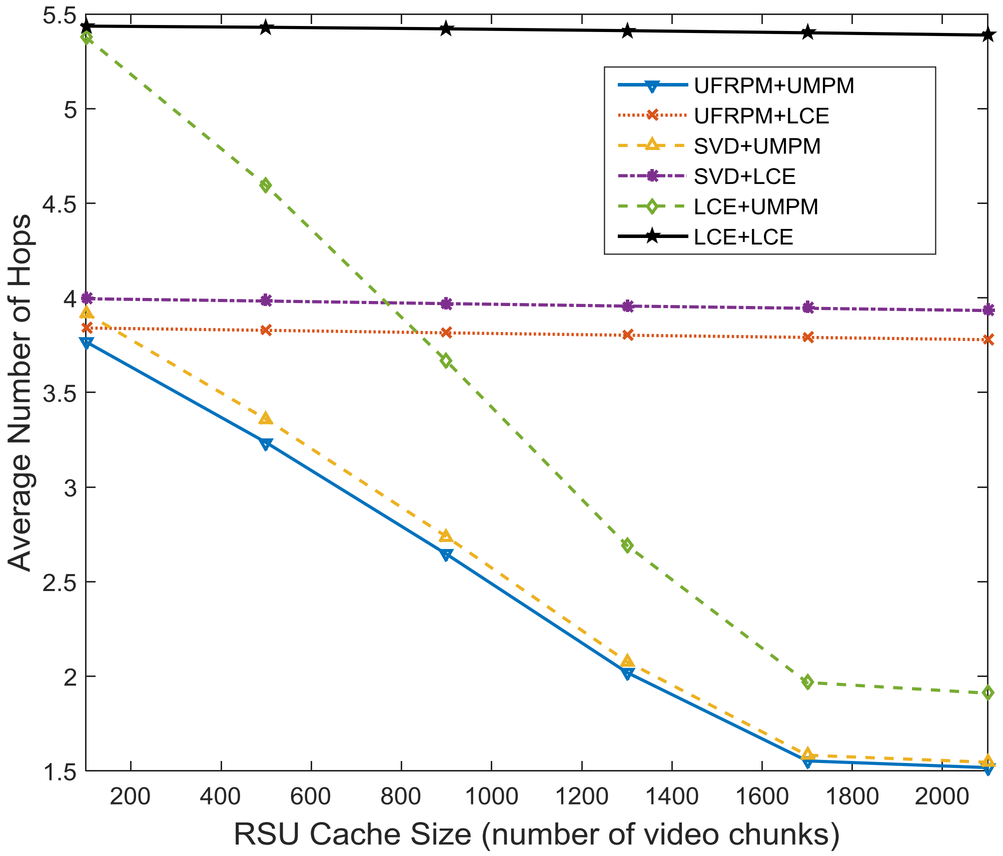


Fig. 6.6: Impact of RSU cache size on the average number of hops

Fig. 6.6 illustrates the impact of RSU cache size in terms of the average number of hops. Obviously, UFRPM + UMPM is the best among all these 6 combinations in terms of the average number of hops. For all combinations that disables the UMPM, the combination that uses UFRPM achieves the smallest average number of hops. However, the reduced average number of hops is slightly affected by the RSU cache size. If UMPM is enabled, the average number of hops can be significantly reduced, such as the comparison of the blue line and the red dash line.

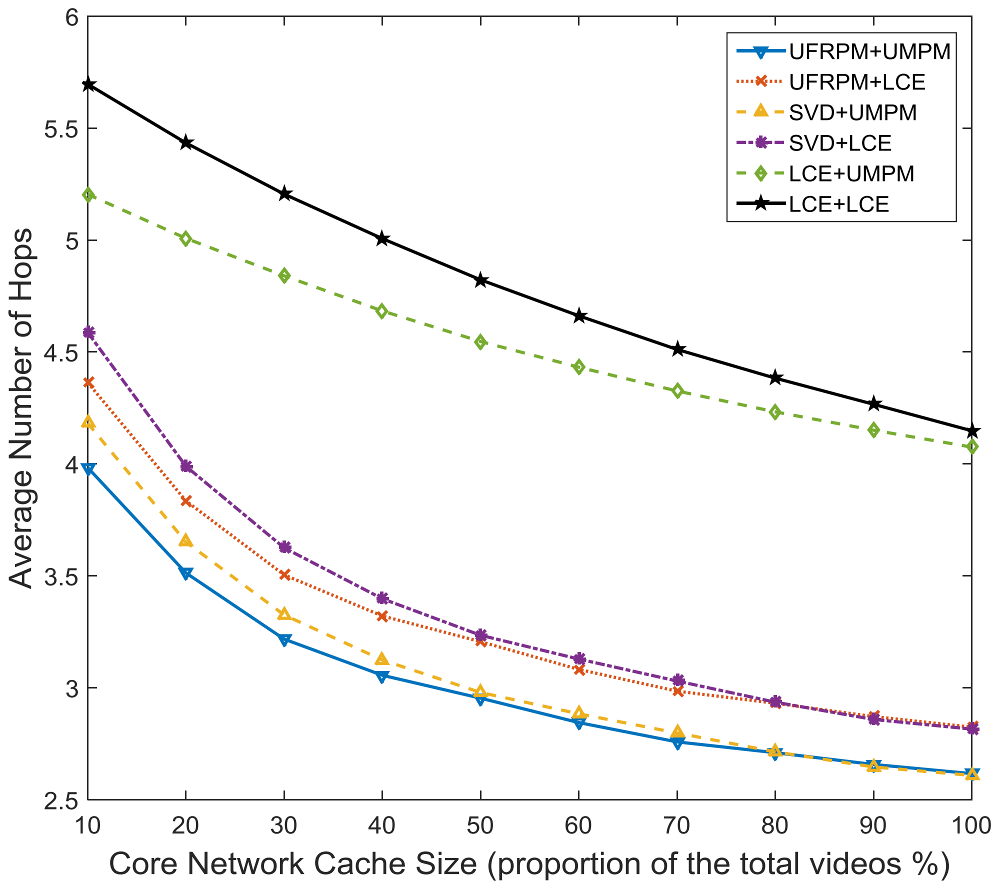


Fig. 6.7: Impact of core network cache size on the average number of hops

Fig. 6.7 presents how the core network cache size influences the average number of hops. Similar to Fig. 6.3, the proportion of the total videos' size that can be cached at the core network nodes is used as the core network cache size. We can see that all these 6 combinations can reduce the average number of hops with the increase of core network cache size. The proposed approach (UFRPM + UMPM) shows the best performance, especially when the core network cache size is small, e.g., 10%–30%. As the core network nodes in real life can only cache a small proportion of the total videos, the proposed proactive caching approach is more efficient compared to other approaches.

