MOBILE CONTENT DELIVERY IN INFORMATION-CENTRIC NETWORK

by

FEIXIONG ZHANG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Yanyong Zhang and Dipankar Raychaudhuri

And approved by

New Brunswick, New Jersey

October, 2016

ABSTRACT OF THE DISSERTATION

Mobile Content Delivery in Information-Centric Network

By FEIXIONG ZHANG

Dissertation Director:
Yanyong Zhang and Dipankar Raychaudhuri

Nowadays, the Internet usage is shifting towards information distribution and retrieval, with mobile data access becoming the norm. The mismatch between the dominant information-centric usage pattern and the location-based Internet architecture results in inefficient content services that heavily rely on application layer overlays. Therefore, the information-centric network (ICN) is proposed as a clean-slate network architecture to support mobile content delivery. ICN treats content as the first-class entity, and identifies content by its name at the network layer. The direct addressability of content in ICN facilitates content-oriented services. In this dissertation, we focus on two ICN architectures, i.e., content-centric networking (CCN) and MobilityFirst (MF), and investigate the corresponding transport control and content caching techniques which are crucial to content delivery.

Content-centric networking (CCN) adopts a receiver-driven, hop-by-hop transport approach that facilitates in-network caching, which in turn leads to multiple sources and multiple paths for transferring content. We propose novel transport protocols, namely CHoPCoP and pCHoPCoP, to satisfy the requirements of CCN. Our transport protocols utilize explicit congestion control to cope with the multiple-source multiple-path situation and provides multi-homing support for CCN. Our evaluation of CHoP-CoP/pCHoPCoP on the ORBIT testbed shows that the proposed transport protocols can effectively deal with congestion in the CCN environment and improve data transmission performance.

Caching is widely used to disseminate content and offload content requests. We move a step further by proposing to have a separate popularity based cache and a prefetch buffer at the network edge to capture both long-term and short-term content access patterns, and use network-level mobility prediction to guide the prefetch. The framework, called EdgeBuffer, is discussed in the context of MobilityFirst architecture. Our simulation effort of EdgeBuffer framework demonstrates a significant cache hit ratio improvement at the edge. Then, we take a step back and compare several different content caching and request forwarding schemes in the general ICN context. Our investigation is intended to better understand whether pervasive caching and nearest replica routing could each bring significant benefits. The evaluation shows that pervasive caching is not better than edge caching; compared to pervasive caching, nearest replica routing brings more benefits, especially in a large network. We then propose a network architecture built upon MobilityFirst which adopts edge caching and approximates nearest replica routing.

Finally, we present our system prototype and a field trial experiment of the MobilityFirst architecture. We design and develop a satellite-based video delivery system built on MobilityFirst, which takes advantage of the satellite network to efficiently distribute content to large area and utilizes edge caching to effectively offload requests. The whole video delivery system is implemented and examined on ORBIT testbed. The field trial validates the feasibility of real world deployment and the benefits it brings to practical use cases.

Acknowledgements

Throughout my graduate school journey at WINLAB, I have received tremendous guidance and help from many people. I would like to express my gratitude to all these people who have made my graduate experience priceless and this dissertation possible.

First of all, I would like to thank my advisors: Prof. Yanyong Zhang and Dipankar Raychaudhuri. During my six years Ph.D. study, they have provided me with great support, encouragement and guidance. Prof. Zhang has guided me through my whole Ph.D. process, and worked with me at every step of my dissertation progress. She has taught me how to conduct research and think in a critical way. In addition to technical aspect, I receive plenty of suggestions from her on career development, teamwork collaboration, communication skills, etc. She has impacted me significantly on my values and life path, and I could not thank her enough. Prof. Ray has always been very encouraging and discussed with me on any problem no matter how busy he is. I have learned a lot from his broad knowledge and exceptional insight. His invaluable experience in industry motivates me to conduct system research with real world impact.

I would like to express my thanks to Prof. K.K. Ramakrishnan. He continuously works with me to improve my Ph.D. research projects and provides me with lots of constructive suggestions. His great passion in research always motivates me to improve my work and be more productive.

I have worked with several industrial collaborators during my Ph.D., especially Dr. Alex Reznik and Hang Liu from InterDigital. My work on CHoPCoP/pCHoPCoP is closely guided by them and we had a large amount of discussion on its design and evaluation. Their insightful knowledge ignited my interest in networking related research.

I would like to thank Prof. Richard Martin for his time to serve on my committee and give me lots of suggestions on improving the dissertation.

I'm grateful to have lots of excellent colleagues at WINLAB. Prof. Roy Yates's insightful comments help me think critically and polish my work. Ivan Seskar gives me enormous help on ORBIT experiments and I learn lots of practical knowledge from him. Chenren Xu has given me lots of suggestions on research and career development. I worked with Jiachen Chen during my final year here, and collaboration with him is rather fruitful. I would also like to thank Francesco Bronzino for his help on MobilityFirst prototype.

I also wish to thank many collaborators, colleagues and friends, who have worked with me and influenced my views and efforts during my Ph.D., including Kiran Nagaraja, Rich Howard, Yi Hu, Kai Su, Shreyasee Mukherjee, Shweta Sagari, Sugang Li, Bernhard Firner, Shridatt Sugrim, Michael Sherman, etc. I would also like to thank all my friends who have accompanied me along the journey.

Finally, I would like to thank my family for their everlasting love, patience and support.

Dedication

To my parents Bixin Zhang and Wanxiu Hu, for their love, continuous support and encouragement.

Table of Contents

Abstract ii

Acknowledgements iv

Dedication vi

List of Tables xi

List of Figures xii

1. Introduction 1

1.1. Request for Information-Centric Network 1

1.2. Dissertation Contribution 3

1.2.1. Transport control in content-centric networking 4

1.2.2. Caching and prefetching in MobilityFirst 5

1.2.3. Comparing caching and request forwarding strategy in ICN . . . 5

1.2.4. Prototype and field trial of MobilityFirst architecture 6

1.3. Organization of Dissertation 7

2. Background of ICN 8

2.1. Content-Centric Network Architecture 9

2.1.1. CCN Name and Packet 9

2.1.2. CCN Node Model 10

2.2. MobilityFirst Architecture 11

2.2.1. Overall Network Architecture 11

2.2.2. Content-Oriented Service 12

2.2.3. Robust Data Delivery 13

3. Transport Protocol for Content-Centric Network with Explicit Congestion Control . . . 15

3.1. Introduction . . . 15

3.2. Related Work on Transport Control . . . 17

3.3. CHoPCoP Design . . . 19

3.3.1. Explicit Congestion Signalling . . . 20

3.3.2. Fair Share Interest Shaping (FISP) . . . 21

3.3.3. AIMD Based Receiver Interest Control (RIC) . . . 23

3.4. pCHoPCoP: parallel CHoPCoP . . . 24

3.4.1. Design Framework . . . 25

3.4.2. Protocol Details . . . 26

3.5. Implementation . . . 27

3.6. Evaluation . . . 29

3.6.1. Experimental Setup . . . 29

3.6.2. A Single-Source, Single-Destination Benchmark Scenario . . . 29

The Effectiveness of REM and RIC . . . 30

The Effectiveness of Fair Share Interest Shaping (FISP) . . . 31

The Sensitivity of CHoPCoP Parameters . . . 31

3.6.3. A Multi-Source, Single-Flow Scenario . . . 34

The Impact of Timeout Parameters at the Receiver . . . 35

The Effectiveness of REM and RIC . . . 36

3.6.4. A Multi-flow Scenario . . . 37

Fairness between the Flows . . . 37

Comparison of FISP with Quota-based Interest Shaping . . . 37

3.6.5. A Multi-Source, Multi-Path Scenario . . . 38

Throughput Comparison . . . 38

A Closer Look at CHoPCoP . . . 38

3.6.6. Multi-homing Scenarios . . . 39

3.7. Conclusion . . . 40

4. Caching and Prefetching Content at the Edge in the MobilityFirst Future Internet Architecture 48

4.1. Introduction 48

4.2. EdgeBuffer: Content Caching and Prefetching at the Edge 52

4.2.1. Popularity-based Caching 52

4.2.2. Mobility Prediction and Prefetching 53

Spatial and Temporal Prediction 54

4.2.3. Content Prefetching Protocol 55

4.3. Evaluation 57

4.3.1. Simulation Setup and Workload Generation 59

4.3.2. Prediction Accuracy 59

4.3.3. Hit Ratio at the Edge Buffers 62

Comparison of Four Edge Caching Strategies 62

Size Allocation between Cache and Prefetch Buffer 65

Prefetching More Chunks 66

Network Transition Probability Threshold 68

4.4. Related Work 68

4.4.1. Caching and Prefetching 68

4.5. Concluding Remarks 69

5. Edge Caching and Nearest Replica Routing in Information-Centric Networking 70

5.1. Introduction 70

5.2. Background of ICN Caching 72

5.3. Design Space For ICN Caching 73

5.4. Comparison of ICN Designs 74

5.4.1. Simulation setup 74

5.4.2. Small-scale network scenario 76

5.4.3. Large-scale network scenario 77

5.5. Approximate Nearest Replica Routing 78

5.5.1. Feasibility at Internet Scale 79

GNRS design 79

Update traffic overhead 79

5.5.2. Effectiveness 80

5.6. Related work 81

5.6.1. Pervasive caching and edge caching 81

5.6.2. Separating control plane from data plane in ICN 81

5.7. Conclusion 82

6. System Prototype and Field Trial of the MobilityFirst Future Internet Architecture 85

6.1. MobilityFirst Prototype Overview 85

6.2. Video Distribution System over Satellite 86

6.3. Core Components and System Prototype 88

6.3.1. Content naming service 89

6.3.2. s-CDN provider and client 89

6.3.3. Satellite-aware multicast routing 91

6.3.4. Satellite transport protocol 92

6.3.5. Edge cache 92

6.4. Field Trial Experiment and Future Work 94

7. Conclusions 96

References 98

Vita 104

Education 104

Work Experience 104

Publications 104

List of Tables

3.1. FISP leads to better delivery ratio and throughput. Without FISP, throughput decreases significantly when queue overflows 32

3.2. Results under different chunk size. Each packet is 1KB in size. 33

3.3. Results under different REM smoothing parameter. Chunk size is 32KB. 33

3.4. Results under different parameter (q_{min}, q_{max}) in the REM mark probability function. Chunk size is 32KB, $P_{max} = 0.002$. Note that q_{max} should be smaller than router buffer capacity C. 34

3.5. Results under different parameter P_{max} in the REM mark probability function. Chunk size is 32KB, $q_{min} = 0.1C$, and $q_{max} = 0.3C$. 34

3.6. Results under different weight parameter γ in the FISP's queue requirement equation. 35

3.7. Results under different parameter (Q_{min}, Q_{max}) in the FISP delay probability function. Chunk size is 32KB, $P_0 = 0.3$, and $Q_{release} = 0.2C$. . . 35

3.8. Results under different parameter P_0 in the FISP delay probability function. Chunk size is 32KB, $Q_{release} = 0.2C$, $Q_{min} = 0.6C$, and $Q_{max} = 0.9C$. 36

3.9. In CHoPCoP, a large timeout value (with large δ values) can eliminate false timeouts. 36

3.10. Throughput comparison among different transport protocols. 38

List of Figures

1.1. Change IP layer in Internet to named object in ICN 3

2.1. Content delivery example in ICN. Content is requested through name at the network layer, and is cached at ICN router when the data returns. Subsequent request is satisfied from a nearby cache. 9

2.2. Packet processing model of CCN node. 10

2.3. Name-locator separation and mapping in MobilityFirst. 11

2.4. MobilityFirst packet structure with key fields. 13

3.1. Functional modules of CHoPCoP content provider, router and receiver (we omit inbound queues in router for simplicity) 19

3.2. Mark probability function for REM 21

3.3. Delay probability function for FISP 21

3.4. pCHoPCoP protocol architecture. The pCHoPCoP receiver consists of a pCHoPCoP engine and multiple CHoPCoP controllers. 25

3.5. A three node baseline topology, consisting of a source (A), a router (B) and a receiver (C). 30

3.6. The performance results for a 3-node baseline topology. ICP and HR-ICP behave very similarly, we thus use the ICP curve to represent both schemes. 41

3.7. Boxplot of router queue size with and without FISP under different CIR. At a low CIR, both schemes perform similarly. At a high CIR, FISP nicely controls the router queue size, and thus avoids queue overflow. Without FISP, the router queue overflows when the CIR is above 160 Interests per second. 42

3.8. A topology that consists of two sources (A and B) that serve the same request from D. 42
3.9. Comparison of CHoPCoP, ICTP, ICP, and HR-ICP when two sources serve the same content flow. We use the ICP curve to represent both ICP and HR-ICP because of their similar performance. 43
3.10. Multiple flow topology where D and E request two different files from the same source A. 44
3.11. CHoPCoP can efficiently utilize the network resource when only one flow exists. When we have multiple flows, it ensures fairness among them. . . 44
3.12. Throughput comparison of FISP with HbH. Receiver D has the same throughput for both schemes. 45
3.13. A larger-scale topology with two sources (A and B) and two receivers (F and G). Link EF is the bottleneck link between A/B and F, while link CD is the bottleneck between A/B and G. 45
3.14. The receiving rate for the two receivers. F initiates content flow C_1 (with source A) while G initiates content flow C_2 with two sources A and B. . 45
3.15. Three multi-homing topologies. Each consists of two sources (A and B), and the receiver E requests content from them using both of its interfaces. 46
3.16. The throughput of the two CHoPCoP pipes and the aggregated pCHoPCoP at the three multi-homing topologies. 46
3.17. The receiver window size at each of the two CHoPCoP pipes for the multi-homing topologies in Figure 3.15. 47
4.1. EdgeBuffer: mobility prediction and content caching/prefetching framework in MobilityFirst 51
4.2. Residence times for the same taxi at the same AP vary significantly. . . 53
4.3. Sequence diagram when client A transits to AP2 from AP1 55
4.4. Content prefetching protocol 57

4.5. Spatial prediction accuracy when picking different number of next AP(s). Predicting next AP can achieve high accuracy with a small K value using our network-level prediction. 60

4.6. Histograms of residence times of an AP. The smooth curve is a log-normal density distribution. 61

4.7. We evaluate temporal prediction error when the residence time history table uses different history sample size. 62

4.8. Cache hit ratio for four edge caching strategies (with SF-100 workload). 63

4.9. For popCache+userPredict and popCache+netPredict, we break down the total cache hits into hits from cache and hits from prefetch buffer. Since the contributions from the cache are almost identical, we only show the cache contribution for popCache+netPredict here. 64

4.10. Hit ratio with different prefetch buffer sizes. Note that an AP has a total of 10GB storage. 65

4.11. Hit ratio at different padding level. 66

4.12. Hit ratio at different transition probability threshold. We also calculate the average value of K at each threshold. 67

5.1. 4-level binary tree with redundant links. Dashed links are redundant links, each of which is present with probability μ. 74

5.2. A core-tree topology example with four core nodes and their corresponding access tree. 75

5.3. Cache performance at different cache budget ratio, content popularity distribution parameter, and redundant link probability for the tree topology, with default value as 0.1%, 1, and 0, respectively. The results show that the performance difference between pervasive caching design and edge caching design is small. 76

5.4. Abilene topology. 78

5.5. Cache performance at different inter link to intra link delay ratio for the large-scale core-tree topology. The cache budget ratio and Zipf distribution parameter is 0.1% and 1. EDGE-NRR-NORM performs the best among all designs. 83
5.6. The effectiveness of our edge cached MF scheme compared with ideal EDGE-NRR at different cache budget ratio, Zipf distribution parameter, and redundant link probability, with default value as 0.1%, 1, and 0, respectively. The results show that our scheme performs very close to ideal EDGE-NRR. 84
6.1. Video distribution system with satellite overlay. The central station is connected to the satellite CDN (s-CDN) provider, and has both satellite medium and terrestrial medium to deliver content. 87
6.2. Prototype of the video distribution system using MobilityFirst. Reliable transport is the hop-by-hop chunk transfer protocol in MobilityFirst, while unreliable transport is specifically designed for the satellite medium. 88
6.3. s-CDN request scheduling engine. The content request with highest rank is scheduled to be processed next. 89
6.4. Data processing flow at the central station. Satellite-aware multicast routing is used to determine whether satellite medium or terrestrial medium is used for forwarding. 91
6.5. Sequence diagram of the satellite transport protocol. 92
6.6. Cache service prototype in MobilityFirst. 94
6.7. Field trial experiment scenario on ORBIT testbed. 94

Chapter 1

Introduction

1.1 Request for Information-Centric Network

The Internet was created in the 1960s and 1970s when telephony is the only successful, global-scale communication system. Following the principle of telephony, the Internet is designed around a point-to-point communication model between two stationary end hosts. Each end host is identified by its network locator, or IP address, and the communication is carried out between the two IP addresses. Built around point-to-point communication model, the Internet is essentially a *location-based* network system.