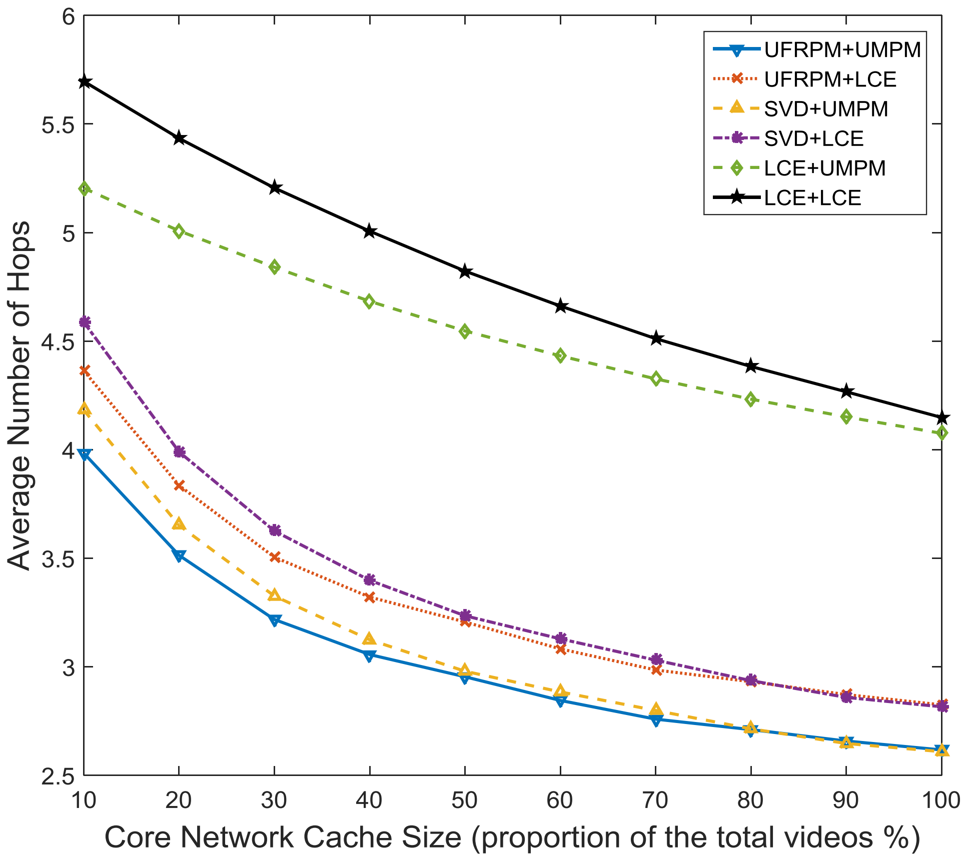


Fig. 6.8: Impact of arrival rate on the average number of hops

Considering that both RSUs and core network nodes have limited storage, the cache size of RSUs is set to 300 (the number of video chunks that can be cached) and the cache size of core network is set to 20% (proportion of the total videos). Next, the impact of the arrival rate on the average number of hops is evaluated. Fig. 6.8 shows that the increase of the arrival rate has a negative impact on the average number of hops for combinations which use UMPM, and has a slight positive impact for combinations which do not use UMPM. The reason behind this phenomenon is that higher arrival rates may increase the chance for requesting the same video chunk at a given time period, which can increase the efficiency of LCE. Evidently, the proposed proactive caching approach, namely UFRPM + UMPM, can achieve the best performance in terms of the average number of hops for all arrival rate settings, although the increase of the arrival rate has a negative impact.

6.5.3.2 Grid Street Scenario

Grid street scenario is another common scenario of AV users in real life. Compared to the highway scenario, the features of the grid street scenario are:

- The AVs in the grid street are moving in different directions (they have to follow streets).
- This chapter assumes that all AVs would not move back and forth on the same road, i.e., AVs would not change their current moving direction to the opposite one and repeat their previous routine.
- AVs could change their moving direction at each crossroad.
- The speed of the AVs is lower than that in the highway scenario (typically around 20-60 km/hr).

6.5.3.2.1 Simulation Setting

Similar to the highway scenario, the arrival of requests is assumed to follow a stationary Poisson process. The grid street scenario is shown in Fig. 6.9. The arrival rate is set as 30/min as well, and the velocity of all vehicles ranges from 20 to 60 km/hr. The grid street is set as 2 km × 2 km. The default RSU cache size is set to 300 video chunks, while the default core network cache size is set to 20% of the total videos.

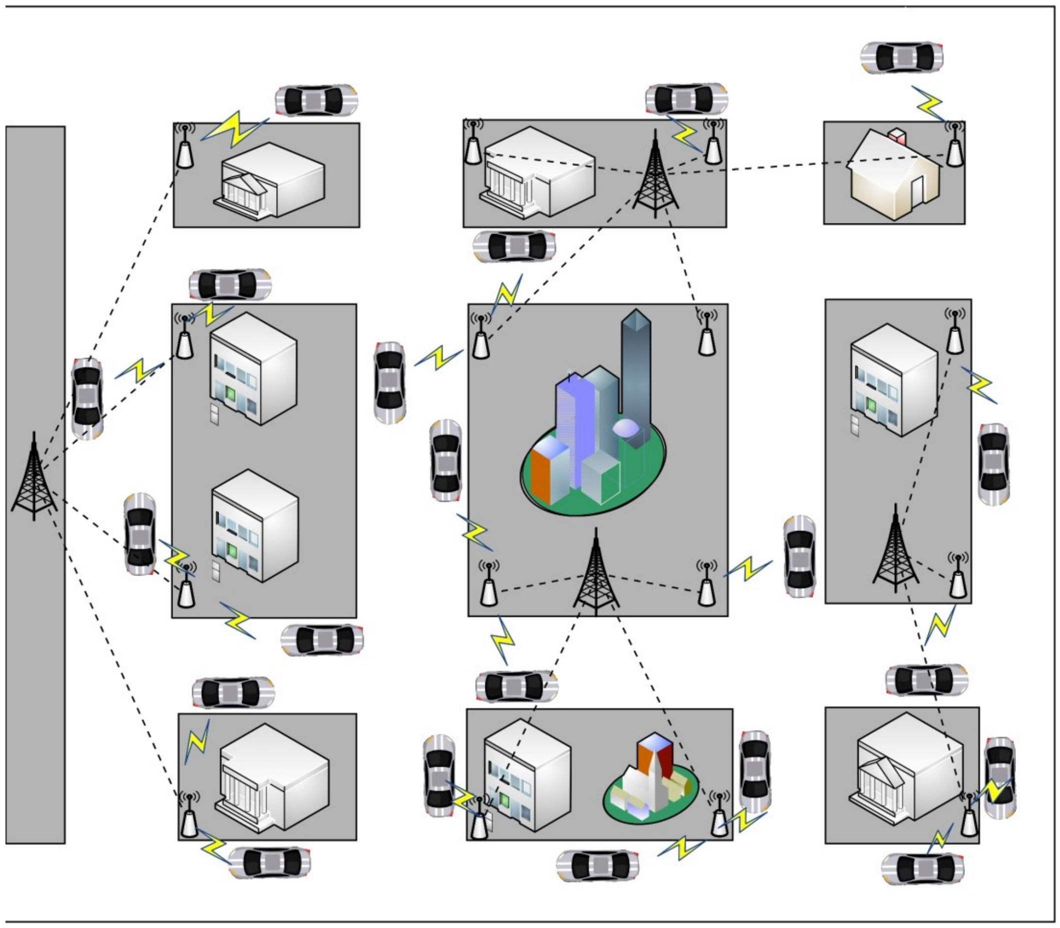


Fig. 6.9: Grid street scenario

6.5.3.2.2 Performance Results of Grid Street Scenario

Fig. 6.10 shows the impact of RSU cache size on the hit ratio. The first observation is that it shares a similar trend with the results from the highway scenario (Fig. 6.3). However, combinations with UMPM enabled can achieve higher hit ratio with smaller RSU cache size compared to the highway scenario. For example, the proposed proactive caching can achieve about 84% hit ratio when RSU cache size is 900 for the grid street scenario, while the proposed approach can only obtain 72% hit ratio under the same conditions for the highway scenario. The reason is that the various moving directions of AVs in the grid street scenario increases the chance of overlapping for a particular video. Obviously, the proposed proactive caching approach outperforms the other combinations regardless of the RSU cache size.

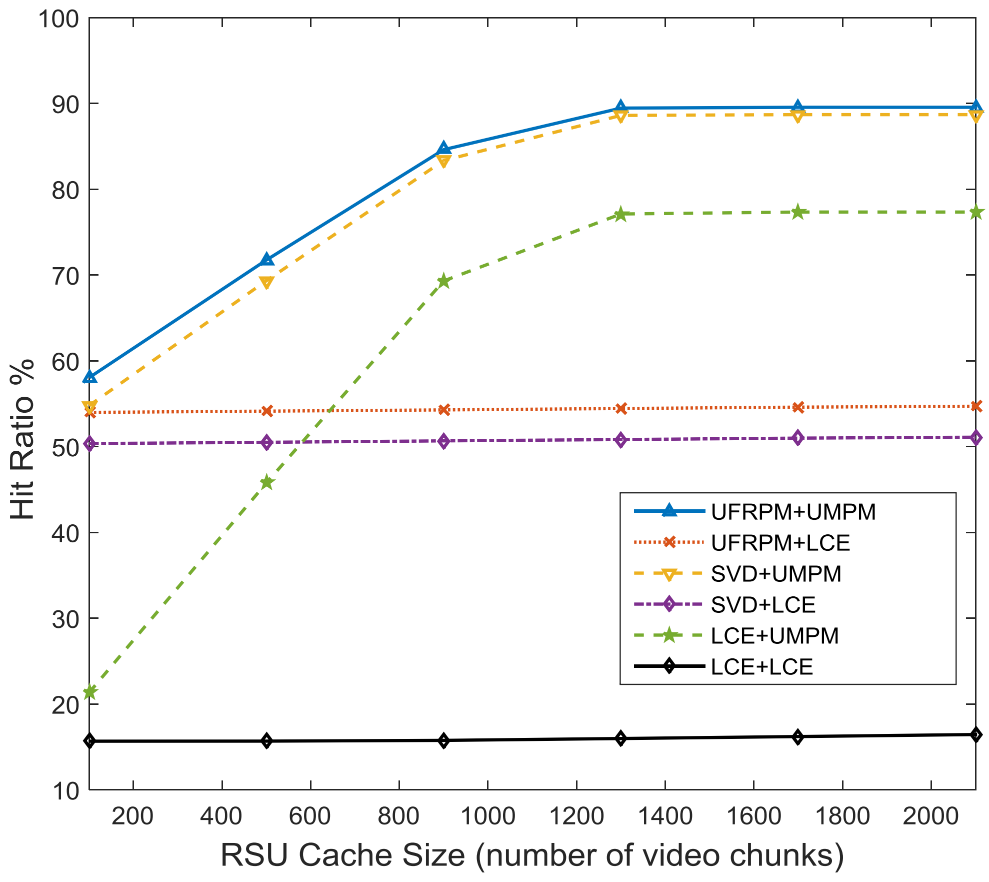


Fig. 6.10: Impact of RSU cache size on hit ratio

Fig. 6.11 demonstrates how the hit ratio changes with the increase of the core network cache size. Apparently, all combinations can improve the hit ratio if the core network cache size increases. Results in Fig. 6.11 share a very similar trend to that in Fig. 6.4. Comparing UFRPM + UMPM (blue line) with UFRPM + LCE (red dash line), we can see the UMPM can improve the hit ratio up to 14% from 41% to 55%. Comparing UFRPM + UMPM (blue line) with SVD + UMPM (yellow dash line), we can see that UFRPM (which considers the historical popularity of videos) outperforms the SVD-based approach by approximately 5% from 50% to 55%.

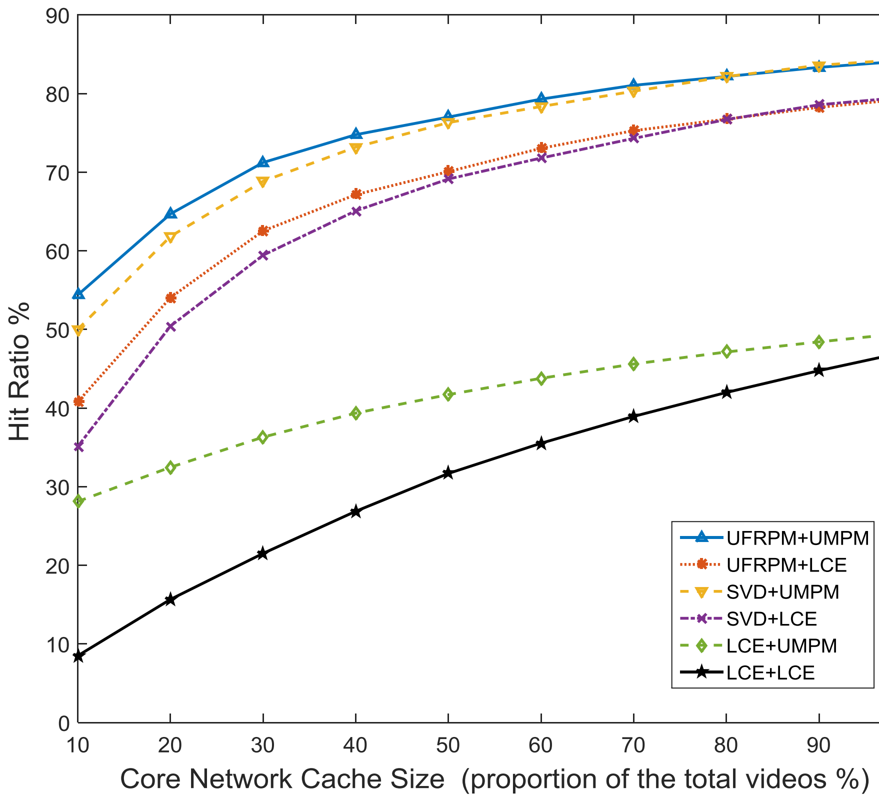


Fig. 6.11: Impact of core network cache size on hit ratio

As shown in Fig. 6.12, when the arrival rate is 10/min, all combinations with UMPM can achieve the highest hit ratio. After that, the hit ratio that can be achieved with the different approaches decreases with the increase of the arrival rate. The increase of the arrival rate does not have too much impact on the combinations which use LCE as the caching decision policy for edge nodes. The reason is the same as described for Fig. 6.5. When the RSU cache size is greater than 1,300, all the 6 combinations cannot generate significant improvement in the reduction of the average number of hops, which means that the core network size becomes the bottleneck.

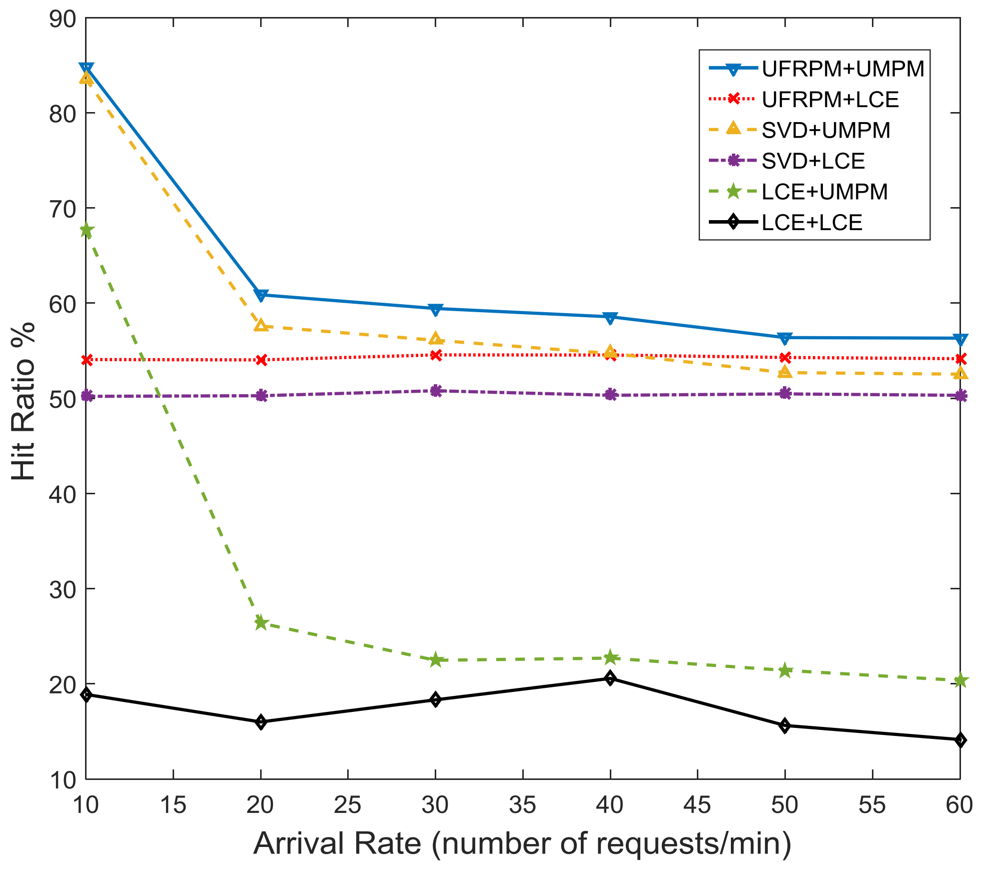


Fig. 6.12: Impact of arrival rate on hit ratio

The impact of RSU cache on the average number of hops is shown in Fig. 6.13. The increase of RSU cache size can reduce the average number of hops for all 6 combinations. By comparing the blue line and red dash line, we can see that the UMPM is much more efficient than LCE in terms of reducing the average number of hops. Comparing the blue line with the purple dash line and the black line, we can see that the proposed approach (UFRPM + UMPM) outperforms the existing approaches. When the RSU cache size is greater than 1,300 video chunks, increasing the RSU cache size is inefficient in reducing the average number of hops. The reason is that the RSU cache size is large enough for caching video chunks. However, if the requested videos are not cached at the core network, RSUs have to retrieve those video chunks from the content provider instead of from the core network nodes, which will cause an increase of the average number of hops.