However, over the course of a few decades, Internet usage has evolved to be dominated by information distribution and retrieval. Internet users care about the access to *information* (e.g., news, video, social feeds, etc), and disregard the exact location or host that they communicate with. More recently, Internet access from mobile devices has grown dramatically in popularity, and it is anticipated that mobile user generated traffic will exceed that from fixed devices in the next following years, as reported in Cisco visual networking index [22]. These two important trends in Internet usage motivate consideration of efficient mobile content delivery in the Internet.

The *location-based* Internet architecture does not inherently support *information-centric* usage pattern and *mobility-oriented* user access. Any information-centric content service requires to first identify the two endpoints, and then uses IP as a point-to-point communication channel to carry out the transmission. As a consequence, existing content service is implemented as an overlay on top of IP (e.g., content delivery networks (CDN) [26], peer-to-peer (P2P)) and uses indirection-based approach (e.g., Home Agent and Foreign Agent in mobile IP [59]) for mobility management.

For either CDN or P2P, each content request needs to be resolved to the IP address(es) of nearby server(s) or peer(s) through complicated application-layer protocol, and then the client builds an IP connection to each of them. The performance of application-overlay based content delivery is heavily restricted by the underlying IP point-to-point communication model. In addition, indirection-based mobility management can potentially cause triangle routing which incurs unnecessary large delay for content delivery, and the indirection agent could be a single point of failure. Therefore, current techniques require complicated request redirection and processing, leading to longer data retrieval latency and wasted network capacity, thus are not efficient and scalable for mobile content delivery.

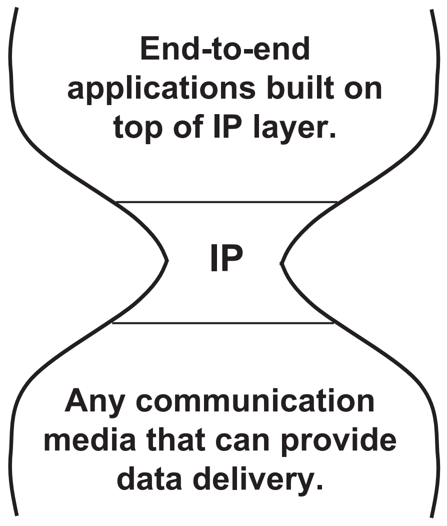
An ideal network system to support mobile content delivery would have the following requirements:

- **Information-centric communication model**. With content-driven Internet usage, it is necessary to use the content as a first-class primitive in packet transmission, i.e., content semantics is directly expressed and addressed at the network layer, same as IP. To support such content semantics, a network object (including host, network, and content) is identified by its name instead of a specific network address. Thus the communication with any network object (e.g., a content) appears to be no different than with a fixed endpoint.

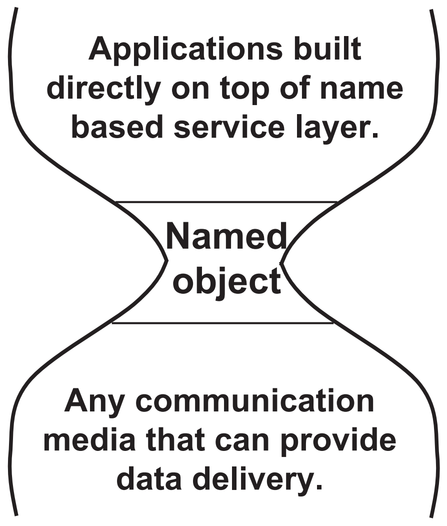
- **Inherent network support of caching**. A named object like a content could be potentially cached at multiple locations, and the network is able to provide the content from the 'best' location when given the content name. The network thus becomes a network of caches, instead of a simple interconnection of nodes.

- **Mobility support**. Because of mobility, a network object and its actual locations are becoming uncorrelated. As a result, it becomes necessary to identify an object via a permanent name and separate the name from its locations, while the network provides the support for dynamic binding of names to network addresses/locators.

To meet with these requirements, the idea of *information-centric network* (ICN) [1, 2] has been intensively discussed recently to better support mobile content-oriented


End-to-end applications built on top of IP layer.
IP
Any communication media that can provide data delivery.

(a) IP based network stack


Applications built directly on top of name based service layer.
Named object
Any communication media that can provide data delivery.

(b) Name based network stack

Figure 1.1: Change IP layer in Internet to named object in ICN

services from a clean-slate network architecture perspective. Several information-centric architectures (such as MobilityFirst [49,62], CCN [40,52], XIA [36]) are also developed based on such concept. As illustrated in Figure 1.1, in ICN, a network object (e.g., a host or content) is identified by its unique name/identifier instead of a network address, and a network entity (e.g., host) can directly distribute or request content through content name at the network layer. The direct addressability of content along with the separation of name and address provides inherent support of mobile content delivery and supports much more efficient content-oriented services.

1.2 Dissertation Contribution

Content delivery in ICN has attracted enormous research efforts in the community, including naming, routing, transport protocol, caching, etc. This dissertation, in particular, investigates several key design components (like transport control, caching, and prefetching) in two representatives of the ICN architectures, i.e., content-centric networking (CCN) [40,52] and MobilityFirst (MF) [49,62].

1.2.1 Transport control in content-centric networking

Content-centric application has different communication characteristics from traditional end-to-end communication model. First, such application is naturally receiver-driven, since the content receiver needs to issue the requests for each segment of content, and the receiver is in the position of better understanding it's own application requirement and content consumption rate. Second, hop-by-hop transfer is desired for content related transport, because content files can be cached along the route to improve throughput, and a router could retrieve content from its cache in addition to simply forwarding packets. Moreover, since a specific content is often widely disseminated and cached in the network, a content flow may have multiple sources – i.e., one content chunk originates from source A, while the next chunk might originate from source B. This implies that content delivery might be carried out as multi-source/multi-path transfer. These features of content delivery are sufficiently distinct from a traditional end-to-end host-based model that a new transport approach is called for.

We propose a new transport scheme under the CCN architecture, called the Chunk-switched Hop Pull Control Protocol (CHoPCoP), that is well-adapted to the characteristics of content delivery. Protocol design innovations of CHoPCoP include a random early marking (REM) scheme that explicitly signals network congestion, and a per-hop fair share Interest shaping algorithm (FISP) and a receiver Interest control method (RIC) that regulate the Interest rates at routers and the receiver respectively. In addition, a parallelized version of CHoPCoP, *pCHoPCoP*, is designed to support bandwidth aggregation on multi-homed clients. We have implemented CHoPCoP/pCHoPCoP on the ORBIT testbed and conducted experiments under various network and traffic settings. The evaluation shows that CHoPCoP can effectively deal with congestion in the multipath environment, and pCHoPCoP can maximize the network utilization for a multi-homed client.

1.2.2 Caching and prefetching in MobilityFirst

Caching has been widely utilized for disseminating content and offloading content requests. However, the effectiveness of traditional caching becomes limited whenever requests enter the long tail of heavy-tailed object distribution [10]. In this work, we propose to take advantage of the storage available at wireless access points to bring content closer to mobile devices, and have a separate popularity based cache and a prefetch buffer at the network edge to capture both long-term and short-term content access patterns, hence extending the caching capability and improving the overall cache hit ratio. Further, we point out that it is insufficient to rely on a device's past history to predict when and where to prefetch, especially in urban settings; instead, we propose to derive a prediction model based on the aggregated network-level statistics.

We discuss the proposed mobile content caching/prefetching method in the context of the MobilityFirst future Internet architecture. In MobilityFirst, when mobile clients move between network attachment points (e.g., Wi-Fi access points), their network association records are logged by the network, which then naturally facilitates the network-level mobility prediction. Through detailed simulations with real taxi mobility traces, we show that such a strategy is more effective than earlier schemes in satisfying content requests at the edge (higher cache hit ratios), leading to shorter content download latencies. Specifically, the fraction of requests satisfied at the edge increases by a factor of 2.9 compared to a caching only approach, and by 45% compared to individual user-based prediction and prefetching.

1.2.3 Comparing caching and request forwarding strategy in ICN

Information-centric networking (ICN) promises to improve content-oriented services by enabling in-network caching and supporting optimal content request forwarding. To achieve this goal, different content caching and request forwarding schemes are proposed in multiple ICN architectures. However, they differ from each other in whether pervasive caching is adopted, and whether nearest-replica routing is supported. It becomes an urgent need to better understand the pros and cons of different content caching and request forwarding design choices in ICN: whether pervasive caching and nearest replica routing could each bring significant benefits? is there an ICN architecture that achieves most of the gain and is relatively easy to deploy?

In this work, we evaluate these options through extensive simulation over both single domain tree network and multi-domain network scenarios. Our evaluation shows that pervasive caching may not bring about substantial benefits compared with simple edge caching configuration; on the other hand, nearest replica routing is beneficial especially in a large network, since it ensures that a request is served from the "best" location. Based on the observations, we propose a caching system built upon MobilityFirst architecture, in which edge caching is adopted, and nearest replica routing is supported through a global name resolution service. Additional evaluation shows that our proposal retains the advantages of ICN (e.g., requests served from "best" location), and is feasible to deploy globally.

1.2.4 Prototype and field trial of MobilityFirst architecture

We have implemented a proof-of-concept prototype of the MobilityFirst future Internet architecture. The prototype follows the name-based communication model and implements a distributed name resolution service that manages the name to address mapping. In addition, the prototype includes a Click-based router [42], and the host-side protocol stack and network API. Content caching is implemented as a plug-in module to the Click router to support in-network content caching.

To validate the whole architecture, we designed and developed a video distribution system based on the MobilityFirst prototype. We worked with SES S.A., a communication satellite operator, to develop such a system that utilizes satellite multicast and MobilityFirsts caching design to facilitate video delivery. Our system combines the benefits of satellite and terrestrial network to efficiently deliver content to edge cache and clients. The field trial examines that MobilityFirst could well support practical use cases and allow for potential better network performance.

1.3 Organization of Dissertation

The rest of the dissertation is organized as follows. Chapter 2 presents the background of ICN, and provides the details of CCN and MobilityFirst architecture. In Chapter 3, we present our transport protocol CHoPCoP/pCHoPCoP for the content-centric architecture. Our transport protocol provides the support of explicit congestion control and multi-homing, and adapts to the new characteristics of CCN. Next, we talk about our caching and prefetching framework under the MobilityFirst future Internet architecture in Chapter 4. The framework utilizes the distributed storage at the wireless edge to facilitate mobile content delivery. The comparison study of caching and request forwarding schemes is presented in Chapter 5. In Chapter 6, we talk about the prototype and field trial efforts for the MobilityFirst architecture. Finally, Chapter 7 concludes the dissertation with future research directions.

Chapter 2

Background of ICN

Information-centric network (ICN) is proposed to inherently support content-oriented service in the network architecture. ICN is built upon a name-based service model: name is decoupled from locations, and network communication is based on name in contrast to today's host-centric abstraction. Thus, the communication with any network object appears to be no different than that if it was a fixed endpoint, resulting in a location-independent communication paradigm. In addition, ICN naturally supports caching where ICN router is equipped with cache storage to serve content requests. As illustrated in Figure 2.1, in ICN paradigm, the client requests content directly by name, and the network then retrieves the content to the client. When the content returns, on-path nodes could cache the content, thus subsequent requests might be satisfied from these caches.

Several information-centric architectures (such as MobilityFirst [49, 62], CCN [40, 52], XIA [36]) are developed based on ICN concept. These ICN architectures differ from each other in how they realize name-based service, especially in the presence of mobility. Some of these architectures (such as CCN [40], TRIAD [35], and ROFL [12]) propose a name-based routing approach which includes an object name into the routers' forwarding table and forwards packets directly based on the name, without using network-level addresses. On the other hand, some other architectures (like MobilityFirst [49], XIA [36], HIP [50] and AIP [4]) places object names outside of the routing plane and uses a name resolution service to translate names to addresses.

In this dissertation, we investigate two representatives of the ICN architectures in detail, i.e., content-centric networking (CCN) [40, 52] and MobilityFirst (MF) [49, 62]. We present the basics of these two architectures in the chapter.

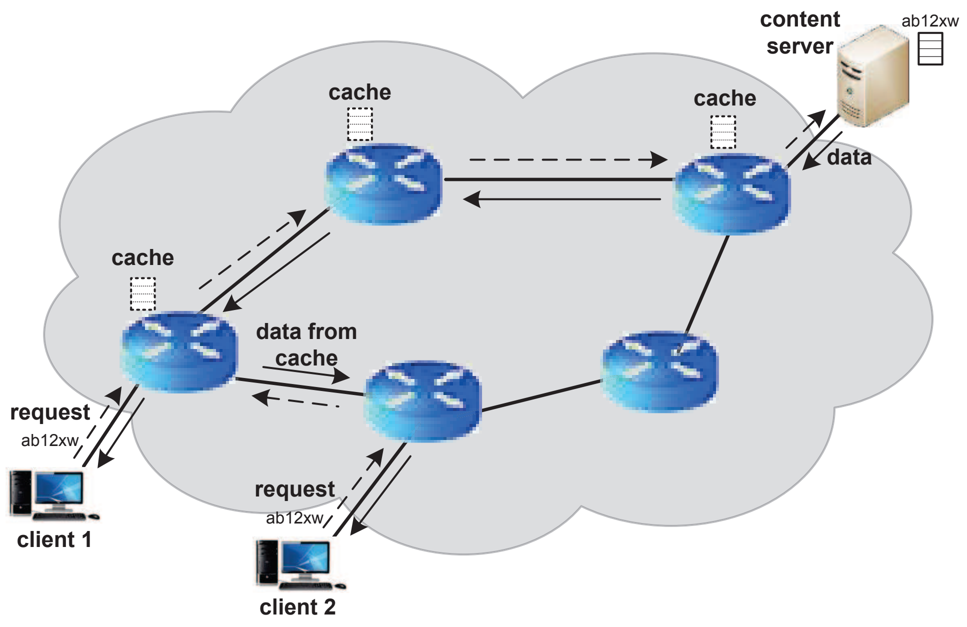


Figure 2.1: Content delivery example in ICN. Content is requested through name at the network layer, and is cached at ICN router when the data returns. Subsequent request is satisfied from a nearby cache.

2.1 Content-Centric Network Architecture

Content-centric networking (CCN) [40], followed by the named data network (NDN) [52] project, proposes an ICN architecture that uses hierarchical name to identify a content and forwards packet based on the content name, i.e., name-based routing.

2.1.1 CCN Name and Packet

CCN uses hierarchical name to identify a content. In addition, a large content might be segmented into a number of chunks, each with a unique name, e.g., /parc.com/videos/WidgetA.mpg/v3/s0, /parc.com/videos/WidgetA.mpg/v3/s1, etc.

CCN defines two types of packet: *Interest* and *Data*. Both Interest and data chunk specify the content name. A receiver asks for a piece of content chunk by issuing an Interest packet with its name, and the corresponding Data with the same name is returned in response to that Interest. The receiver thus controls the progression of content retrieval by adapting the Interest sending rate.

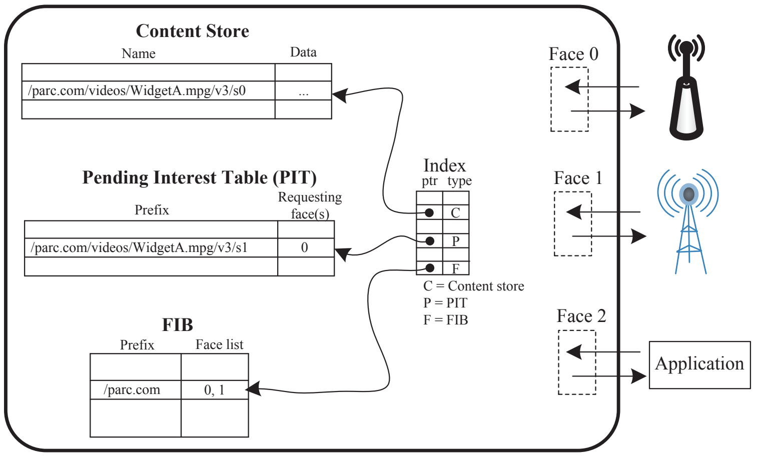


Figure 2.2: Packet processing model of CCN node.

2.1.2 CCN Node Model

As shown in Figure 2.2, a CCN node contains three main data structures for data processing: forwarding information base (FIB), content store (CS) and pending Interest table (PIT). FIB is used to forward Interest packets towards potential source(s) of matching Data. Its function is the same as an IP FIB, except it allows for a list of outgoing faces rather than a single one. The content store is the buffer memory of the node to store passing Data chunks for caching purpose. The PIT keeps track of forwarded Interest packets so that returned data chunks can follow the reverse path to its receiver.

When an Interest packet is received from interface *f*, the router performs the following operations: (1) checks CS and returns a copy through f if cache hits; (2) otherwise, conducts a PIT lookup to verify if an entry for the same content name already exists. If so, appends f to the entry and discards the Interest; (3) if not, creates a new PIT entry and forwards the Interest through the interface indicated by FIB.

When a Data chunk is received, the router forwards the chunk to all interfaces specified by the corresponding PIT entry and then removes that entry. The router may also choose to cache the chunk in CS, if appropriate.