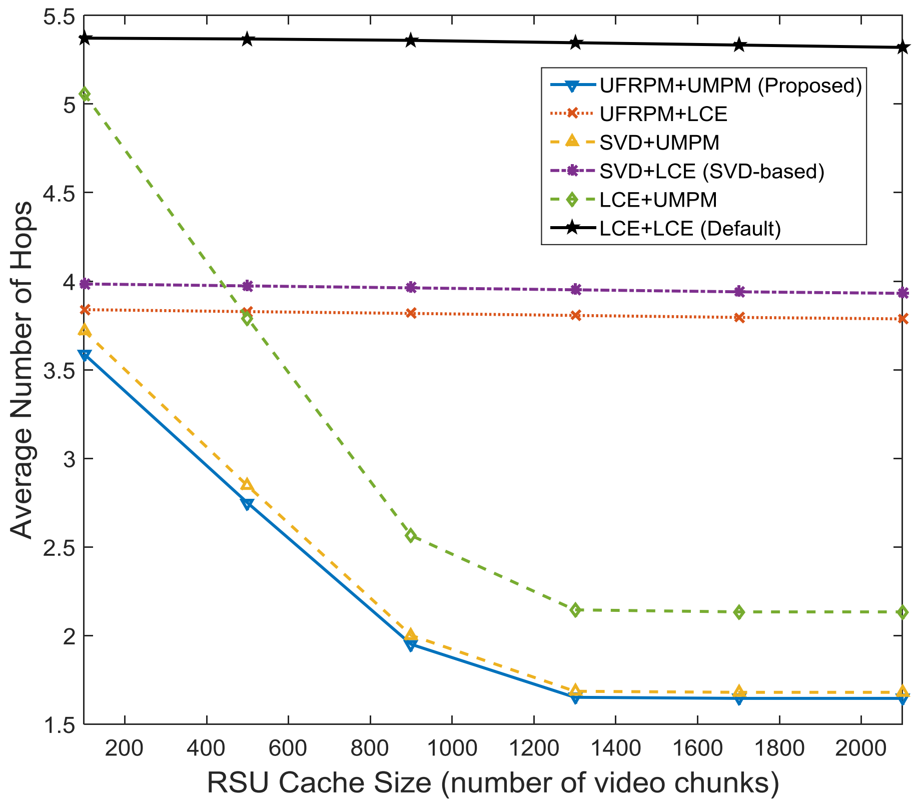


Fig. 6.13: Impact of RSU cache size on the average number of hops

Fig. 6.14 illustrates the impact of core network size on the average number of hops. Similar to Fig. 6.7, all combinations can reduce the average number of hops with the increase of core network cache size. Compared to the highway scenario, the proposed approach and the SVD-based approach in the grid street scenario are more efficient in terms of reducing the average number of hops. More precisely, the proposed approach can achieve 3.6 hops in Fig. 6.14 when core network cache size is 10%, while it can only achieve 4 hops in Fig. 6.7. The reason behind this outcome is that AVs in the highway scenario are moving much faster than AVs in the grid street scenario. RSUs in the highway scenario will replace the cached videos much more frequently than RSUs in the grid street scenario. Consequently, the time of video chunks that stay at the RSUs in the highway scenario is much shorter than in the grid street scenario, which lowers the chance of the cached video to be requested by another AV user. Comparing the proposed approach (UFRPM + UMPM) with the SVD-based approach (SVD + LCE) and the default approach in ICN (LCE + LCE), we can see that the proposed approach is always the best regardless of the core network cache size.

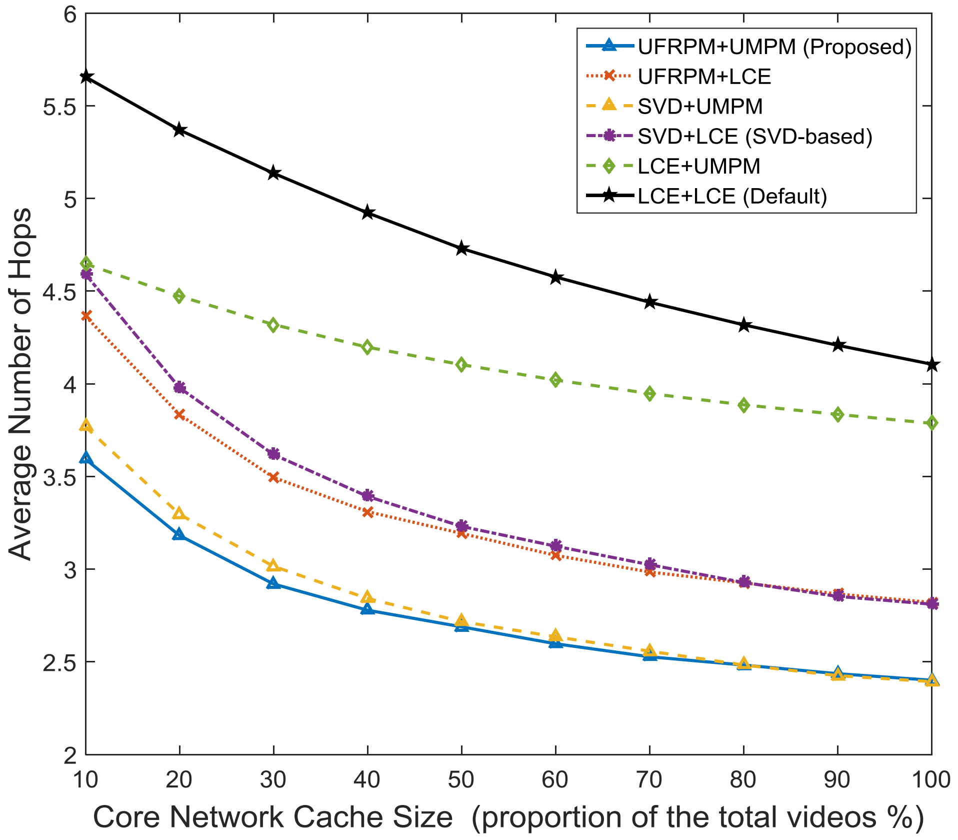


Fig. 6.14: Impact of core network cache size on the average number of hops

Fig. 6.15 demonstrates how the arrival rate influences the average number of hops. We can see that the proposed approach outperforms the other 5 combinations regardless of the arrival rate. Notably, combinations with UMPM are getting inefficient in reducing the average number of hops when the arrival rate increases, while combinations with LCE generate less impact with the increase of arrival rate. Because edge nodes are more sensitive to the changes of the arrival rate due to the fact that video chunks are cached at the edge nodes based on the mobility information of AV users, while videos are cached at the core network nodes based on the predicted ratings and historical popularity of videos. When the arrival rate increases, more requests will arrive at the edge nodes. However, the limited storage of RSUs increases the chance that a request cannot be satisfied at the RSUs. Hence, those unsatisfied requests have to be forwarded to the core network nodes, or even the content provider, which increases the average number of hops.

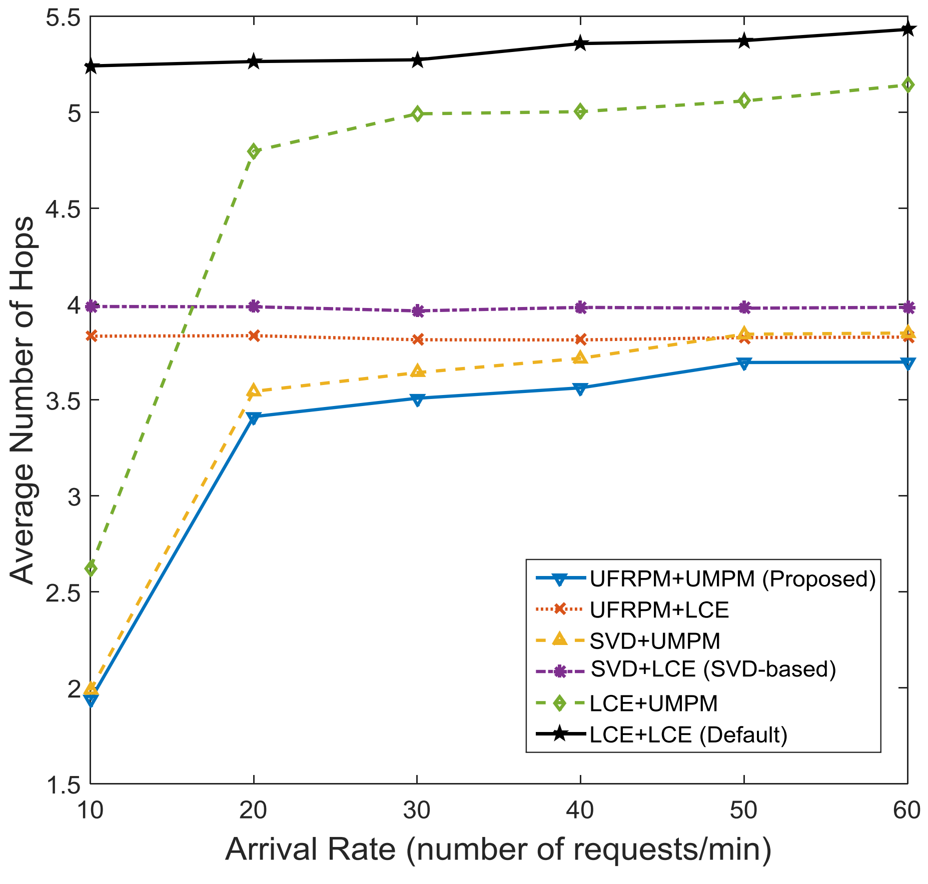


Fig. 6.15: Impact of arrival rate on the average number of hops

6.6 Summary

This chapter proposed a novel hierarchical proactive caching approach for ICN-based AV networks. By adopting the NMF technique to predict the user future ratings, users' future demands can be predicted by considering the historical popularity of videos and users' preferences, namely, the predicted ratings. Based on the predicted demands, the proposed proactive caching approach can cache videos at the video level at the core network nodes. Videos can be proactively cached at the edge nodes at the chunk level using the proposed UMPM before the AV users arrive, which can improve QoE in general.

The proposed proactive caching approach was evaluated in two scenarios: a highway scenario and a grid street scenario. The evaluation results from both scenarios show that the proposed approach is significantly more efficient compared to the existing approaches in terms of hit ratio and the average number of hops.

Chapter 7: Conclusion and Future Research

This chapter first summarizes the contributions that have been done for the thesis. Then, it provides several potential research directions.

7.1 Conclusion

In this thesis, the concept of ICN in-network caching has been used in conjunction with different types of networks to make a step forward towards next-generation networks. More specifically, the thesis has made contributions in the following five areas.

1) RPC for pure ICN networks: The proposed RPC approach tries to cache videos hierarchically, i.e., edge, core. More specifically, RPC tries to cache popular videos at the edge of the network, which is closer to the users. Consequently, the average video retrieval delay and the workload of on the video providers can be significantly reduced. Compared to ABC, RPC can reduce the publisher load ratio by 23%, and reduce the average number of hops by 26.7% when the cache size is 15 GB.
2) SDN-based caching approach for pure ICN networks: By leveraging the global view that is provided by the SDN controller, the SDN-based caching approach can make effective caching decisions. Extensive simulation results showed that the proposed SDN-based approach can achieve similar performance outcomes compared to that of the optimal solution in terms of the cache hit ratio (less than 2% difference) and the average number of hops (less than 4% difference) for content retrieval, while substantially reducing the computational complexity. Moreover, the simulation results also show that the SDN-based approach also

outperforms RPC in terms of hit ratio and the average number of hops. Hence, it is suitable for caching in dynamic networks.

3) ICN-based caching approach for ICN-5G networks: To overcome the frequent handoffs and shorter connection durations in 5G networks, an ICN-based caching approach was proposed in this thesis. The proposed approach makes caching decision based on the mobility of users and the popularity of requested videos. Popular videos which are requested by high mobility users will be cached at the CR which is connected to multiple BSs, while popular videos which are requested by low mobility users will be cached at the BS. Therefore, high mobility users can fetch the requested videos from a CR instead of the video provider, while low mobility users can fetch the requested videos from the BS directly. In this way, the QoE for mobile users in terms of average retrieval delay can be improved significantly (up to 23 *ms* compared to CDIC), and the network traffic (up to 57%) can be reduced as well.
4) LCC approach for ICN-IoT networks: To reduce the energy consumption of IoT devices, an IoT data lifetime-based cooperative caching (LCC) approach was proposed in Chapter 5. LCC caches IoT data at intermediate nodes (e.g., BS) based on the freshness of the IoT data and the data request rate. Hence, users can fetch the requested data from the intermediate nodes without waking up IoT devices. Consequently, IoT devices can spend more time in sleep mode to save energy, while user requests can still be served. Simulation results show that LCC can reduce the

energy consumption by up to 48.9% and the average number of hops by 31% compared to no caching.

5) Proactive caching approach for AV users: By using the NMF technique to predict user future ratings of videos and considering the historical popularity of videos, the proposed proactive caching approach can predict the future demands, which improves the caching effectiveness. Since routes are pre-planned, current location and velocity of the AVs can be easily obtained from the self-driving system of AVs. Therefore, the future position of AVs can be easily predicted. Based on these two predictions, the proposed proactive caching approach can proactively cache the video that AV users may like at the next RSU. Simulation results showed that the proposed proactive caching approach is more efficient than the existing approaches in terms of the hit ratio (up to 40% compared to the SVD-based approach in both the highway scenario and the grid street scenario) and the average number of hops (up to 62.5% compared to the SVD-based approach in the highway scenario, and up to 62% in the grid street scenario).

In summary, by using the in-network caching feature of ICN, the performance of ICN-based in-network caching can be improved for various potential next-generation network architectures.

7.2 Future Works

ICN is still an emerging research area. A number of interesting research problems and directions warrant future investigation. The following describes four such areas.

1. For the reactive caching approaches, caching videos at the chunk level is not considered in this dissertation. Further, chunks of the same video may have

different popularities. However, a video may have hundreds or thousands of chunks, which means that tracking and storing the popularity of each chunk in the traditional fashion is challenging and complex. Hence, how to track and store the popularity of each chunk in an efficient way is a new research problem.

2. For caching in ICN-IoT networks, one potential direction is to extend the proposed LCC approach for more complicated IoT scenarios, including machine-to-machine (M2M) communications, in which mobile devices can transmit IoT data with each other. Thus, the energy consumption of mobile devices needs to be considered as well.
3. For proactive caching approach, the NMF technique based on ratings is used to predict the users' future ratings on videos. However, users in real life may not rate a video, which will cause inaccurate NMF prediction ratings. A potential solution to this problem is that some other information, such as the completion rate of watching a video, can be used to generate ratings. Therefore, how to solve this problem in another research problem.
4. For proactive caching for AV users, this dissertation only considers the vehicle-to-infrastructure (V2I) communications. On the other hand, videos can be cached at AVs as well. As a result, vehicle-to-vehicle (V2V) communications should also be considered for caching. Consequently, how to make an efficient caching decision at AVs is a future research direction.
5. Since caching needs power, storage, memory and computation cost, it would be interesting to evaluate the caching performance with respect to the cost. Hence,

finding out the relationships between the cost and the benefit would be another potential research direction.