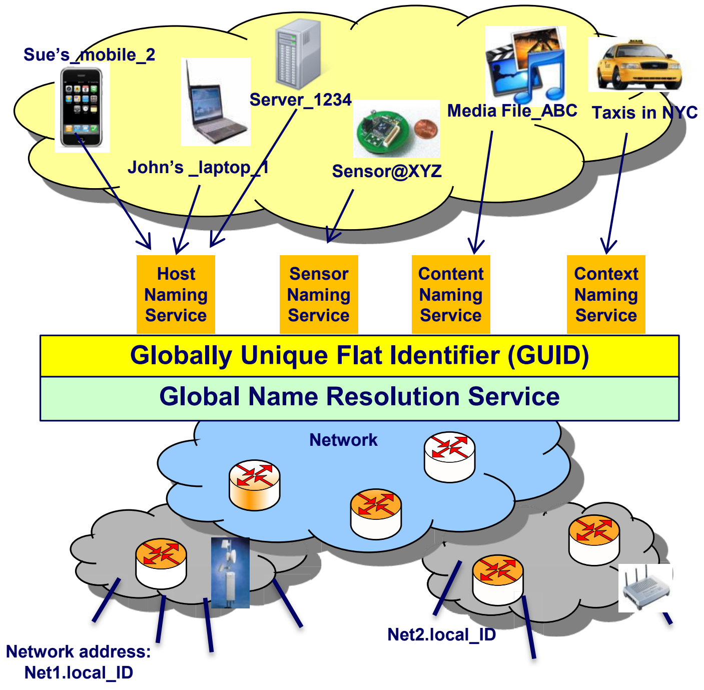


Figure 2.3: Name-locator separation and mapping in MobilityFirst.

2.2 MobilityFirst Architecture

The MobilityFirst (MF) project aims to design a clean slate Internet architecture to support large-scale, efficient and robust network services with mobility as the norm. One of the fundamental objectives of MF is to accommodate content-oriented applications and services in mobile scenarios. We talk about the detailed design of the architecture here.

2.2.1 Overall Network Architecture

The MobilityFirst architecture is built upon a name-based service layer that serves as the "narrow-waist" of the protocol stack. As shown in Figure 2.3, this layer uses flat globally unique identifiers (GUIDs) to name all network-attached objects including hosts, content, services and even abstract context.

A GUID can be assigned to a network object by one of multiple name certification services (NCSs), and is derived through a cryptographic hash of the public key that corresponds to that object. The GUID then works as the long-lasting network identifier for the object, and is decoupled from the object's locator or network address(NA). The dynamic mapping of GUIDs to NAs is maintained by a logically centralized, but physically distributed infrastructure, namely GNRS.

2.2.2 Content-Oriented Service

In MF, GNRS provides the basis for content operations. GNRS is structured in two levels: a Local Name Resolution Service (LNRS) at each domain/AS, and the global-level GNRS (gGNRS). LNRS contains the mappings of local objects within the AS which is composed by one or multiple central servers, while gGNRS is designed as an in-network 1-hop DHT with servers distributed all across the Internet. Each server in this distributed service advertises responsibility to a portion of an orthogonal name-space, which is used to designate (using consistent hashing) a host for mappings of a particular GUID. The basic operations supported by GNRS are *insert*, *update* and *lookup*, and they correspond to a fresh insert of ⟨GUID, NA⟩ mapping, an update to an existing mapping, and the lookup of an object's location, respectively. The structuring design of GNRS is aimed to support response latencies lower than 100ms for all these operations in order to enable real-time applications for mobile hosts. Such objective is validated in one manifestation of GNRS scheme [74].

To make a content available over the network, the publisher inserts its name-to-locator mapping to GNRS after assigning a GUID to the content. Specifically, a GNRS *insert message*: ⟨GUID, Addr⟩ is sent to an LNRS server. The LNRS server on receiving the insert message first initiates an update at the local service plane. It then applies a consistent hash function on the GUID to determine the gGNRS server that will host the mapping ⟨GUID, NA⟩. An insert message is then forwarded towards that server.

Resolution of content location is initiated by issuing a GNRS *lookup message*: ⟨GUID, options⟩ to LNRS. The LNRS server first checks with the local service plane whether the content requested is located within the local network. If so, the request need not be forwarded to the global plane and a response is returned from the LNRS

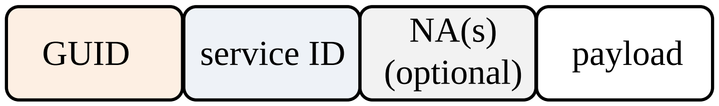


Figure 2.4: MobilityFirst packet structure with key fields.

server. Otherwise, a consistent hash determines the gGNRS server that hosts the mapping, and the lookup message is forwarded. Address received in the lookup response can then be used to address a content retrieval request.

When a content or its host moves, the host initiates an *update message*: ⟨GUID, Addr, options⟩ to LNRS. If the update is only to the local address, the update is handled at LNRS plane alone. If not, the LNRS forwards an update message to the gGNRS server determined by the hash to update the global mapping.

Figure 2.4 shows the packet structure in MobilityFirst. While GUID in packet header serves as the authoritative routing information, the NA if bound provides the fast forwarding path. In addition, in order to support a wide variety of applications and services, MobilityFirst enables the use the service flags (i.e., service ID) in the packet header so that network entities can tailor the behaviour of packet processing accordingly to suit the service's requirements.

2.2.3 Robust Data Delivery

MobilityFirst implements robust and efficient data delivery by combining hop-by-hop delivery of large data chunks (HOP transport) [46] with storage-aware routing (GSTAR) [53] within the network.

Content is segmented into several large blocks called chunks, with each chunk representing an autonomous data unit routed through the network. Data chunk is reliably transferred hop-by-hop, considerably reducing end-to-end retransmission. On each hop, after it transmits a chunk, it sends out a corresponding control message called CSYN to request acknowledgement from downstream node. The downstream node then acknowledges with a CACK message which contains a bitmap of reception status for every packet in that chunk. The transmission for this chunk finishes when all packets of the chunk are received by the downstream node, otherwise, the lost packets are retransmitted following the same procedure until there is no loss.

In addition, a storage-aware routing protocol is proposed that can seamlessly adapt to wireless link variations. The protocol exploits storage available at each router to overcome intermittent link quality variation and disconnections. In particular, we use short term ETT (most recent few estimates) and long term ETT (statistics over a longer period) to make a relative estimation of current link quality where ETT is derived from periodic probe messages sent to neighbors. If link/path quality is good, the chunks are forwarded along their way. If not, the chunk is temporarily held in a local storage until link quality improves.

Chapter 3

Transport Protocol for Content-Centric Network with Explicit Congestion Control

In this chapter, we analyze the characteristics of content delivery in content-centric networking (CCN) architecture, and its implications on transport control. We propose a novel transport control protocol [86, 87] that well meets the requirements of content-oriented service. We present our transport protocol, named CHoPCoP/pCHoPCoP, in detail here.

3.1 Introduction

During the last decade, content retrieval has dominated the Internet usage. To address the challenges posed for content retrieval, content-centric networking (CCN) [40, 52] has been proposed. Being a significant shift in the network design philosophy, CCN is centered on named content instead of host addresses. Routing towards a content is based on the content name instead of the host address, and data retrieval is initiated by issuing *Interest* at the content receiver. Compared to application-layer overlay solutions such as Content Distribution Networks (CDN) and Peer-to-peer systems (P2P), CCN holds the promise of providing a more efficient and cost-effective solution to content dissemination.

CCN's unique characteristics introduce new design challenges for the underlying transport protocol. First, CCN is naturally receiver-driven, since the content receiver needs to issue an Interest first in order to request a Data chunk. Second, hop-by-hop transfer is desired for CCN transport, because content files can be cached along the route to improve throughput. Moreover, since a specific content is often widely disseminated and cached in the network, a CCN flow may have multiple sources – i.e., one content chunk originates from source A, while the next chunk might originate from source B. Such multi-source/multi-path transfer in CCN makes congestion estimation based on a single RTT value fall short. These features of CCN are sufficiently distinct from a traditional end-to-end host-based model that a new transport approach is called for.

Recently transport protocol design for CCN has received attention in the research community and several proposals have been published [13–16, 58, 65, 79]. In order to deal with the challenge caused by the multiple-source, multiple-path transfer, a recent study proposes the use of multiple RTT estimators at the receivers to gauge network congestion of each path. Additionally, recent studies also suggest the routers adopt a hop-by-hop Interest shaping scheme to actively prevent network congestion.

In this chapter, we propose a new transport scheme, called the Chunk-switched Hop Pull Control Protocol (CHoPCoP), which has the following design elements:

- *Random early marking(REM).* REM uses explicit congestion signalling instead of RTT-based congestion prediction. Router detects congestion by monitoring the size of outgoing data queue. It then explicitly marks data packets to notify receivers when the network is congested.

- *Fair share Interest shaping(FISP).* CHoPCoP router decides whether to forward an Interest immediately or delay it temporarily based upon the available queue sizes and the flow demands. FISP is triggered to delay Interests when REM can't effectively prevent congestion, for instance in the absence of cooperation from receivers, thus actively protects the router from congestion. FISP also realizes fair bandwidth sharing among flows by fair scheduling of multiple Interest queues at an interface.

- *AIMD-based receiver Interest control(RIC).* The receiver adjusts its Interest window based on the AIMD (Additive Increase Multiplicative Decrease) mechanism. Here, receiver detects congestion mostly by marked packets from upstream routers.

We further extend CHoPCoP to a parallelized version (pCHoPCoP) for multi-homing scenarios in which the receiver has multiple active network interfaces. Here, the pCHoPCoP receiver consists of a pCHoPCoP engine and multiple CHoPCoP controllers, one for each network interface; and transport control functions are distributed among these components.

We have implemented CHoPCoP/pCHoPCoP using the Click Modular Router [42] and evaluated its performance on the ORBIT testbed [57]. We conduct experiments over various network and traffic settings and compare our protocol with several existing ones. Our evaluation shows that *i)* explicit congestion control provides congestion detection timely and correctly in a multi-source/multi-path setting, and significantly alleviates detection errors due to source/path change, thus improves network stability and efficiency; *ii)* our FISP scheme ensures fair sharing of network resources among different flows, it also provides protection against misbehaving receivers while still makes the most of network resources; *iii)* our receiver Interest control scheme guarantees full bandwidth utilization while reacts to REM signal to avoid saturating the network; *iv)* the pCHoPCoP receiver can utilize all its interfaces and realize bandwidth aggregation over each of them, thus maximize the network utilization for the multi-homed host.

The rest of the chapter is organized as follows. Section 3.2 describes the overall features of CCN and the related work on transport control in CCN. A detailed description of CHoPCoP and pCHoPCoP is given in Section 3.3 and Section 3.4. Our implementation is presented in Section 3.5. Section 3.6 presents ORBIT-based evaluation results, and concluding remarks are given in Section 3.7.

3.2 Related Work on Transport Control

There has been huge amount of work on transport protocols in the Internet. Here we focus on transport schemes proposed for CCN. We give a discussion of such work in three aspects.

RTT-based congestion control: Some designs, for instance ICP [13] and ICTP [65], rely on a single round trip time (RTT) estimator at the receiver to predict network status which are not suitable for CCN because CCN flows may have multiple sources and multiple paths. To adapt to CCN's multipath nature, authors in [15, 16] propose per-route transport control, in which a separate RTT estimation is maintained for each route at the receiver, similar to MPTCP [76]. Multiple RTT based scheme works well for multipath transfer, however, it adds lots of complexity to the receiver and heavily relies on the accuracy of timing. Our scheme, on the other hand, utilizes explicit congestion signalling from network to effectively notify the receiver about network condition. Compared with multiple RTT based scheme, our approach leads to a much simpler receiver design and is not restricted to limitations of timers which have long been criticized [30, 32, 61, 89].

hop-by-hop Interest shaping: Since in CCN one Interest packet retrieves at most one Data chunk, a router can control the rate of future incoming data chunks by shaping the rate of outgoing Interests. In [14, 58], authors propose quota-based Interest shaping scheme to actively control traffic volume. They assign a quota (in terms of the number of pending Interests) to each flow, and if the number of pending Interests for a flow exceeds the quota, that flow's Interests will be delayed or dropped. In [79], authors use per interface rate limit to avoid congestion at an interface and per prefix-interface rate limit to control Interest rate of each content prefix. The quota-based Interest shaping and the rate limiting Interest control require to assign an appropriate quota value for each flow or rate limit value for each content, which is challenging in practice, if not impossible. And they can be rather inefficient under dynamic traffic setting. Our fair share Interest shaping scheme also considers about fairness, however, resources are shared by all flows and Interest from a flow is delayed temporarily only when shared resources become limited and the corresponding flow unfairly consumes resources.

flow-aware traffic control: Authors in [58] propose a flow-aware network paradigm for CCN. They define content flow as packets bearing the same content name identified on the fly by parsing the packet headers. Moreover, routers impose per-flow fair bandwidth sharing by having multiple data queues at each interface for active flows and using deficit round robin (DRR) [69] for fair scheduling among these queues. Different

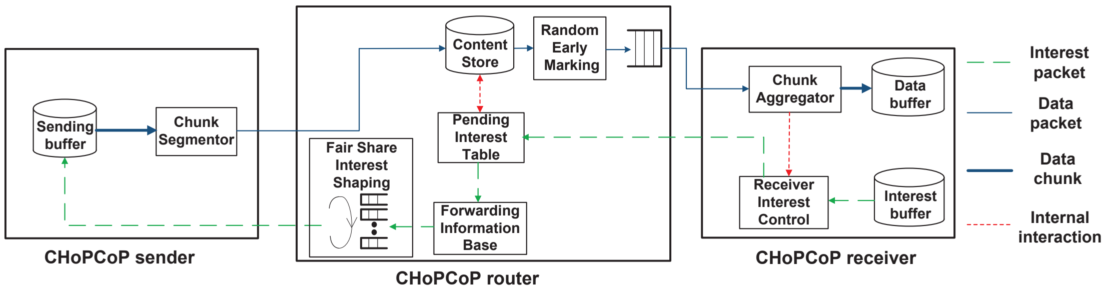


Figure 3.1: Functional modules of CHoPCoP content provider, router and receiver (we omit inbound queues in router for simplicity)

from such scheme, our FISP realizes per-flow fair sharing by having per-flow Interest queues at an interface and utilizing modified DRR to approximate max-min fairness.

Multi-homing support: Several transport layer protocols [37,38,47] are proposed to realize bandwidth aggregation for multi-homed hosts in the Internet. They propose an end-to-end transport approach in which either the sender [38,47] or the receiver [37] maintains multiple transport pipes/subconnections, one for each interface. We extend CHoPCoP to a parallelized version by borrowing some of their mechanisms.

3.3 CHoPCoP Design

In this section, we describe the design of the CHoPCoP protocol in detail. CHoPCoP consists of the following three functional modules: (1) explicit congestion signalling by random early marking; (2) fair share Interest shaping; and (3) AIMD based receiver Interest control. Figure 3.1 illustrates the functional modules of CHoPCoP content provider, router, and receiver. In our router model, the memory allocated for buffering packets at interfaces is separated from that used for caching (CS). Each interface has separate inbound buffers/queues and outbound buffers/queues for Interest and Data.

CCN data chunk is large in size [40] because of extra fields in the packet (e.g. signature). Large chunk can cause fragmentation, which may drastically harm throughput since the loss of a single fragment will cause retransmission of the whole chunk, like IPv4 fragmentation. Similar to two-level content segmentation in [65], we thus propose that a chunk is segmented to multiple small packets before the content provider sends it out and the receiver aggregates received packets into a complete chunk. If certain data packets in a chunk are lost, the receiver will issue a specific Interest packet containing the offsets of lost data packets within that data chunk, causing only those lost data packets to be retransmitted.

3.3.1 Explicit Congestion Signalling

Since content flow in CCN may have multiple sources and multiple paths, using a single RTT estimation cann't work well. Although using multiple RTT estimations [15] is proposed to address the issue, we take a different approach that each intermediate router estimates its congestion level and then notifies the receiver of the congestion event. Upon reception of such a notification, a properly functioning CHoPCoP receiver slows down the Interest issuance rate using the method presented in Subsection 3.3.3.

The technique we propose to achieve explicit congestion signalling is referred to as *Random Early Marking* (REM), similar to mechanism in RED [32] and ECN [30]. In REM, before a router forwards a data packet through interface f, it samples the occupancy of the corresponding outbound data queue, denoted by q. The router then smooths the sampling by calculating the moving average of queue occupancy $\bar{q}$ as

$$\bar{q} = (1-\mu)\bar{q} + \mu q, \tag{3.1}$$

where $0 < \mu < 1$ is a design parameter which sets the weight of the current sampling. In this way, we can avoid the adverse impact caused by temporary increases in the data queue. The router marks the packet with a probability $P = \frac{\bar{q}-q_{min}}{q_{max}-q_{min}} P_{max}$ if the value of $\bar{q}$ is between q_{min} and q_{max}, and with a probability $P = P_{max} + \frac{\bar{q}-q_{max}}{q_{max}}(1-P_{max})$ if $\bar{q}$ is between q_{max} and $2 * q_{max}$ (shown in Figure 3.2). If the queue occupancy is above the threshold $2 * q_{max}$, the router always marks the packet. Such gentle variation of packet marking probability is proved to make explicit congestion control much more robust to the setting of parameters [31].

REM predicts network congestion much more accurately than single RTT estimation in a multi-source environment. It actively notifies the receiver prior to congestion taking place, thus keeps the network stable. REM enjoys other additional benefits

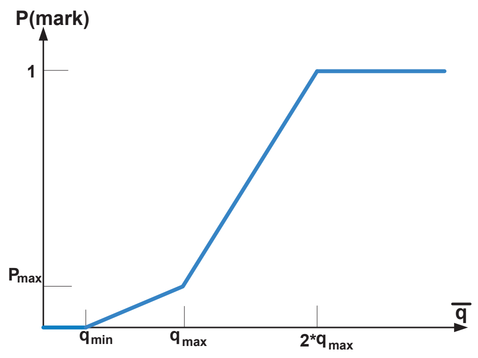


Figure 3.2: Mark probability function for REM

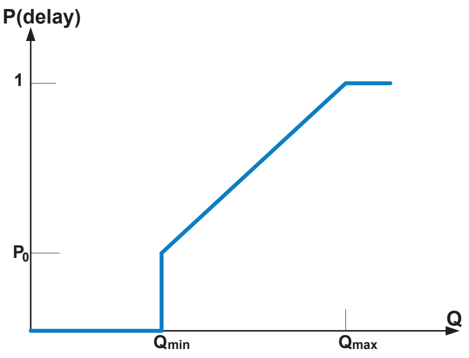


Figure 3.3: Delay probability function for FISP

same as RED/ECN, including the avoidance of global synchronization and bias against bursty traffic, and the guarantee of statistical fairness [32]. Compared to multiple RTT estimation proposal [15], REM vastly simplifies congestion detection at receiver and is free from limitations of timer [30, 32, 61, 89]. Compared with RED/ECN, in REM the congestion signal is delivered to the receiver and directly used for controlling rate instead of being reflected back to the sender, thus causing less delay.

3.3.2 Fair Share Interest Shaping (FISP)

A lightweight flow-aware network paradigm is well-adapted to CCN and can bring significant advantages [58]. This requires to realize per-flow fair bandwidth sharing at router and overload control even when the receiver is non-cooperative.

In CHoPCoP, we propose a fair share Interest shaping scheme, FISP in short, to address these issues. As seen in Figure 3.1, multiple Interest queues are allocated in an interface, one for each active content flow, identified by the content name. An active content flow corresponds to a FIB entry that has at least one PIT entry in active, thus flow information can be easily extracted from FIB and PIT. A modified DRR [69] is further used for fair scheduling among different queues: the deficit counter of a queue here is decreased by the size of the corresponding Data after servicing an Interest. To deduct such size value, we can use the segmentation information from the Interest (e.g. in the CCNx prototype [17]), or define a standard field as suggested in [14]. The analysis in [44] shows that the number of active flows that need scheduling remains in hundreds even though there may be tens of thousands of flows in progress, thus demonstrates that fair sharing is scalable.

Moreover, FISP is triggered to delay certain Interests when a router's data queue continues to grow even though REM has marked outgoing data packets, which is particularly helpful when the receiver is non-cooperative. FISP realizes this using an algorithm that consists of the following four phases.

In the first phase, FISP checks whether delaying Interests should take place. To do so, FISP periodically counts the total queue requirement at an interface, Q, as

$$Q = q_d + \gamma q_i, \tag{3.2}$$

where q_d quantifies the occupancy of outgoing data queue which is directly extracted from REM, q_i quantifies buffer resources needed by data chunks that will arrive at a response to outstanding Interests, and γ is a weight parameter. q_i is implemented along with PIT. It is increased when an Interest corresponding to the interface is sent upstream and decreased when data comes back. If the value of Q exceeds a preset threshold value, Q_{min}, the router starts Interest shaping for this interface.

In the second phase, FISP determines which flow's Interests should be delayed by looking at each flow's queue requirement. We use Q_j to denote flow j's queue requirement calculated similar to Equation 3.2. If Q_j exceeds its *fair share*, i.e., $Q_j \geq \frac{Q_{min}}{n}$ (n is the number of flows the interface currently has), then flow j's packets are delayed with a probability $P = \frac{Q-Q_{min}}{Q_{max}-Q_{min}} + P_0$ as pictorially shown in Figure 3.3. The Interests that are delayed are sent to a delay queue instead of being sent upstream and will re-enter outgoing Interest queues after a certain interval. The delayed Interests will not be counted towards the queue requirement for the interface.

In the third phase, we consider the overly-congested scenario. Once the queue requirement Q exceeds another threshold, Q_{max}, then any incoming Interest will be delayed.

In the fourth phase, if the router finds that the queue requirement, Q, falls below $Q_{release}$, then it will release all the delayed Interests to the outgoing Interest queues.

We note that the relationship between the three threshold values is $Q_{release} < Q_{min} < Q_{max}$.

Note that Q_{min} in FISP is larger than $2 * q_{max}$ in REM, and the delay probability starts from P_0 instead of 0, ensuring FISP is triggered to delay Interests after REM, when router queue continues to accumulate even after packet marking.

FISP actively protects the network from congestion even with non-cooperative receivers. Compared with quota-based Interest shaping [14, 58], FISP is more efficient in resource utilization, while it also ensures fairness among different flows. Moreover, the whole protocol is kept simple since the delaying algorithm in FISP is not triggered in normal condition.

3.3.3 AIMD Based Receiver Interest Control (RIC)

In CHoPCoP, each receiver maintains an Interest window that keeps track of pending Interests. The size of this window determines the Interest rate from the receiver, which in turn impacts the traffic volume in the network. We need to keep the receiver Interest window large enough to enjoy the available bandwidth, while not saturating network capacity. Here, we use the AIMD (Additive Increase Multiplicative Decrease) mechanism to manage receiver Interest control (RIC). Specifically, the receiver adjusts its Interest window proportional to congestion level implied in explicit congestion signalling.

RIC consists of two phases, namely, the slow start phase and the congestion avoidance phase. The slow start phase begins when the receiver sends out the first Interest for a given content. In this phase, the Interest window rapidly increases to utilize the network capacity by incrementing the window size by one every time it receives a complete data chunk. After the window size, W, reaches a threshold, or when the network is congested, the receiver starts the congestion avoidance phase. Here, the receiver window is either increased at a much slower rate, or decreases, depending upon whether the congestion is detected. Before congestion is detected, every time after the receiver receives W data chunks, we increase the window size by α where $\alpha < W$. After congestion is detected, however, the window size decreases to βW where $\beta < 1$. The values of α and β determine how aggressive the user is in tracking available bandwidth, as has been well studied in TCP [21].

In CHoPCoP, the receiver detects congestion either when it has a timeout or receives marked packets. We use different β values in these situations. In the former case, we simply use a fixed value, β_0, and in the latter, we calculate β as

$$\beta = \beta_2 - \frac{(\beta_2 - \beta_1)N_{marked}}{N}, \tag{3.3}$$

where N_{marked} and N denote the number of marked packets and the total number of packets in a chunk respectively.

Equation 3.3 shows that the receiver reacts to REM in proportion to the extent of congestion, not only its presence. Note that in normal condition the receiver detects congestion through receiving REM notification; timeout only takes place when REM fails.

Whenever a timeout occurs, the receiver needs to retransmit the Interest. RIC retrieves the offsets of lost data packets within the data chunk from the chunk aggregator and sends out the Interest with the offset information. The timeout value is calculated as:

$$R\bar{T}T = \sigma R\bar{T}T + (1 - \sigma)RTT, \tag{3.4}$$

$$TimeOut = \delta R\bar{T}T, \tag{3.5}$$

Where RTT denotes the current RTT sample value, while $R\bar{T}T$ denotes the moving average of RTT value. Note that we would set the timeout parameter relatively large because REM is adopted and timer is not critical any more.

3.4 pCHoPCoP: parallel CHoPCoP

CHoPCoP is designed for the data transmission scenario in which the receiver has only one active interface for fetching content. For multi-homed hosts, the receiver could have multiple active interfaces fetching content from several content providers, thus an extension for CHoPCoP is needed which is called pCHoPCoP. A multipoint-to-point pCHoPCoP connection is regarded as multiple CHoPCoP running paralleled to

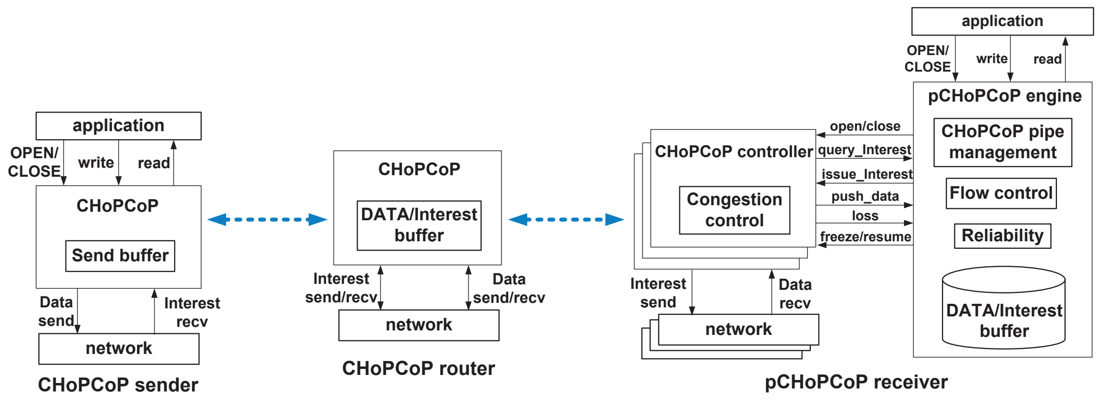


Figure 3.4: pCHoPCoP protocol architecture. The pCHoPCoP receiver consists of a pCHoPCoP engine and multiple CHoPCoP controllers.

download a content from several interfaces. We present its overall architecture and function elements here.

3.4.1 Design Framework

Figure 3.4 shows the overall architecture of pCHoPCoP. The sender and router work in the same way as CHoPCoP. However, for pCHoPCoP receiver, the transport layer consists of two functional units: the pCHoPCoP engine and the CHoPCoP controller. The pCHoPCoP engine controls the cooperation and progression of different CHoPCoP controllers. The application pushes Interest packets to the pCHoPCoP engine, and then pCHoPCoP engine distributes these Interest packets to different CHoPCoP controllers. A CHoPCoP controller receives data chunks from its corresponding network interface, and then sends them to the pCHoPCoP engine; finally the application pumps data from pCHoPCoP engine.

A pCHoPCoP connection consists of multiple CHoPCoP pipes[1]. Transport control functions related to per-pipe characteristics are put at each CHoPCoP controller while other functions pertaining to aggregate connection are managed at the pCHoPCoP engine. Here, reliability and buffer management pertain to the aggregate connection, and are handled by pCHoPCoP engine. In addition, flow control also lies at pCHoPCoP

[1]The data transmission path formed by a CHoPCoP controller is denoted as CHoPCoP pipe.

engine. On the other hand, congestion control, being per-pipe functionality, is still handled by individual CHoPCoP controllers. Therefore, while an individual CHoPCoP controller controls how much data it can request along its path, the pCHoPCoP engine controls what data to request through each CHoPCoP controller.

pCHoPCoP achieves bandwidth aggregation by effectively scheduling transmissions (Interests) to each CHoPCoP pipe. Briefly, CHoPCoP controller calls for Interests from pCHoPCoP engine when there is space in its congestion window and pCHoPCoP engine assigns Interests to the CHoPCoP controller based on the pending Interest space. The individual CHoPCoP controller is responsible for loss detection and reports any loss detected to pCHoPCoP engine such that the corresponding Interest will be reassigned to another CHoPCoP controller that has space in its window.

3.4.2 Protocol Details

The pCHoPCoP engine maintains two key data structures for performing effective Interest packet scheduling: the pending Interest space and the pipe state table.

pending Interest space: The data structure keeps the ranges of chunk IDs for data yet to be requested, which includes the chunk IDs of data that need to be re-requested, and chunk IDs greater than the highest chunk ID requested so far.

pipe state table: The data structure keeps track of which CHoPCoP pipes are active and which are not. As the available memory of the pCHoPCoP receiver is limited, the pCHoPCoP engine needs to consider whether it has enough space in the receiving buffer or not when a CHoPCoP controller calls for an Interest. The pCHoPCoP engine will thus freeze or reactivate a CHoPCoP pipe accordingly.

As seen in Figure 3.4, four functions act as the interface between the application and the pCHoPCoP engine.

OPEN()/CLOSE(): The application opens a pCHoPCoP socket by using the OPEN() call and terminates the pCHoPCoP socket by the CLOSE() call.

write()/read(): The application publishes its ranges of chunk IDs for data to be requested to the pCHoPCoP engine using the write() call and fetches data from the receiving buffer of the pCHoPCoP engine using the read() call.

In addition, eight functions act as the interface between the pCHoPCoP engine and CHoPCoP controllers.

open()/close(): The pCHoPCoP engine creates a CHoPCoP controller by using the open() call and then the CHoPCoP controller starts the communication through its corresponding interface. The pCHoPCoP engine removes a CHoPCoP pipe by sending a close() call to the corresponding CHoPCoP controller. The pipe state table will be updated for these two operations.

query_Interest()/issue_Interest()/push_data(): When a CHoPCoP controller has space in its congestion window, it queries the pCHoPCoP engine for Interest through the query_Interest() call. The pCHoPCoP engine then distributes a suitable Interest packet to the CHoPCoP controller using the issue_Interest() call. When a CHoPCoP controller gets a data chunk, it sends it to the pCHoPCoP engine by using the push_data() call.

freeze()/resume(): When the pCHoPCoP engine doesnt have enough buffer, it uses freeze() call to freeze some CHoPCoP pipes and updates the pipe state table. When enough buffer becomes available again, the pCHoPCoP engine uses resume() call to reactivate some sleeping pipes according to the pipe state table.

loss(): When a CHoPCoP controller detects a loss, it uses the loss() call to inform the pCHoPCoP engine. The pCHoPCoP engine then inserts the chunk ID in the pending Interest space, thus the Interest will be retransmitted later.

3.5 Implementation

We have implemented a complete network stack for our protocol as a user-level daemon, using the Click Modular Router. In our implementation, packets are directly fetched from physical interfaces to the user space or pushed to physical interfaces from the user space through PCAP sniffing. A link-state routing scheme is implemented as the name-based routing substrate where each node broadcasts its link state advertisement (LSA) with the content information it has.

Packet and header format: The following packet types are defined in our implementation: Interest, Data (chunk/packet), "hello" message and LSA. Each packet header contains the *pkt_type* field that identifies the packet's type. "hello" message and LSA are used for building the routing table, similar to OSPF discussed in [40]. Both Interest and data contain *content_name* and *chunk_ID* identifying the requested chunk within the specified content. A data chunk is segmented into multiple data packets, each containing *seq_num* that is the offset of the packet in the chunk. An Interest can specify *seq_num* asking for a specific data packet. A *service_ID* byte is included in the data packet header with the last bit reserved for congestion signalling.

Data chunk: In our current implementation, a data packet is 1KB in size and a chunk consists of 32 data packets, which complies with the multi-segment setting specified in [65].

Memory management: The queues for each interface can be dimensioned using the traditional bandwidth-delay product rule [5]. We thus set the outbound data queue size equal to the average RTT value multiplied by the link bandwidth. Here we assume 300ms as the average Internet RTT value. Since the inbound data queue is some transient buffer where data packets are processed rather fast by the router engine, its size is set to a small fixed value as 1MB. For an Interest queue, its size is set to 100KB accordingly.

REM (random early marking): Parameters of the packet marking probability function is set as follows: $q_{min} = 0.1C$, $q_{max} = 0.3C$, $P_{max} = 0.002$, where C denotes the queue capacity. For REM smoothing, we set $\mu = 0.05$ in Equation 3.1. The parameter setting is based on discussions in RED [32]. Note that the value of P_{max} is rather small because a data chunk is regarded as marked at the receiver as long as any of its segmented packets gets marked.

FISP (Fair-Share Interest shaping): We set $\gamma = 0.6$ in Equation 3.2. For the delay probability function, $Q_{release} = 0.2C$, $Q_{min} = 0.6C$, $Q_{max} = 0.9C$, $P_0 = 0.3$, where C denotes the buffer capacity for the interface.

RIC (receiver Interest control): The threshold of Interest window for slow start phase is set to 20. AIMD has the following parameters: $\alpha = 1, \beta_0 = \beta_1 = 0.5, \beta_2 = 0.8$. We set $\sigma = 0.7$ in Equation 3.4 and $\delta = 6$ in Equation 3.5.