References

[1] I. Abdullahi, S. Arif, S. Hassan, “Survey on caching approaches in information centric networking”, Journal of Network and Computer Applications. 2015, 56:48-59.

[2] H. Ahlehagh and S. Dey, “Video caching in radio access network: Impact on delay and capacity”, Wireless Communications and Networking Conference (WCNC). IEEE, 2012, pp. 2276-2281

[3] M.R. Akdeniz, Y. Liu, M.K. Samimi, S. Sun, S. Rangan, T.S. Rappapoprt and E. Erkip, “Millimeter wave channel modeling and cellular capacity evaluation”, IEEE Journal on Selected Areas in Communications. 2014, 32(6): 1164-1179.

[4] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R.L. Aguiar and A.V. Vasilakos. “Information-Centric Networking for the Internet of Things: Challenges and Opportunities”, IEEE Network, 2016, 30(2): 92-100.

[5] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, “Energy conservation in wireless sensor networks: a survey”, Ad Hoc Networking. 2009, 7(3):537–568.

[6] Andreev, Sergey, Olga Galinina, Alexander Pyattaev, Jiri Hosek, Pavel Masek, Halim Yanikomeroglu, and Yevgeni Koucheryavy. "Exploring synergy between communications, caching, and computing in 5G-grade deployments." IEEE Communications Magazine. 2016, 54(8): 60-69.

[7] S. Arianfar, P. Nikander, and J. Ott, “On content-centric router design and implications”, Proceedings of the 2010 ACM Re-Architecting the Internet Workshop, 2010, p. 5.

[8] E. Baccelli, C. Mehlis, O. Hahm, T.C. Schmidt, M. Wählisch, “Information Centric Networking in the IoT: Experiments with NDN in the Wild”, Proceedings of the 1st ACM International Conference on Information-centric Networking (ICN). ACM, 2014, pp.77-86.

[9] Y. Bai, M.R. Ito, “Proactive resource allocation schemes”, Proceedings of the 2005 IEEE International Conference on Communications. IEEE, 2005, pp. 53-58.

[10] Y. Bao, X. Wang, S. Zhou, Z. Niu, “An energy-efficient client precaching scheme with wireless multicast for video-on-demand services”, Proceedings of the IEEE 18th Asia-Pacific Conference on Communications (APCC). IEEE, 2012, pp. 566-571.

[11] M.F. Bari, S.R. Chowdhury, R. Ahmed, R. Boutaba, B. Mathieu, “A survey of naming and routing in information-centric networks”, IEEE Communications Magazine. 2012, 50(12):44-53.

[12] E. Bastug, M. Bennis, “Living on the edge: The role of proactive caching in 5G wireless networks”, IEEE Communications Magazine, 2014, 8(52):82-89.

[13] E. Baştuğ, M. Kountheis, M. Bennis and M. Debbah, “On the delay of geographical caching methods in two-tiered heterogeneous networks”, Proceedings of the 2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, 2016, pp. 1-5.

[14] C. Bernardini, T. Silverston, and O. Festor, “MPC: Popularity-Based Caching Strategy for Content Centric Networks,” Proceedings of the 2013 IEEE International Conference on Communications (ICC), 2013, pp. 3619-3623.

[15] M. Bilal, S. G. Kang, “A cache management approach for efficient content eviction and replication in cache networks”, IEEE Access. 2017, 5: 1692-1701.

[16] B. Blaszczyszyn and A. Giovanidis, “Optimal geographic caching in cellular networks”, Proceedings of the IEEE 2015 International Conference on Communications. (ICC). IEEE, 2015, pp. 3358-3363.

[17] G. Carofiglio, L. Mekinda and L. Muscariello, “Lac: Introducing latency-aware caching in information-centric networks," Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks (LCN), IEEE, 2015, pp. 422-425.

[18] “CCNx project”, http://blogs.parc.com/ccnx/

[19] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, “I Tube, You Tube, Everybody Tubes: Analyzing the World’s Largest User Generated Content Video System”, Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC). ACM, 2007, pp. 1-14.

[20] Z. Chang, Y. Gu, Z. Han, X. Chen and T. Ristaniemi, “Context-aware data caching for 5G heterogeneous small cells networks”, IEEE International Conference on Communications (ICC). IEEE, 2016, pp. 1-6.

[21] L.E. Chatzieleftheriou, M. Karaliopoulos and I. Koutsopoulos, “Caching-aware recommendations: Nudging user preferences towards better caching performance”, Proceedings of the 2017 IEEE International Conference on Computer Communications, 2017, pp.1-9.

[22] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu and Y. Liu, “Joint Resource Allocation for Software-Defined Networking, Caching, and Computing”, IEEE/ACM Transactions on Networking. 2018, 26(1): 274-287.

[23] Y. Chen, M. Ding, J. Li, Z. Lin, G. Mao and L. Hanzo, "Probabilistic small-cell caching: Performance analysis and optimization", IEEE Transactions on Vehicular Technology. 2016, 66(5): 4341-4354.

[24] B. Chen, C. Yang, Z. Xiong. "Optimal caching and scheduling for cache-enabled D2D communications", IEEE Communications Letters. 2017, 21(5):1155-1158.

[25] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in- network caching for content-oriented networks", Proceedings of the 2012 IEEE International Conference on Computer Communications NOMEN Workshop, 2012, pp. 316-321.

[26] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim: An highly scalable ccn simulator", Proceedings of the 2013 IEEE International Conference on Communications (ICC). IEEE, 2013, pp. 2309-2314.

[27] Cisco Visual Networking Index: Forecast and Methodology, 2016–2021, September 15, 2017. Accessed on April 14, 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html#_Toc484813971

[28] Y. Cui, J. Song, M. Li, Q. Ren, Y. Zhang, X. Cai. "SDN-based big data caching in ISP networks", IEEE Transactions on Big Data. 2017, 4(3):356-367.

[29] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonides, J. Kurose, D. Towsley and R. Sitaraman. "On the complexity of optimal request routing and content caching in heterogeneous cache networks", IEEE/ACM Transactions on Networking. 2017, 25(3): 1635-1648.

[30] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson and H. Wymeersch. “Location-aware communications for 5G networks: How location information can improve scalability, latency, and robustness of 5G”, IEEE Signal Processing Magazine. 2014, 31(6): 102-112.

[31] FP7 PURSUIT Project, 2011. [Online]. Available: http://www.fp7-pursuit.eu.

[32] X. Ge, S. Tu, G. Mao, C.X. Wang and T. Han, “5G ultra-dense cellular networks”, IEEE Wireless Communications, 2016, 23(1): 72-79.

[33] Grouplens, “MovieLens Latest Datasets”, [Online]. Available: https://grouplens.org/datasets/movielens/

[34] G. Han, L. Liu, J. Jiang, L. Shu and G. Hancke, “Analysis of Energy-Efficient Connected Target Coverage Algorithms for Industrial Wireless Sensor Networks”, IEEE Transactions on Industrial Informatics, 2017, 13(1): 135-143.

[35] O. Hahm, E. Baccelli, T.C. Schmidt, M. Wählisch, C. Adjih, and L. Massoulié, “Low-power Internet of Things with NDN & Cooperative Caching”, Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN). ACM, 2017, pp.99-108.

[36] N.B. Hassine, R. Milocco, P. Minet, “ARMA based popularity prediction for caching in content delivery networks”, Wireless Days. IEEE, 2017, pp. 113-120.

[37] D.T. Hoang, D. Niyato, D.N. Nguyen, E. Dutkiewicz, P. Wang, Z. Han, “A Dynamic Edge Caching Framework for Mobile 5G Networks”, IEEE Wireless Communications, 2018 Aug 22(99):1-9.

[38] Z. Hu, Z. Zheng, T. Wang, L. Song and X. Li, “Caching as a Service: Small-Cell Caching Mechanism Design for Service Providers”, IEEE Transactions on Wireless Communications. 2016, 15(10): 6992-7004.

[39] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N.H. Briggs and R.L. Braynard, “Networking Named Content”, Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. ACM, 2009, pp.1-12.

[40] W. Jianping, C. Yong, L. Xing, and C. Metz, “4over6 for the china education and research network,” IEEE Internet Computing, 2006, 10(3): 80-85.