3.6 Evaluation

In this section, we describe our evaluation effort of CHoPCoP/pCHoPCoP on the ORBIT testbed [57] and present detailed experimental results here. Our evaluation focuses on CHoPCoP's capability in stabilizing network condition and adapting to CCN's multipath nature, and pCHoPCoP's capability in achieving aggregated bandwidth on multi-homed hosts. For such purpose, we conduct detailed experiments on several simplified but representative network topologies and compare CHoPCoP with existing CCN transport protocols that utilize a single RTT estimation (i.e. ICP [13], HR-ICP [14] and ICTP [65]) and protocols that use quota-based hop-by-hop Interest shaping (i.e. HbH in HR-ICP). We don't compare CHoPCoP with multiple RTT estimation proposals here since our scheme can be an alternative to those proposals in controlling multipath transfer.

3.6.1 Experimental Setup

The ORBIT testbed provides a realistic network environment to support protocol evaluation in both wired and wireless settings. Each ORBIT node in our experiments has a 2.93GHz Intel i7 quadcore processor, 4GB memory, and runs Linux 2.6. Also, each node is equipped with two wired interfaces, eth0 and eth1. Both interfaces on all the ORBIT nodes are connected, each forming a gigabit LAN. We use the link layer packet filtering techniques discussed in [9] to create logical topologies. In addition, we use Click Modular Router's link emulation elements to configure wired link parameters such as link bandwidth and delay.

3.6.2 A Single-Source, Single-Destination Benchmark Scenario

We first conduct several benchmark experiments using a three node baseline topology, shown in Figure 3.5. Here, the source node (A) is connected to the router (B) via a long

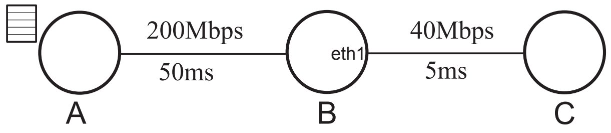


Figure 3.5: A three node baseline topology, consisting of a source (A), a router (B) and a receiver (C).

Internet connection (with a 200Mbps bandwidth and a 50ms latency), and the router is connected to the receiver (C) via a local Internet access (with a 40Mbps bandwidth and a 5ms latency). We set the outbound data queue size of the router's eth1 interface to be $40Mbps \times 300ms = 1500KB$ according to the bandwidth-delay product rule [5]. We consider a 320MB content file that consists of 10,000 chunks in total, each chunk 32KB in size. There is only one flow in this setting.

The Effectiveness of REM and RIC

First, we compare CHoPCoP with existing protocols: ICTP, ICP, and HR-ICP. Here, the receiver is cooperative and slows down Interest issuing when marked packets are observed, so the main components that are effective in CHoPCoP are REM and RIC. We will show that CHoPCoP stabilizes the network condition and achieves higher throughput.

Figures 3.6(a)-(d) summarizes the results, showing how the receiver window size, the transient receiving data rate, the number of timeout instances, and the router queue size, change over time. The results show that CHoPCoP significantly outperforms the others. The receiver side Interest window is much smoother, with an average size of 23.27 and a standard variance of 2.57 (Figure 3.6(a)); the receiving data rate is much higher (also smoother), with 39.91Mbps at the steady state (Figure 3.6(b)); and no timeout is observed at the receiver (Figure 3.6(c)). Specifically, the throughput[2] of CHoPCoP is 85.75% and 13.69% higher than that of ICP/HR-ICP and ICTP respectively. This is because it effectively smooths the outgoing data queue at the router (with

[2]We use the term throughput for the average data delivery rate, while the term receiving rate for instant data delivery rate.

an average of 235.8KB and standard variance of 116.5), as shown in Figure 3.6(d).

The inferior performance of ICTP/ICP/HR-ICP can be explained below. In ICTP, after the receiver window reaches a certain value (∼22 in our experiment), the network capacity is fully utilized, thus the receiving Interest rate remains the same even if the window keeps increasing. However, the increasing window results in a large queue at the router, which may easily cause congestion. On the other hand, in ICP/HR-ICP, a relatively small timeout setting causes a large number of false timeouts and thus poor throughput.

The Effectiveness of Fair Share Interest Shaping (FISP)

Next, we consider a non-cooperative receiver who issues Interests at a constant Interest rate (CIR), even when the router has signalled congestion through packet marking. In this case, FISP is triggered to actively delay Interest in order to mitigate congestion since the receiver doesn't respond to REM signalling.

Here, the router's outgoing data queue is 1500KB in size, and we have $Q_{max} = 0.9C = 1350KB$. We consider CIR of 140, 160, and 200 Interests per second in each run, requesting a 320MB file. We show the router queue size with and without FISP in Figure 3.7, and the delivery ratio and throughput in Table 3.1.

The results show that with FISP, the router's outgoing data queue can be kept at $\sim 1050KB$ when $CIR > \frac{40Mbps}{32KB} \approx 150$ Interests per second. Without FISP, router queue overflows and the router keeps congested. On the other hand, FISP's Interest shaping incurs the accumulation of Interest packets at the router which might exceed the capacity of the internal delay queue and cause Interest dropping. We set the capacity of the internal delay queue to be 5000 packets, and additional experiments show that FISP can handle a maximum CIR of around 300 without Interest dropping.

The Sensitivity of CHoPCoP Parameters

Here, we evaluate how CHoPCoP's performance is impacted by different parameter settings. In particular, we investigate the sensitivity of CHoPCoP parameters including the chunk size, REM smoothing parameter μ in Equation 3.1, REM's marking probability function in Figure 3.2, FISP's weight parameter in Equation 3.2, and FISP's delay probability function in Figure 3.3.

Table 3.2 shows that having a chunk size of 32 packets gives the best performance. If the chunk size is too large, the control granularity is not fine enough to have timely response to network dynamics; If the chunk size is too small, the control overhead becomes more pronounced.

Table 3.3 shows that REM is insensitive to the value of μ, making the calculation algorithm of router's average queue occupancy easier to configure in an actual network.

Table 3.4 and Table 3.5 show the impact of using different REM marking probability function parameters. When (q_{min}, q_{max}) is too small, packets will be marked unnecessarily; thus router queue size keeps small, and the receiving rate at the receiver doesn't reach optimal. The system throughput is optimized when parameter (q_{min}, q_{max}) increases to the value (0.05C, 0.2C). After that point, the receiving rate keeps almost the same although the receiver's Interest window size and the router's queue size increases along with the parameter. In Table 3.5, the average receiving rate is insensitive to the parameter P_{max}. These two tables show that the setting of REM mark probability function is relatively easy to configure since the throughput is not sensitive to it.

We next look at the impact of different parameters in FISP. Here the receiver issues Interests at a constant Interest rate of 160 Interests per second without responding

CIR	Data Delivery Ratio		Throughput	
	With FISP	Without FISP	With FISP	Without FISP
140	100%	100%	36.37 Mbps	36.36 Mbps
160	100%	5.86%	37.44 Mbps	2.4 Mbps
200	100%	1.91%	37.62 Mbps	977.9 Kbps

Table 3.1: FISP leads to better delivery ratio and throughput. Without FISP, throughput decreases significantly when queue overflows

Chunk Size (pkts per chunk)	Average Receiving Rate (Mbps)	Average Interest Window Size	Average Router Queue Size (KB)
2	10.08	106.61	0.71
8	34.40	91.60	84.84
32	36.93	21.92	197.69
128	33.77	6.23	384.56
512	24.30	1.76	420.26

Table 3.2: Results under different chunk size. Each packet is 1KB in size.

Smoothing Parameter μ	Average Receiving Rate (Mbps)	Average Interest Window Size	Average Router Queue Size (KB)
0.01	37.02	23.04	228.81
0.05	36.93	21.92	197.69
0.25	36.94	21.87	193.59

Table 3.3: Results under different REM smoothing parameter. Chunk size is 32KB.

to REM signal. As seen in Subsection **??**, the router queue would overflow without Interest control in such Interest rate.

The weight parameter γ in Equation 3.2 determines FISP'S sensitivity in responding to future chunk arrival in the near future, thus highly affects whether FISP can effectively control the aggressive traffic. As seen in Table 3.6, when γ is too small, FISP is not sensitive to the size of pending Interest, which causes router queue overflow and packet loss, thus a low throughput; when γ is too large, FISP keeps the router's average queue size very small, which might unnecessarily delay Interests and be in conflict with REM's setting. There, an appropriate value of γ is very important here, but FISP can work well in a relatively large range of γ value as shown in the table.

Table 3.7 and Table 3.8 illustrate the impact of using different FISP delay probability function parameters. These two table show that FISP can keep the router from congestion and realize high throughput at different (Q_{min}, Q_{max}) and P_0, which indicates the insensitivity of throughput to FISP delay probability function parameters.

(q_{min}, q_{max})	Average Receiving Rate (Mbps)	Average Interest Window Size	Average Router Queue Size (KB)
(0.01C, 0.05C)	29.47	13.49	30.69
(0.05C, 0.2C)	36.43	19.51	122.79
(0.1C, 0.3C)	36.93	21.92	197.69
(0.2C, 0.4C)	37.09	25.12	296.24
(0.3C, 0.5C)	37.10	29.21	420.56
(0.8C, 0.9C)	37.10	76.57	959.93

Table 3.4: Results under different parameter (q_{min}, q_{max}) in the REM mark probability function. Chunk size is 32KB, $P_{max} = 0.002$. Note that q_{max} should be smaller than router buffer capacity C.

P_{max}	Average Receiving Rate (Mbps)	Average Interest Window Size	Average Router Queue Size (KB)
0.001	36.97	22.96	225.63
0.002	36.93	21.92	197.69
0.02	36.29	19.54	130.08
0.1	35.95	18.39	95.04
0.2	35.78	18.03	85.58

Table 3.5: Results under different parameter P_{max} in the REM mark probability function. Chunk size is 32KB, $q_{min} = 0.1C$, and $q_{max} = 0.3C$.

3.6.3 A Multi-Source, Single-Flow Scenario

In CCN, a content request might be served by multiple sources, resulting in multiple paths. Single RTT based protocols, however, implicitly assume there is only a single path for a content flow, thus can't accurately estimate the network condition. In this section, we take a close look at the issues caused by having multiple sources for the same flow, as well as the effectiveness of CHoPCoP in handling these issues.

Weight Parameter γ	Average Receiving Rate (Mbps)	Data Delivery Ratio	Average Router Queue Size (KB)
0.4	27.86	99.7%	1221.7
0.5	37.44	100%	1108.6
0.6	37.44	100%	1053.8
0.8	37.44	100%	959.17
1.5	37.44	100%	635.3

Table 3.6: Results under different weight parameter γ in the FISP's queue requirement equation.

(Q_{min}, Q_{max})	Average Receiving Rate (Mbps)	Data Delivery Ratio	Average Router Queue Size (KB)
(0.4C, 0.7C)	37.44	100%	764.78
(0.6C, 0.9C)	37.44	100%	1053.8
(0.8C, 1C)	37.44	100%	1211.0

Table 3.7: Results under different parameter (Q_{min}, Q_{max}) in the FISP delay probability function. Chunk size is 32KB, $P_0 = 0.3$, and $Q_{release} = 0.2C$.

The Impact of Timeout Parameters at the Receiver

In this set of experiments, we show that a large timeout parameter is essential to avoid false timeouts at the receiver because of the inherent large RTT variation in a multi-source environment. To demonstrate the point, we consider a topology that consists of two sources. The detailed topology and the link parameters are shown in Figure 3.8. Here, we use a relatively small content file with only 100 chunks to eliminate the router queuing delay from the RTT measurement. These 100 chunks are randomly placed at the two sources, and thus both sources serve the same flow.

We run CHoPCoP with two timeout values using different δ values in Equation 3.5: $\delta = 2$ for a small timeout value while $\delta = 6$ for a large value. Table 3.9 shows the number of timeouts and throughput observed at the receiver. The results clearly show that small timeout value leads to many more timeouts. In fact, timeout takes place almost every time when the source switches from B to A. On the other hand, we observe no timeout if we choose a large timeout value, resulting in 55% higher throughput.

P_0	Average Receiving Rate (Mbps)	Data Delivery Ratio	Average Router Queue Size (KB)
0.1	37.44	100%	1062.2
0.3	37.44	100%	1053.8
0.5	37.44	100%	1048.9

Table 3.8: Results under different parameter P_0 in the FISP delay probability function. Chunk size is 32KB, $Q_{release} = 0.2C$, $Q_{min} = 0.6C$, and $Q_{max} = 0.9C$.

The Effectiveness of REM and RIC

Next, we show that REM is effective in detecting networking congestion timely and correctly when multiple sources exist, and thus keeping the network stable and efficient. We use the same topology as shown in Figure 3.8, with $\delta = 6$. Here, we consider a content file that consists of 10000 chunks, which are randomly placed at the two sources.

We compare CHoPCoP with existing protocols: ICTP, ICP, and HR-ICP, and show results in Figure 3.9. The results show that CHoPCoP is very effective in fully utilizing network capacity while keeping the network stable by explicitly signalling the receiver about potential congestion in advance. As a result, CHoPCoP improves throughput by 107.6% compared to ICTP, and by a factor of 2 compared to ICP/HR-ICP.

Poor performance of ICP and HR-ICP shows single RTT estimator cann't predict network congestion in multi-source environment. We also find that for ICTP, out-of-sequence packet arrival takes place frequently which triggers fast retransmission and fast recovery and in turn lowers the throughput. This shows that using packet sequence to detect packet loss similar to "triple duplicate ACK" is not reliable either.

δ	Timeout #	Throughput (Mbps)
2	37	25.1
6	0	38.79

Table 3.9: In CHoPCoP, a large timeout value (with large δ values) can eliminate false timeouts.

3.6.4 A Multi-flow Scenario

We further investigate how CHoPCoP behaves when multiple flows exist. We conduct the experiments on the topology in Figure 3.10, which consists of two receivers (D and E). These two receives request different content files hosted by A, resulting in two flows in the topology.

Fairness between the Flows

First, we show CHoPCoP adapts to the number of flows in the network – when we have one flow, it allows the flow to efficiently utilize the network resources, while when we have multiple flows, it provides fair sharing among them. For this purpose, in the topology in Figure 3.10, we have receiver D and E request two different content files (320MB each) from the source A. D starts its request at time 0 while E starts at time 20s.

The results in Figure 3.11 confirm our point. Before E starts, there is only one flow, initiated by D, and as a result, this flow owns the resources exclusively, with average window size of 42.24 and throughput of 37.89Mbps. As soon as E starts, the two flows share the resources in a fair manner, with average window size of 23.83 and 24.04, and throughput of 19.54Mbps and 19.44Mbps for D and E, respectively. Finally after D ends, the flow initiated by E owns the resources exclusively.

Comparison of FISP with Quota-based Interest Shaping

Next, we compare the performance of FISP against quota-based Interest shaping scheme, HbH, presented in HR-ICP [14], using the topology in Figure 3.10. In this set of experiments, receiver D sends out Interests at a constant rate of 20 Interests per second, while receiver E's Interest rate varies from 40 to 180, with a 20 Interests per second increase in each run.

The results in Figure 3.12 show that our scheme, FISP, provides better resource utilization, better handles traffic fluctuation, and thus gains higher network throughput. Specifically, after the network reaches saturation, FISP achieves the throughput as high as 37.40Mbps for FISP while HbH only has 27.26Mbps, with a 37.2% of improvement.

3.6.5 A Multi-Source, Multi-Path Scenario

We further look at a larger-scale network topology that has 7 nodes and 2 flows to validate CHoPCoP's capabilities. The topology and detailed link parameters are shown in Figure 3.13.

Throughput Comparison

We first compare the throughput of CHoPCoP with existing protocols: ICTP, ICP and HR-ICP. Here, receiver F and G simultaneously request two different content files C1 and C2 (320M each) respectively. Content C1 is located at source A, while content C2 is randomly distributed at the two sources.

Table 3.10 summarizes the throughput of two receivers under different schemes during the period when both flows are active. CHoPCoP achieves a performance gain of 103% over ICTP, and a factor of 6 over ICP/HR-ICP.

Transport Protocol	Throughput (F) (Mbps)	Throughput (G) (Mbps)	Total (Mbps)
CHoPCoP	36.86	59.44	96.30
ICTP	34.12	13.71	47.83
ICP	7.85	5.90	13.75
HR-ICP	7.68	6.05	13.73

Table 3.10: Throughput comparison among different transport protocols.

A Closer Look at CHoPCoP

Next, we take a closer look at how CHoPCoP behaves in such a larger-scale network with multiple flows. In this experiment, receiver F requests a 240MB content file C_1 from time 0, while G requests a 320MB content file C_2 from time 20s. Again, C_1 is at source A while C_2 is randomly distributed at the two sources. Here, we have two content flows, C_1 and C_2, the latter with two sources.

Figure 3.14 shows how the receiving rate for both flows changes with time. From time 0 to time 20s, flow C_1 is the only flow in the network, achieving a throughput of 36.86Mbps, close to the bandwidth of its bottleneck link EF(40Mbps). After flow C_2 starts at 20s, the two flows share the network resources. In this case, their total throughput is limited by the bandwidth of link CD (100Mbps). As a result, flow C_1's throughput remains around 36.86Mbps (because of its bottleneck link), while C_2's throughput is 59.44Mbps. After C_1 ends at 50s, C_2 owns the resources exclusively, achieving a throughput of 93.65Mbps, which is close to the bottleneck link bandwidth of 100Mbps.

This detailed breakdown shows that CHoPCoP achieves efficient resource utilization when flows with different bottleneck links dynamically join and leave the network.

3.6.6 Multi-homing Scenarios

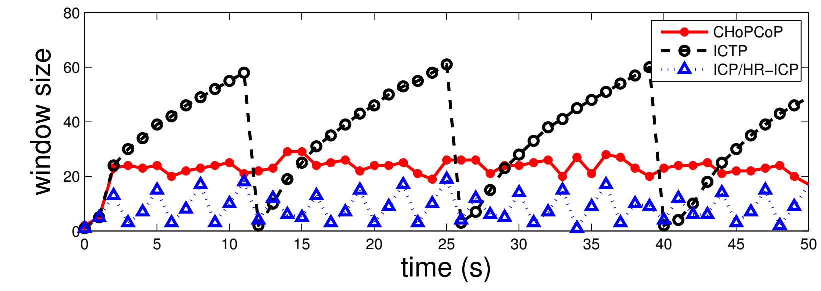
Finally, we evaluate pCHoPCoP's functionality of bandwidth aggregation using three multi-homing network topologies in which the receiver has two interfaces, as seen in Figure 3.15. In these scenarios, a 320MB content is located at both node A and B, and the receiver node E has two interfaces, *int1* and *int2*. Node E requests the content from both of its interfaces, resulting in two CHoPCoP pipes, *CHoPCoP1* and *CHoPCoP2*. Note that *int2* in topology (c) is a Wi-Fi interface.

We plot the throughput and the window size of the two CHoPCoP pipes and aggregated pCHoPCoP, as seen in Figure 3.16 and Figure 3.17. For multi-homing topology (a), since CHoPCoP1 and CHoPCoP2 are symmetric in terms of bandwidth and delay, the throughput and receiver window dynamics are almost the same for these two pipes; for topology (b), since the bottleneck for CHoPCoP2 is link DE with 80Mbps bandwidth, CHoPCoP2 has a higher throughput and larger receiver window than CHoPCoP1; for topology (c), because of the Wi-Fi link at CHoPCoP2, it has a lower throughput and smaller receiver window than CHoPCoP1. In summary, the results show that while each CHoPCoP controller can fully utilize network capacity over its pipe and keep the network from congested by maintaining a relatively stable receiver window size, the pCHoPCoP engine aggregates the utility of all its corresponding CHoPCoP controllers, thus resulting in much higher overall throughput at the receiver and less time for retrieving the content.

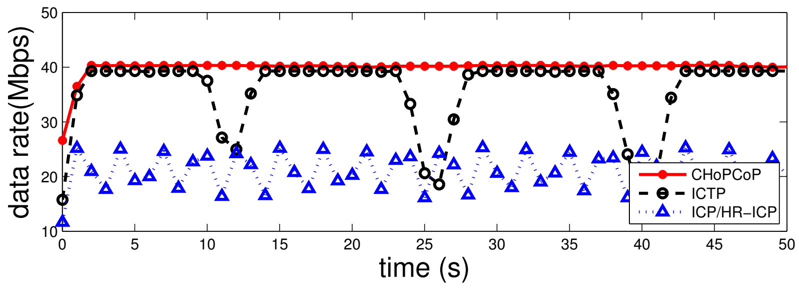
3.7 Conclusion

In this chapter, we present the design, implementation and evaluation of CHoPCoP, a CCN transport protocol. In addition to being receiver driven and hop-by-hop transport, CHoPCoP utilizes explicit congestion signalling to tackle with CCN's multipath nature. We also propose fair share Interest shaping scheme to provide bandwidth sharing among different flows. Moreover, our Interest shaping scheme will actively delay Interests when explicit congestion control can't effectively control network congestion.

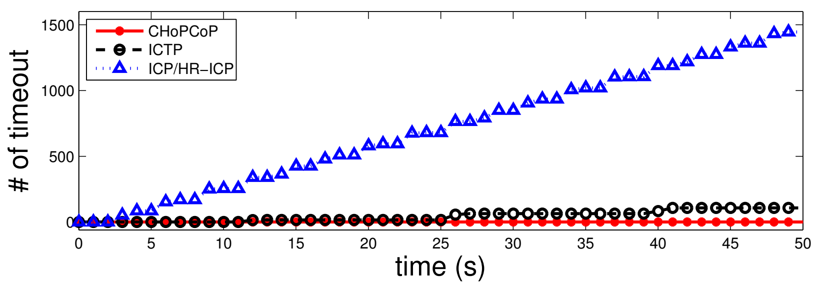
We have implemented the complete protocol stack using the Click Modular Router, and evaluated its performance on the ORBIT testbed. Our experimental results show that explicit congestion signalling, when coupled with our AIMD-based receiver Interest control, can successfully stabilize the router queues and improve the throughput in a multi-source/multi-path environment. When there are multiple flows in the network, our fair share Interest shaping scheme ensures fairness and provides more efficient resource utilization than earlier quota-based Interest shaping algorithms. Using a topology with multiple sources and multiple flows, we show that CHoPCoP can improve the total network throughput by at least 103% over existing solutions that use single RTT estimation. Finally, using three multi-homing topologies, we show that pCHoPCoP can effectively realize bandwidth aggregation over all available network interfaces/pipes for the receiver.



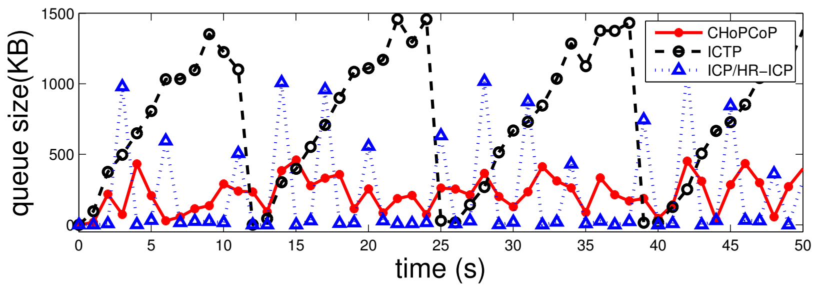
(a) receiver window size



(b) receiving data rate

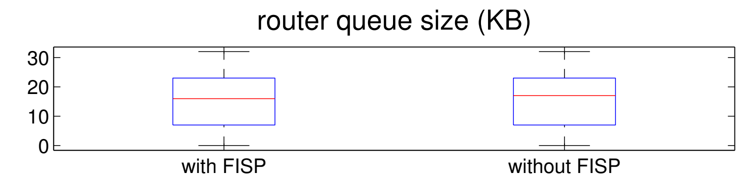


(c) cumulative timeout number

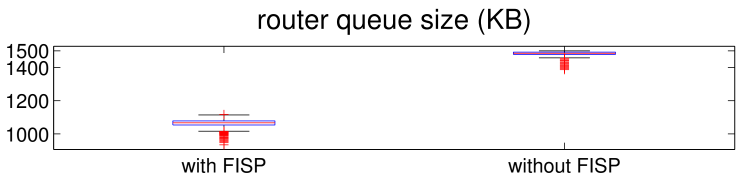


(d) router data queue size

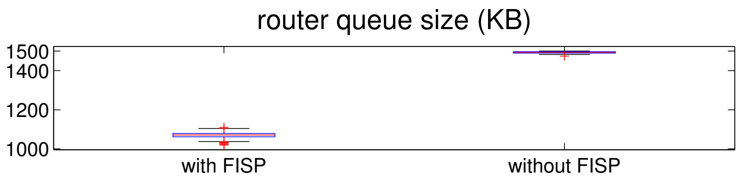
Figure 3.6: The performance results for a 3-node baseline topology. ICP and HR-ICP behave very similarly, we thus use the ICP curve to represent both schemes.



(a) Interest rate: 140 per second



(b) Interest rate: 160 per second



(c) Interest rate: 200 per second

Figure 3.7: Boxplot of router queue size with and without FISP under different CIR. At a low CIR, both schemes perform similarly. At a high CIR, FISP nicely controls the router queue size, and thus avoids queue overflow. Without FISP, the router queue overflows when the CIR is above 160 Interests per second.

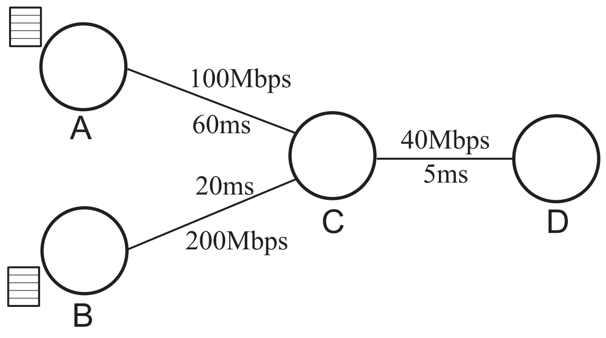
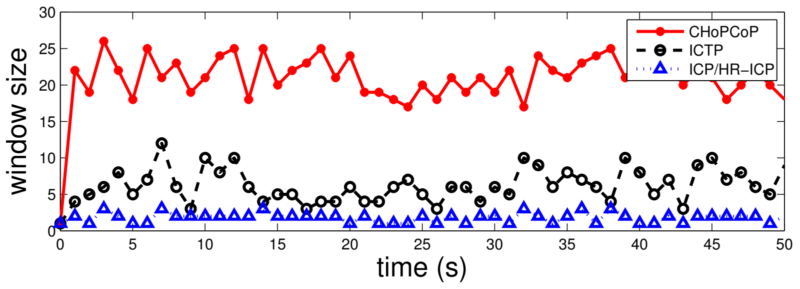
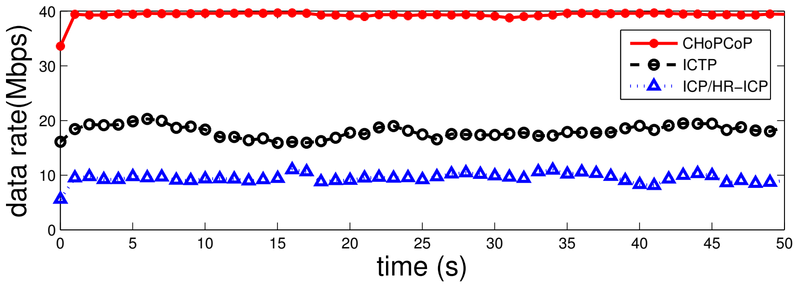


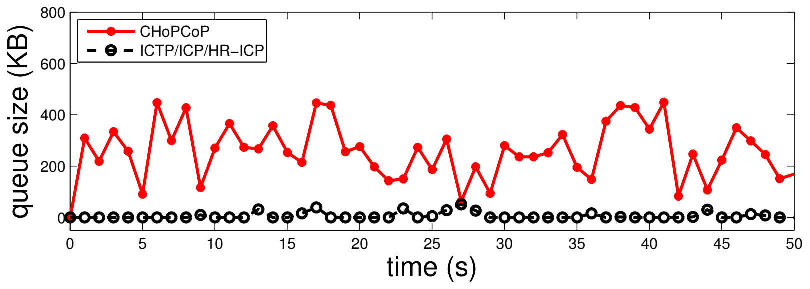
Figure 3.8: A topology that consists of two sources (A and B) that serve the same request from D.



(a) receiver window size



(b) receiving data rate



(c) router queue size (ICTP overlaps with ICP here)

Figure 3.9: Comparison of CHoPCoP, ICTP, ICP, and HR-ICP when two sources serve the same content flow. We use the ICP curve to represent both ICP and HR-ICP because of their similar performance.

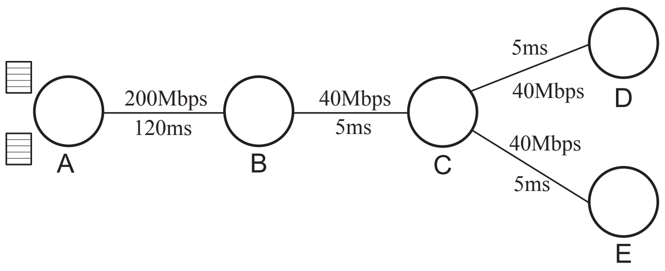
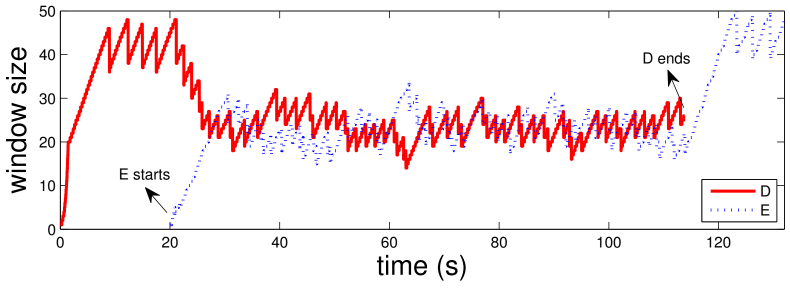
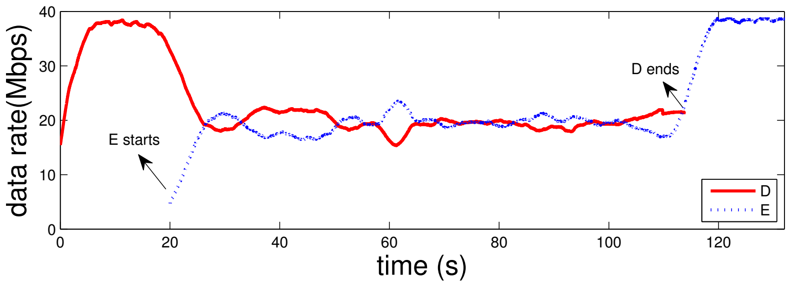


Figure 3.10: Multiple flow topology where D and E request two different files from the same source A.



(a) receiver window size



(b) receiving data rate

Figure 3.11: CHoPCoP can efficiently utilize the network resource when only one flow exists. When we have multiple flows, it ensures fairness among them.

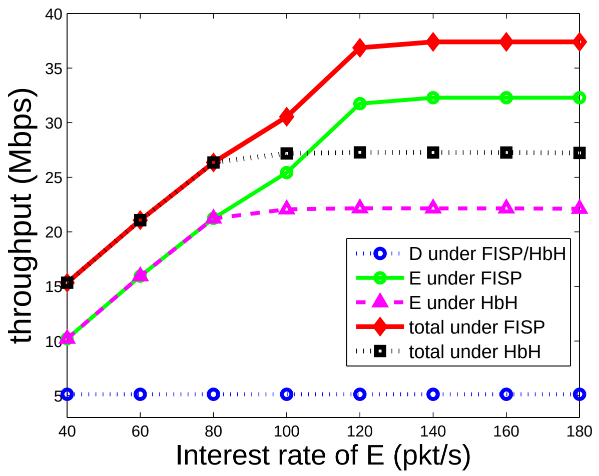


Figure 3.12: Throughput comparison of FISP with HbH. Receiver D has the same throughput for both schemes.

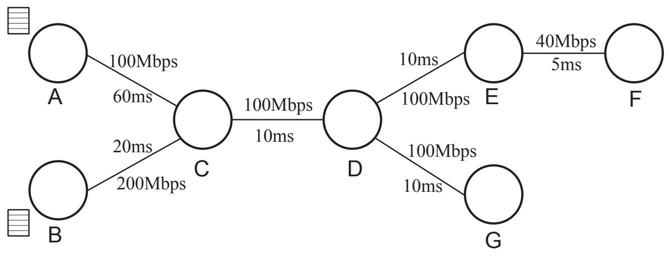


Figure 3.13: A larger-scale topology with two sources (A and B) and two receivers (F and G). Link EF is the bottleneck link between A/B and F, while link CD is the bottleneck between A/B and G.

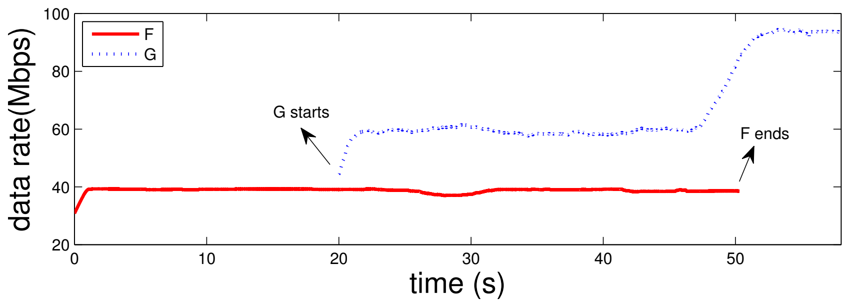


Figure 3.14: The receiving rate for the two receivers. F initiates content flow C_1 (with source A) while G initiates content flow C_2 with two sources A and B.