[41] R. Jmal, L. C. Ftheati. “An OpenFlow architecture for managing content-centric-network (OFAM-CCN) based on popularity caching strategy”, Computer Standards & Interfaces. 2017, 51:22-29.

[42] D. Kim and Y.B. Ko, “On-demand anchor-based mobility support method for named data networking”, Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT), IEEE, 2017, pp. 19-23.

[43] Y. Koren, R. Bell, C. Volinsky, “Matrix factorization techniques for recommender systems”, Computer. 2009, 42(8): 30-37.

[44] Y. Koren, R. Bell, C. Volinsky, “Matrix factorization techniques for recommender systems”, Computer, 2009, 42(8): 30-37.

[45] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, “Software-defined networking: A comprehensive survey”, Proceedings of IEEE. 2015, 103(1):14-76.

[46] D.K. Krishnappa, M. Zink, C. Griwodz and P. Halvorsen, "Cache-centric video recommendation: an approach to improve the efficiency of youtube caches", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 11(4): 48:1-48:20.

[47] N. Laoutaris, H. Che and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis", Performance Evaluation. 2006, 63(7): 609-634.

[48] N. Laoutaris, S. Syntila and I. Stavrakakis, "Meta algorithms for hierarchical web caches", Proceedings of 2004 International Conference on Performance, Computing and Communications, 2004, pp. 445-452.

[49] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning", Nature, 2015, 521(7553): 436-444.

[50] Z. Li, J. Lin, M.I. Akodjenou, G. Xie, M.A. Kaafar, Y. Jin and G. Peng, "Watching videos from everywhere: a study of the PPTV mobile VoD system", Proceedings of the ACM 2012 Internet Measurement Conference, 2012, pp. 185-198.

[51] Z. Li, G. Simon, "Cooperative caching in a content centric network for video stream delivery," Journal of Network and Systems Management. 2015, 23(3): 445-473.

[52] C. Li, L. Toni, J. Zou, H. Xiong and P. Frossard. "QoE-Driven Mobile Edge Caching Placement for Adaptive Video Streaming", IEEE Transactions on Multimedia. 2017, 20(4): 965-984.

[53] X. Li, X. Wang, K. Li, Z. Han and V. C. Leung. "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design", IEEE Transactions on Wireless Communications. 2017, 16(10):6926-6939.

[54] S. Li, J. Xu, M. Van Der Schaar and W. Li, “Popularity-Driven Content Caching”, Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM). IEEE, 2016, pp. 1-9.

[55] H. Li, C. Yang, X. Huang, N. Ansari and Z. Wang, “Cooperative RAN Caching based on Local Altruistic Game for Single and Joint Transmissions”, IEEE Communications Letters. 2016, 21(4): 853-856.

[56] C. Liang, Y. He, F. R. Yu and N. Zhao. “Enhancing QoE-Aware Wireless Edge Caching With Software-Defined Wireless Networks”, IEEE Transactions on Wireless Communications. 2017, 16(10): 6912-6925.

[57] C. Liang, F.R. Yu, X. Zhang. “Information-centric network function virtualization over 5G mobile wireless networks”, IEEE network. 2015, 29(3):68-74.

[58] Z. Liu, Y. Ji, X. Jiang, Y. Tanaka, “User-behavior Driven Video Caching in Content Centric Network”, Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking, 2016, pp. 197-198.

[59] W.X. Liu, J. Zhang, Z.W. Liang, L.X. Peng and J. Cai, “Content popularity prediction and caching for ICN: A deep learning approach with SDN”, IEEE access. 2018;6:5075-5089.

[60] “MATLAB”, https://www.mathworks.com

[61] C. Mbarushimana, A. Shahrabi. “Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks”, In 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07). IEEE. 2007, vol. 2, pp. 679-684.

[62] Z. Ming, M. Xu, and D. Wang, “Age-based cooperative caching in information-centric networks”, Proceedings of the 2012 IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2012, pp. 268-273.

[63] S. Misra, R. Ttheani, F. Natividad, T. Mick, N.E. Majd and H. Huang, “AccConF: An Access Control Framework for Leveraging In-network Cached Data in the ICN-Enabled Wireless Edge”, IEEE Transactions on Dependable and Secure Computing, 2017, 16(1):5-17.

[64] M.J. Miller, N.H. Vaidya, “A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio”, IEEE Transactions on mobile Computing, 2005, 4(3): 228-242.

[65] S. Muralidharan, A. Roy, N. Saxena, “MDP-IoT: MDP Based Interest Forwarding for Heterogeneous Traffic in IoT-NDN Environment”, Future Generation Computer Systems. 79 (2018): 892-908.

[66] “ndnSIM”, http://ndnsim.net/2.3/index.html

[67] M. E. Newman. “Power laws, Pareto distributions and Zipf's law”, Contemporary physics. 2005, 46(5): 323-51.

[68] Y. Nishiyama, M. Ishino, Y. Koizumi, T. Hasegawa, K. Sugiyama, and A. Tagami, “Thesis on routing-based mobility architecture for ICN-based cellular networks”, Proceedings of the 2016 IEEE International Conference on Computer Communications Workshop, April, 2016, pp. 467-472.

[69] D. Niyato, D.I. Kim, P. Wang, L. Song, “A Novel Caching Mechanism for Internet of Things (IoT) Sensing Service with Energy Harvesting”, Proceedings of the IEEE 2016 International Conference on Communications (ICC). IEEE, 2016, pp.1-6.

[70] A. Noor, G. Farhadi, A. Ito, M. Gerla, “Popularity-based partial caching for Information Centric Networks”, IEEE Ad Hoc Networking Workshop (Med-Hoc-Net), 2016, pp. 1-8.

[71] B.C. Ooi, K.L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. GAO, Z. Luo, A.K. Tung, Y. Wang and Z. Xie, “SINGA: A distributed deep learning platform”, Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015:685-688.

[72] H.A. Pedersen, and S. Dey, “Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing”, IEEE/ACM Transactions on Networking (TON). 2016, 24(2): 996-1010.

[73] J. Prados-Garzon, O. Adamuz-Hinojosa, P. Ameigeiras, J.J. Ramos-Munoz, P. Andres-Maldonado and J.M. Lopez-Soler, “Handover implementation in a 5G SDN-based mobile network architecture”, Proceedings of the 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). IEEE, 2016, pp. 1-6.

[74] J. Qiao, Y. He, and X.S. Shen, “Proactive caching for mobile video streaming in millimeter wave 5G networks”, IEEE Transactions on Wireless Communications, 2016, 15(10): 7187-7198.

[75] J. Quevedo, D. Corujo, and R. Aguiar, “Consumer Driven Information Freshness Approach for Content Centric Networking”, Proceedings of the 2014 IEEE

International Conference on Computer Communications Workshop (INFOCOM WKSHPS). IEEE, 2014, pp. 482-487.

[76] R. Ravindran, A. Chakraborti, S.O. Amin., A. Azgin and G. Wang, "ICN-5G: delivering ICN services over 5G using network slicing", IEEE Communications Magazine, 55(5), pp.101-107.

[77] A.B. Reis, S. Sargento, F. Neves, O.K. Tonguz. "Deploying roadside units in sparse vehicular networks: What really works and what does not", IEEE transactions on vehicular technology. 2013, 63(6):2794-2806.

[78] D. Rossi and G. Rossini. "Caching performance of content centric networks under multi-path routing (and more)", Technical Report, Telecom ParisTech, 2011, pp. 1-6.

[79] H. Salah and T. Strufe, "CoMon: An Architecture for Coordinated Caching and Cache-Aware Routing in CCN", Proceedings of the 2015 IEEE Consumer Communications and Networking Conference, 2015, pp. 663-670.

[80] A. Seetharam, "On Caching and Routing in Information-Centric Networks", IEEE Communications Magazine. 2018, 56(3): 204-209.

[81] Z.G. Sheng, S. Yang, Y.F. Yu, A.V. Vasilakos, J.A. McCann, and K.K. Leung, "2013. A Survey on the IETF Protocol Suite for the Internet of Things: Standards, Challenges, and Opportunities", IEEE Wireless Communications, 2013, 20(6): 91–98.