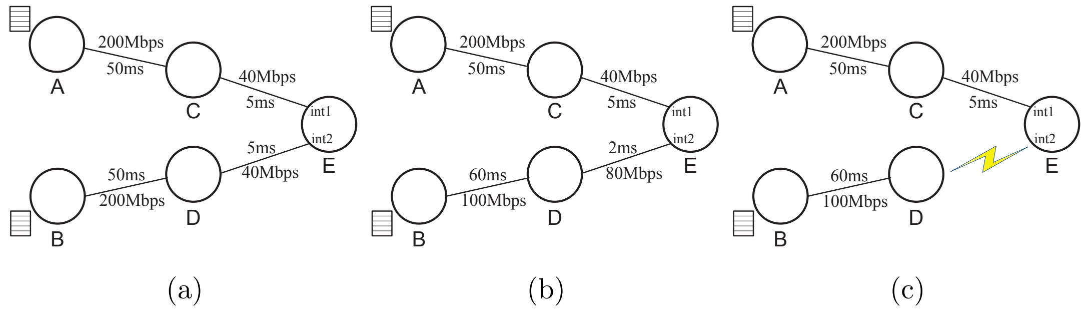


Figure 3.15: Three multi-homing topologies. Each consists of two sources (A and B), and the receiver E requests content from them using both of its interfaces.

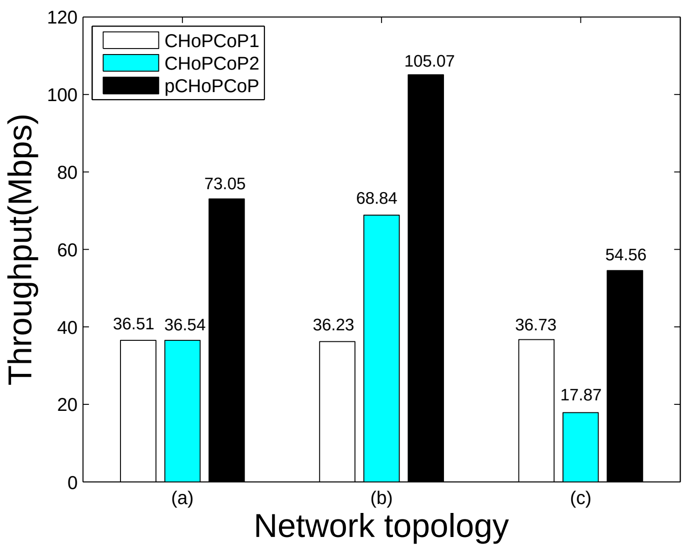
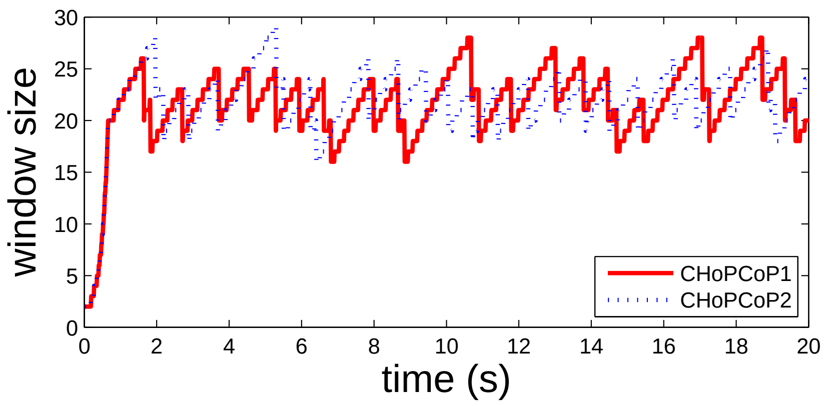
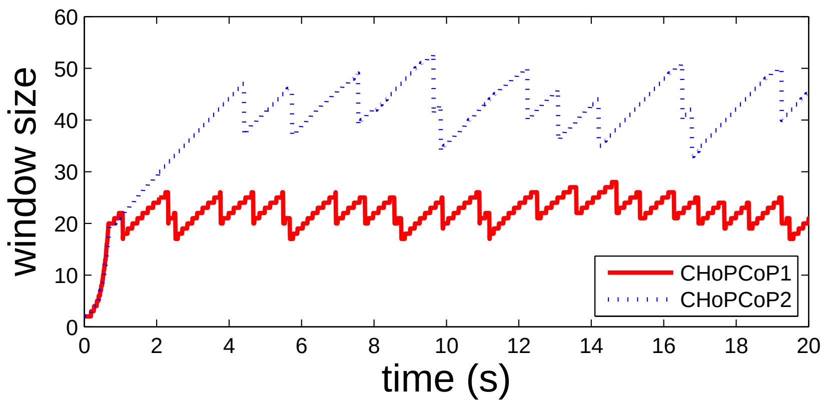


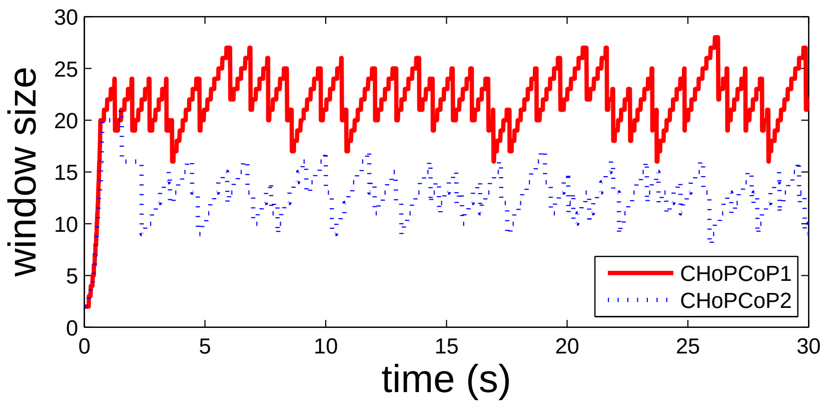
Figure 3.16: The throughput of the two CHoPCoP pipes and the aggregated pCHoPCoP at the three multi-homing topologies.



(a)



(b)



(c)

Figure 3.17: The receiver window size at each of the two CHoPCoP pipes for the multi-homing topologies in Figure 3.15.

Chapter 4

Caching and Prefetching Content at the Edge in the MobilityFirst Future Internet Architecture

This chapter presents EdgeBuffer framework [84], in which we propose to have both caching and prefetching functionality at the wireless edge to improve overall cache hit ratio. The framework is discussed in the context of MobilityFirst architecture, but could be easily adapted to other architectures including the TCP/IP stack.

4.1 Introduction

We are currently witnessing tremendous growth in the access of content using mobile devices. In fact, Cisco Visual Networking Index [22] reports that global mobile data traffic reached 2.5 exabytes per month at the end of 2014, and global mobile devices and connections in 2014 grew to 7.4 billion. The rapid rise of mobile content access has led to a dense deployment of wireless networks especially in urban environments, including WiFi access points (APs) and cellular base stations (macro cells and small cells). Users will now traverse multiple access points (AP) and base stations, encountering frequent network connection transitions within a session (e.g., watching a YouTube video), which in turn makes mobile content delivery while maintaining the user's seamless quality of experience, a daunting task.

The current TCP/IP Internet architecture is poorly suited for content delivery to mobile devices in this manner. A common criticism is the so-called identity-location conflation problem [66], where an IP address in the Internet is used to both identify an interface as well as its network location. As a result, connections break when an endpoint changes network addresses, requiring application-layer workarounds to provide the mobility support. Several future Internet architecture designs have been proposed to refactor naming, addressing and routing for location-independent communication [36,40,49,50]. In this chapter, we consider MobilityFirst [49,62], a mobility-centric architecture that supports large-scale, efficient and robust network services with mobility as the norm. In MobilityFirst, we associate every endpoint with a global unique identifier (GUID), which is decoupled from its network address or location, and rely on a dynamic global name resolution service (GNRS), such as those in [43, 67, 73, 74], to track and resolve the mapping between GUID and its network address(es). MobilityFirst aims to support smooth mobile content delivery exploiting real-time GNRS updates and queries.

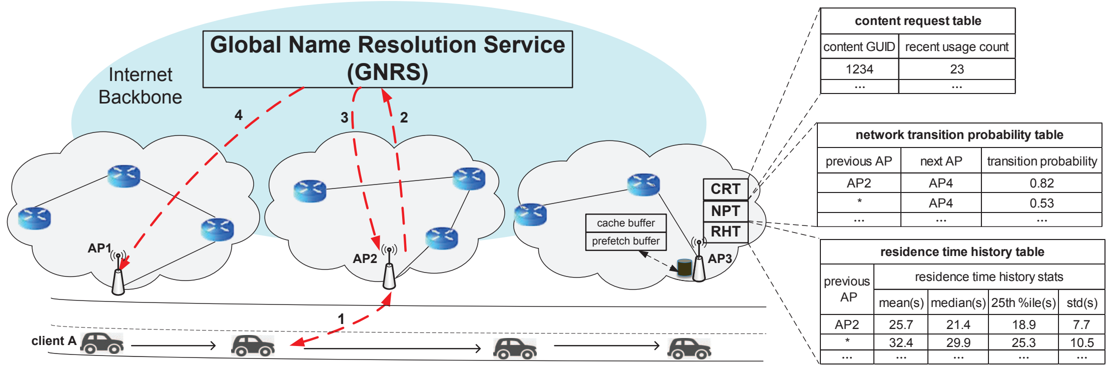
The key to mobile content delivery is to ensure seamless delivery when devices move between networks. In MobilityFirst, a mobile device updates GNRS with its latest network address when it moves to (i.e., connects to) a new access point (AP). We envision that these APs, when equipped with storage, can be utilized as a distributed content caching system (similar to the concept of PacketCloud [20] and EdgeClouds [19]), such that a mobile client downloads chunks of content from APs it connects to over time, thus avoiding reaching out to the original content server. The download performance (i.e., latency, throughput) can be greatly improved when the content chunks are available locally at the AP [19, 20, 24, 25]. However, achieving the above goal is not trivial: we need to determine what chunks should be placed at which AP(s) prior to the client's arrival at that AP. There is a need to both anticipate what content users may request (e.g., popularity) as well as take advantage of the individual user's current content access pattern. To effectively achieve this goal, we propose an *EdgeBuffer* framework in which an AP's storage is separated into cache buffer and prefetch buffer, with the former caching popular content chunks while the latter buffering chunks that are likely to be accessed by individual mobile client. That is, while most popular content chunks at an AP are cached at the cache buffer, those most "urgent" content chunks that are expected to be needed for a client in the near future are prefetched and buffered at the AP's prefetch buffer. Caching popular content chunks has been extensively studied in the literature [8, 26, 88]. In this chapter, we focus on prefetching content chunks for individual mobile devices, by predicting the next network the device is moving to, and at what time, and examine how best to partition the storage at the AP.

In the literature, earlier studies [25, 55] have looked at mobility prediction and content prefetching based upon personalized mobility models. These studies reported that individual mobility patterns are highly predictable in certain areas (e.g., suburban locations), due to limited and predictable human habits (e.g., daily commute) and road conditions. Though these observations are true in less populated areas, we find that they do not hold true in urban areas (e.g., city of San Francisco, CA) due to more complex road networks, frequent traffic congestion, and vehicles such as taxis taking different passengers and thus not following the same route. As a result, personalized (such as daily) mobility patterns are less likely to be useful in these areas. We need to take into consideration the latest mobility information from nearby devices to make accurate predictions. Therefore, in this chapter, we develop network (access point) level mobility models based upon aggregated mobility information collected by the network, which better captures time-varying mobility patterns than personalized mobility models.

MobilityFirst is ideally suited to develop such a network-level mobility model. Since each mobile device updates its latest network address with GNRS when entering a new access network, GNRS naturally observes each device's mobility pattern. Based upon the aggregated mobility patterns from all the devices that pass the network, we build a mobility model for each network – we model user movement as a second-order Markov chain with fallback to first-order [70], and build such network transition probability table (NPT) at each AP. Given the previous AP of a mobile client, the NPT at current AP returns the probable future AP and corresponding transition probability for the client. Each AP also builds a residence time history table (RHT) by extracting statistics from recent residence times observed by the AP. An AP then estimates a particular device's residence time based upon the residence times of past clients.

In this chapter, we have made the following contributions:

- We have developed a content caching and prefetching framework in which an access point has separate cache buffer and prefetch buffer for popularity-based



content request table

content GUID	recent usage count
1234	23
...	...

network transition probability table

previous AP	next AP	transition probability
AP2	AP4	0.82
*	AP4	0.53
...	...	...

residence time history table

previous AP	residence time history stats			
	mean(s)	median(s)	25th %ile(s)	std(s)
AP2	25.7	21.4	18.9	7.7
*	32.4	29.9	25.3	10.5
...	...	...	...	...

Figure 4.1: EdgeBuffer: mobility prediction and content caching/prefetching framework in MobilityFirst

content caching and mobility prediction-based prefetching. Our framework can capture both long-term aggregated content access pattern and short-term individual user access pattern, thus considerably improving the cache hit ratio.

- We have investigated various aspects of the caching and prefetching framework, especially on how to split the storage into a cache component and a prefetch component, and size them correctly, determined by the mobility and access patterns at the AP.

- We have developed a network-level mobility prediction model within the MobilityFirst architecture, taking into consideration latest mobility information from nearby mobile devices. This model is more accurate than prediction based upon individual device traces.

We have generated an access point level user mobility workload based upon a real taxicab trace [60] and the wireless coverage database [75], and use it for simulation. Our evaluation shows that our scheme improves the overall cache hit ratio by 289% and 45% compared to caching only approach and individual user based prediction approach.

4.2 EdgeBuffer: Content Caching and Prefetching at the Edge

With the increasing deployment of wireless access points, a mobile client may achieve better connectivity to retrieve content, but at the same time need to associate with multiple access points during a content retrieval session. Caching and prefetching content chunks at these APs can reduce the content download latencies, improve the throughput, and mitigate jitter. For example, when a mobile client connects to an AP, it issues a request (with a specific content chunk number); the request can be satisfied locally if the requested chunk is already cached at the AP due to its popularity. Additionally, if the network can predict which AP the client will connect to next, and at what time, then the predicted next AP can prefetch suitable chunks before the client arrives. In both of these two cases, the client can directly access the content from the network edge, instead of going to the hosting server or CDN. This is the main idea of EdgeBuffer, which uses the distributed storage at APs to capture long-term aggregated content access patterns (i.e., popularity based), as well as the short-term individual client's access pattern based upon the client's mobility information.

4.2.1 Popularity-based Caching

The access point can build a content request table capturing long-term, local content request pattern as shown in Figure 4.1. The table maintains a recent usage count (RUC) for each chunk of content that has been requested through the AP within a time window. The RUC value is managed using an explicit aging method with a periodic aging function as discussed in [88]:

$$C(x) \leftarrow \alpha * C(x), 0 < \alpha < 1, \tag{4.1}$$

where $C(x)$ is the RUC value of content x. Normally, every time when a content is requested, its RUC value increments by one; periodically (e.g., every aging period), each RUC value $C(x)$ is multiplied by α $(0 < \alpha < 1)$.

When cache buffer is full and a new chunk arrives, the AP compares its RUC value with that of cached chunks. Cache replacement takes place when the new arriving

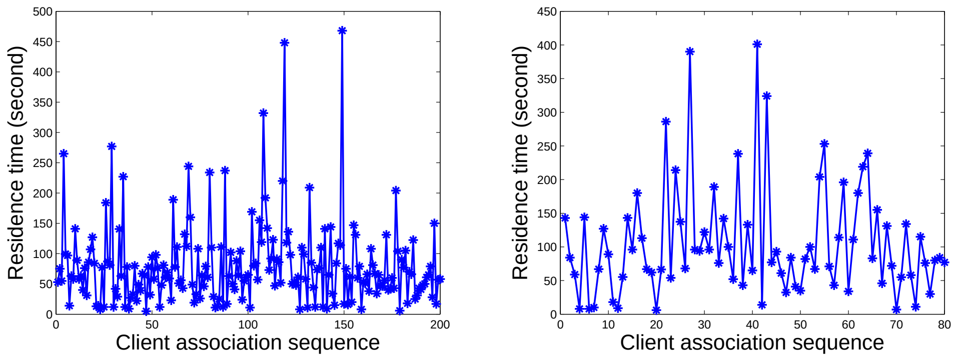


Figure 4.2: Residence times for the same taxi at the same AP vary significantly.

chunk's RUC value is higher than the lowest RUC value of cached chunks.

4.2.2 Mobility Prediction and Prefetching

In addition to caching popular content chunks, we also focus on prefetching chunks for mobile clients. The key to prefetching suitable content chunks before the client arrives at an AP is accurate mobility prediction – as soon as a client associates to an AP, the AP needs to predict how long the client will stay in the current network, and which network it is moving towards.