[82] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, "Collaborative filtering recommender systems", The adaptive web 2007, 291-324. Springer, Berlin, Heidelberg.

[83] P. Sermpezis, T. Giannakas, T. Spyropoulos and L. Vigneri, “Soft Cache Hits: Improving Performance through Recommendation and Delivery of Related Content”, IEEE Journal on Selected Areas in Communications, Special Issue on Caching for Communication Systems and Networks, 2018, 36(6):1300-1313.

[84] P. Si, H. Yue, Y. Zhang and Y. Fang, “Spectrum management for proactive video caching in information-centric cognitive radio networks”, IEEE Journal on Selected Areas in Communications. 2016, 34(8): 2247-2259.

[85] Y. Sun, S.K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M.A. Kaafar and S. Uhig, “Trace-driven analysis of ICN caching algorithms on video-on-demand workloads”, Proceedings of the 2014 ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT). ACM, 2014, pp. 363-376.

[86] SS. Tanzil, W. Hoiles, V. Krishnamurthy, “Adaptive approach for caching YouTube content in a cellular network: Machine learning approach”, IEEE Access. 2017(5): 5870-81.

[87] B. Tan and L. Massoulie, “Optimal content placement for peer-to-peer video-on-demand systems”, IEEE/ACM Transactions on Networking. 2013, 21(2): 566–579.

[88] T.X. Tran, A. Hajisami, P. Pandey and D. Pompili, “Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges”, IEEE Communications Magazine. 2017 Apr 14;55(4):54-61.

[89] N.S. Vo, T.Q. Duong and M. Guizani, “QoE-Oriented Resource Efficiency for 5G Two-Tier Cellular Networks: A FemtoCaching Framework”, Global Communications Conference (GLOBECOM). IEEE, 2016, pp. 1-6.

[90] S. Vural, N. Wang, P. Navaratnam, R. Tafazolli, “Caching Transient Data in Internet Content Routers”, IEEE/ACM Transactions on Networking. 2017, 25(2):1048-61.

[91] IBM, “ILOG CPLEX optimization studio”, [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimizatio-studio

[92] “video channel of Sina”, http://video.sina.com.cn

[93] J. Wang, “A survey of web caching approaches for the internet”, ACM SIGCOMM Computer Communication Review, 1999, 29(5): 36-46.

[94] S. Wang, J. Bi, J. Wu and A. V. Vasilakos. “CPHP: In-networking caching for information-centric networking with partitioning and hash-routing”, IEEE/ACM Transactions on Networking. 2016, 24(5): 2742-2755.

[95] Y. Wang, Y. Chen, H. Dai, Y. Huang, L. Yang, “A learning-based approach for proactive caching in wireless communication networks”, Proceedings of the IEEE 9th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE. 2017, pp. 1-6.

[96] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, “Cache in the air: exploiting content caching and delivery techniques for 5G systems”, IEEE Communications Magazine 2014, 52(2): 131-139.

[97] Y. Wang, M. Ding, Z. Chen and L. Luo, “Caching Placement with Recommendation Systems for Cache-Enabled Mobile Social Networks”, IEEE Communications Letters, 21(10): 2266-2269.

[98] Y. Wang, M. Ding, Z. Chen and L. Luo, “Caching Placement with Recommendation Systems for Cache-Enabled Mobile Social Networks”, IEEE Communications Letters, 21(10): 2266-2269.

[99] Y. Wang, Z. Li, G. Tyson, S. Uhlig and G. Xie. “Design and evaluation of the optimal cache allocation for content-centric networking”, IEEE Transactions on Computers. 2016, 65(1):95-107.

[100] Z. Wang, H. Li, Z. Xu, “Real-world traffic analysis and joint caching and scheduling for in-RAN caching networks”, Science China Information Sciences. 2017, 60(6): 062302.

[101] Z. Wang, H. Li and C. Yang, “Feasibility analysis and self-organizing algorithm for RAN cooperative caching”, Wireless Communications and Networking Conference (WCNC). IEEE, 2016, pp. 1-6.

[102] R. Wang, X. Peng, J. Zhang and K. B. Letaief, “Mobility-Aware Caching for Content-Centric Wireless Networks: Modeling and Methodology”, IEEE Communications Magazine, 2016, 54(8): 77-83.

[103] L. Wang, G. Tyson, J. Kangasharju, J. Crowcroft. “Milking the cache cow with fairness in mind”, IEEE/ACM Transactions on Networking. 2017, 25(5): 2686-2700.

[104] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, A. Wolisz, “Industrial Wireless IP-Based Cyber-Physical Systems”, Proceedings of the IEEE, 2016, 104(5): 1025–1038.

[105] S. Wilk, D. Schreiber, D. Stohr and W. Effelsberg, “On the effectiveness of video prefetching relying on recommender systems for mobile devices”, Proceedings of

the 13th IEEE 2016 Consumer Communications & Networking Conference (CCNC), 2016, pp. 429-434.

[106] J. Xie, R. Xie, T. Huang, Y. Liu, J. Liu and F.R. Yu, "Caching Resource sharing in radio access networks: a game theoretic approach", Journal of Zhejiang University-SCIENCE. 2016: 11-08.

[107] Y. Xu, S. Ma, Y. Li, F. Chen, S. Ci, "P-CLS: a popularity-driven caching location and searching approach in content centric networking", Proceedings of the IEEE 34th International Performance Computing and Communications Conference (IPCCC), 2015, pp. 1-8.

[108] C. Xu, P. Zhang, S. Jia, M. Wang and G.M. Muntean, "Video Streaming in Content-Centric Mobile Networks: Challenges and Solutions", IEEE Wireless Communications. 2017, 24(5): 157-165.

[109] H. Yan, D. Gao, W. Su, CH. Foh, H. Zhang and AV. Vasilakos, "Caching Strategy Based on Hierarchical Cluster for Named Data Networking", IEEE Access. 2017(5): 8433-8443.

[110] J. Yin, L. Li, H. Zhang, X. Li, A. Gao, Z. Han, "A prediction-based coordination caching scheme for content centric networking", Proceedings of the IEEE 27th Wireless and Optical Communication Conference (WOCC). IEEE, 2018, pp. 1-5.

[111] L. Yu, X. Sun, Z. Huang, "Robust spatial-temporal deep model for multimedia event detection", Neurocomputing, 2016, 213: 48-53.

[112] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, X.S. Shen. "Toward efficient content delivery for automated driving services: An edge computing solution", IEEE Network. 2018, 32(1):80-86.

[113] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos and B. Zhang, “Named data networking”, ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66-73.

[114] X. Zhang, Y. Cao, “A Cooperation-Driven ICN-based Caching Approach for Mobile Content chunk Delivery at RAN”, IEEE International Wireless Communications and Mobile Computing Conference, 2017, pp. 1437-1442.

[115] G. Zhang, Y. Li and T. Lin, “Caching in information centric networking: A survey”, Computer Networks. 2013, 57(16): 3128-3141.

[116] X. Zhang, Q. Zhu, “Hierarchical Caching for Statistical QoS Guaranteed Multimedia Transmissions over 5G Edge Computing Mobile Wireless Networks”, IEEE Wireless Communications, 2018, 25(3):12-20.

[117] H. Zhao, Q. Zheng, W. Zhang, B. Du, and H. Li. “A segment-based storage and transcoding trade-off strategy for multi-version VoD systems in the cloud”, IEEE Transactions on Multimedia. 2017, 19(1):149-159.

[118] H. Zhu, Y. Cao, Q. Hu, W. Wang, T. Jiang, Q. Zhang. “Multi-Bitrate Video Caching for D2D-Enabled Cellular Networks”, IEEE MultiMedia. 2018, 26(1):10-20.

[119] Z. Zhao, W. Chen, X. Wu, P.C. Chen and J. Liu, “LSTM network: a deep learning approach for short-term traffic forecast”, IET Intelligent Transport Systems, 2017, 11(2): 68-75.

[120] N. Zhao, X. Liu, F.R. Yu, M. Li and V.C. Leung, “Communications, caching, and computing oriented small cell networks with interference alignment”, IEEE Communications Magazine. 2016, 54(9): 29-35.