Previous work has proposed to conduct mobility prediction based upon each individual user's mobility trace [25, 55], with the assumption that user mobility/commute trace is highly regular and predictable. However, we argue that this assumption only holds true in less populated areas, but not true in urban areas with complex road networks and frequent traffic congestions, such as San Francisco, CA. In such urban areas, the time, as well as the route, taken to travel between two locations may vary considerably. To support our point, from SF taxicabs mobility dataset [60] that consists of 536 cabs' GPS trace, we randomly picked two cabs and plotted their residence times at a specified AP in Figure 4.2. Figure 4.2 shows that the residence time for the same cab at the same AP varies significantly in crowded areas. In this chapter, we therefore set out to investigate mobility prediction at the network/AP level using aggregated mobility traces.

Spatial and Temporal Prediction

Conducting mobility prediction at the network/AP level is challenging on today's Internet due to the lack of aggregated mobile network association traces. GNRS in MobilityFirst, however, naturally keeps track of each mobile client's network association trace – whenever a mobile client gets associated to a new AP/network, it updates the name resolution service about the association, thus making network-level mobility prediction feasible.

In this work, we consider mobility prediction in two orthogonal dimensions: spatial prediction (which AP will the client connect to next) and temporal prediction (how long will the client stay at the current AP). That is, spatial prediction determines where to prefetch content chunks, and temporal prediction determines what content chunks to prefetch. To facilitate these two types of prediction, we assume that each AP maintains the corresponding network association records. Each record consists of the following three fields: (1) previous AP, (2) residence time, and (3) next AP. Based upon the association records, the AP builds a network transition probability table and residence time history table, as illustrated in Figure 4.1.

Our spatial prediction predicts a client's next AP given the combination of its previous and current APs. The key data structure for spatial prediction is what we call network transition probability table (NPT). We build NPT from the AP's association records through a second-order Markov predictor with fallback to first-order. NPT takes the (previous AP, next AP) tuple as input, and returns the transition probability to the corresponding next AP.

Our temporal prediction predicts a client's residence time given the combination of its previous and current APs. From an AP's association records, we build its residence time history table (RHT) – we first classify the records into different bins according to the previous AP, and then calculate the statistics of residence times in each bin, including the mean, median, 25 percentile and standard deviation. If the previous AP information is not available, then the prediction falls back to look at all the residence times without binning.

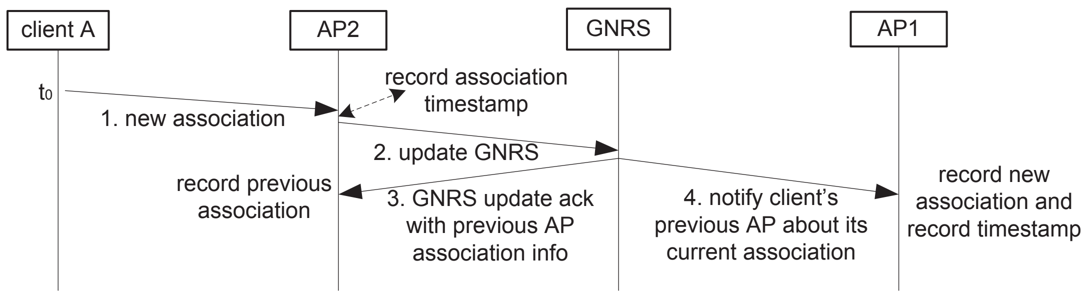

client A
AP2
GNRS
AP1
t_0
1. new association
record association timestamp
2. update GNRS
record previous association
3. GNRS update ack with previous AP association info
4. notify client's previous AP about its current association
record new association and record timestamp

Figure 4.3: Sequence diagram when client A transits to AP2 from AP1

Figure 4.1 and Figure 4.3 illustrate how these two tables are built using an example. In the example, client A was associated with AP1, AP2, and then AP3. After leaving AP1, it connected to AP2, and sent out an update message to GNRS through AP2. GNRS then responded to AP2 with the client's previous location (AP1 in this case), and notified AP1 with the user's current location (AP2). AP2 then logged the association record (without the next AP field) as shown in Algorithm 1, and AP1 filled the corresponding association record's next AP field with AP2, as shown in Algorithm 2. The same process took place after the client left AP2 and connected with AP3. In this way, AP2 figured out each association record's previous AP, next AP, and residence time. Using these association records, AP2 can construct its NPT and RHT.

```
Algorithm 1: Recording client association history
  Input: GNRS update ack message $M_u$, association timestamp $t_{assoc}$
  Output: client association record $R_{assoc}$
  Data: client $c$, previous AP $AP_{prev}$
1 $(AP_{prev}, c) \leftarrow$rcvGNRSAckMsg$(M_u)$
2 $R_{assoc}$.insertPreviousAP$(c, AP_{prev})$
3 $R_{assoc}$.insertAssocTime$(c, t_{assoc})$
```

4.2.3 Content Prefetching Protocol

Using NPT and RHT, an AP predicts when a mobile client will leave the current network (temporal prediction), and to which network it will move to (spatial prediction).

Algorithm 2: Building network transition probability table (NPT) and residence time history table (RHT)

```
Input: GNRS notification message $M_n$, the time receiving the msg $t_{leave}$, client
        association record $R_{assoc}$
Output: $NPT$, $RHT$
Data: client $c$, previous AP $AP_{prev}$, next AP $AP_{nxt}$
1 $(AP_{nxt}, c) \leftarrow$rcvGNRSNotiMsg$(M_n)$
2 $AP_{prev} \leftarrow$ getPreviousAP$(R_{assoc}, c)$
3 $t_{assoc} \leftarrow$ getAssocTime$(R_{assoc}, c)$
4 $\Delta t \leftarrow t_{leave} - t_{assoc}$
5 $R_{assoc}$.remove$(c)$
6 $NPT$.update$(AP_{prev}, AP_{nxt})$
7 $RHT$.update$(\Delta t, AP_{prev})$
```

Following prediction, the predicted next network conducts content prefetching. Below we explain our prefetching protocol using the example shown in Figure 4.4:

1. *Association:* Let us assume that at time t_0, a client leaves AP0 and makes an association with AP1.

2. *Temporal Prediction:* As soon as the association is made, AP1 predicts the client's residence time, T_r, as the median value of the residence time samples that are considered. Since we need start prefetching before the estimated departure time $(t_0 + T_r)$, AP1 sets T_p as the 25th percentile of the residence time samples. As such, the system will start prefetching around time $t_0 + T_p$. Please note that past residence time statistics are available in the RHT.

3. *Spatial Prediction:* At time $t_0 + T_p$, if the client is still associated with AP1, AP1 then predicts the client's next AP by looking up the NPT and picking the K (usually $K < 3$) most likely next APs (as shown in Algorithm 3). In this example, we have $K = 2$, and next APs are AP2 and AP3.

   At the same time, AP1 examines which chunk the client is going to download next

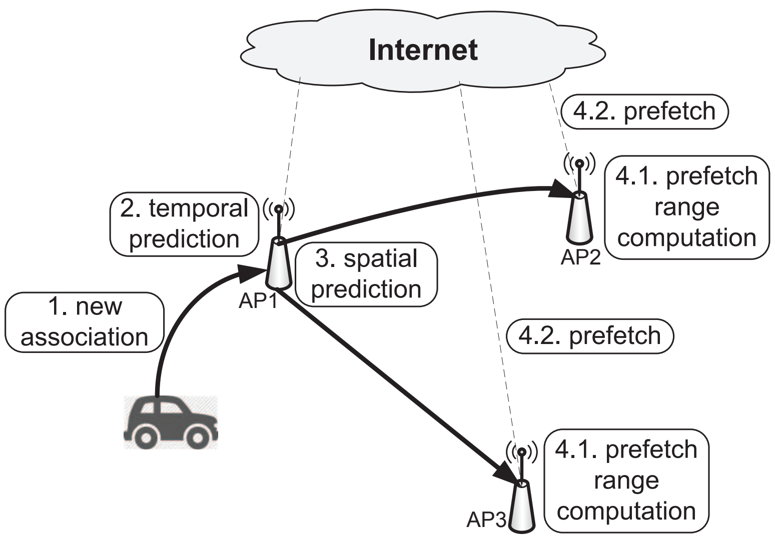


Figure 4.4: Content prefetching protocol

at the estimated departure time $(t_0 + T_r)$, which will also be the first prefetch chunk.

Finally, AP1 sends out a *Prefetch message* to K next APs. The Prefetch message contains the following information: content ID, chunk ID, and the transition probability.

4. *Prefetching:* Suppose the Prefetch message reaches AP2 and AP3 after a short delay. Both APs predict the residence time for the client, and calculate what chunks the client will request during the estimated residence time as shown in Algorithm 4. If these chunks are not yet cached in the popularity-based cache, the APs will fetch them right away. The prefetch buffer uses LRU as the replacement policy.

4.3 Evaluation

We have developed a trace-driven simulator and conducted detailed evaluations of the proposed mobile content delivery scheme.

Algorithm 3: Predicting client mobility at an AP

```
Input: client $c$, elapsed time since client's association $t_0$, previous associated AP
       $AP_{prev}$, current transferring chunk ID $chk_0$ of content $cont_c$, transition
       probability threshold $P_{thres}$
Output: Prefetch message

1  if $checkClientAssoc(c) == true$ then
2      $t_1$ ←getResiMedianbyRHT($AP_{prev}$)
3      $\Delta t \leftarrow t_1 - t_0$
4      $chk_s$ ←calcPrefetchStartChk($cont_c, chk_0, \Delta t$)
5      $P_v$ ←getNxtAPVectorbyNPT($AP_{prev}$)
6      $P = 0$
7      $i = 0$
8      while $P < P_{thres}$ do
9          $(AP_{nxt}, P_{transit})$ ←getAPwithTopProb($P_v, i$)
10         SndPrefMsg($AP_{nxt}, cont_c, chk_s, P_{transit}$)
11         $P \leftarrow P + P_{transit}$
12         $i \leftarrow i + 1$
13     end
14 end
```

Algorithm 4: Calculating prefetch range

```
Input: Prefetch message $M_p$, popularity cache buffer $B_{cach}$, prefetch buffer $B_{pref}$
Output: prefetch chunk range

1  $(AP_{src}, cont_c, chk_s)$ ←rcvPrefMsg($M_p$)
2  $\Delta t \leftarrow$ getResiMeanbyRHT($AP_{src}$)
3  $chk_e \leftarrow$ calcPrefetchEndChk($cont_c, chk_s, \Delta t$)
4  $chk\_range$ ←checkMem($B_{cach}, B_{pref}, cont_c, chk_s, chk_e$)
5  if $chk\_range \neq 0$ then
6      prefetch($chk\_range, cont_c$)
7  end
```

4.3.1 Simulation Setup and Workload Generation

We developed a discrete event-driven simulator to evaluate EdgeBuffer. We generated realistic taxi mobile access workload by integrating a GPS trace of taxicabs in San Francisco, USA (SF dataset) [60] and a wireless AP coverage database in SF (WiGLE database) [75]. The SF dataset contains GPS coordinates of 536 taxis for around 3 weeks in the San Francisco Bay Area. In the original trace, location update interval was mostly 1 minute, and we inserted finer-grained location updates (an update interval of 1 second) to the original dataset. The WiGLE database provides the AP deployment map in SF, and we cleaned up the database by merging APs that are within a certain distance of each other. In this study, we focus on two distance thresholds: 250m and 100m, referring to the resulting workloads as SF-250 and SF-100 respectively. Specifically, there are 687 APs in SF-250, and 2589 APs in SF-100 workload.

In the simulator, we use one node to represent the content hosting server and another node to represent the name resolution server. Each AP (corresponding to a node in simulator) is connected to both servers through a 50Mbps link with 20 millisecond delay. We further assume that a client is always associated with the closest AP, and sequentially downloads content chunks from the connected AP with an average link bandwidth of 2Mbps and 4Mbps for SF-250 and SF-100, respectively. The content server contains 100,000 files. Each file is 100 MB in size and has 100 chunks. The popularity of the content follows a Zipf distribution with a default parameter of 0.75, consistent with the observations in the web request traces [10] and Akamai CDN request traces [29]. In the simulations, we assume each AP has 10GB of storage for content caching and prefetching, with 8GB and 2GB allocated for each by default.

4.3.2 Prediction Accuracy

We first evaluate the accuracy of spatial prediction and temporal prediction for our network-level prediction (netPredict) and individual user based prediction (userPredict). Note that to implement user based prediction, we use a taxi's AP transition

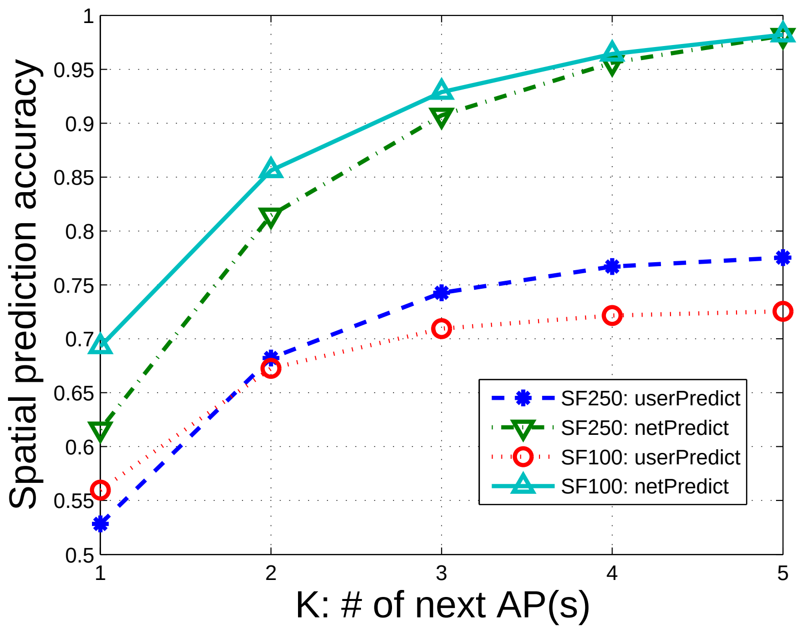
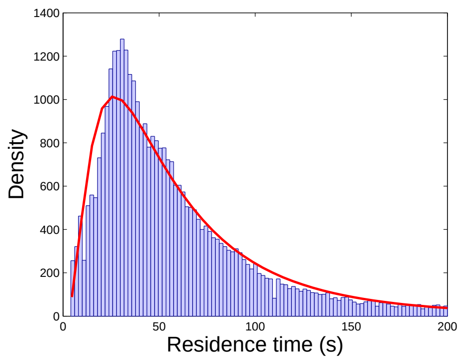


Figure 4.5: Spatial prediction accuracy when picking different number of next AP(s). Predicting next AP can achieve high accuracy with a small K value using our network-level prediction.

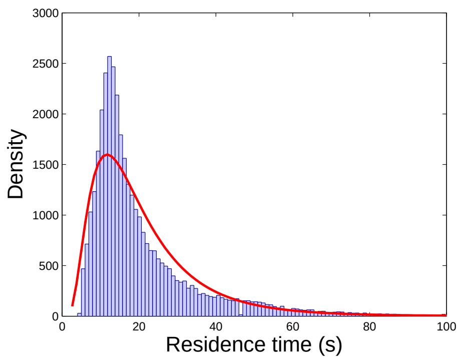
history to predict the next AP(s) that it will move to, and use the taxi's history residence time at an AP to predict its residence time at this AP.

We measure the spatial prediction accuracy as the percentage of the incidents in which a client indeed moves to one of the K predicted AP(s). We report the result in Figure 4.5. The figure shows that our network-level prediction has a much higher prediction accuracy than the user based prediction. Moreover, using our network-level prediction method, we can achieve a very high spatial prediction accuracy with a small K value (e.g., the value is 0.929 and 0.907 with $K = 3$ for SF100 and SF250). In the following simulations, unless otherwise specified, we set the transition probability threshold P_{thres} in Algorithm 3 as 0.8, which corresponds to $K = 2.46$ in average.

Unlike spatial prediction, accurate temporal prediction is much harder due to the high variation in residence times. To support our point, we randomly picked one AP from SF-250 and SF-100 workloads and plot the histogram of the AP's residence times



(a) SF-250



(b) SF-100

Figure 4.6: Histograms of residence times of an AP. The smooth curve is a log-normal density distribution.

in Figure 4.6. The histogram shows that the residence time at an AP follows a log-normal distribution, which is in agreement with the conclusion in [41], in which the authors discovered that the residence times in a campus environment follow a log-normal distribution.

In our temporal prediction, we predict a client's residence time as the median value of history residence time samples. We then measure the temporal prediction error as the average difference between the estimated residence time and the actual value. Using this definition, we investigate the estimation error at different history sample size, as seen in Figure 4.7. The results show that our prediction method is consistently more accurate than user based prediction because it can dynamically capture the traffic condition. Moreover, a modest history sample size is needed to get the optimal prediction accuracy. Specifically, at sample size of 50, the estimation error for our network-level prediction is 25.2s and 12.8s, while the estimation error for individual user based prediction is 30.6s and 15.8s, for SF-250 and SF-100. The prediction for SF-100 is better than SF-250 because of a smaller AP range. We use this history sample size as the default value.

We have also studied the prediction accuracy for a Beijing traxi drive trace [81], and the results show very similar trends. In the interest of space, we do not present these results here.

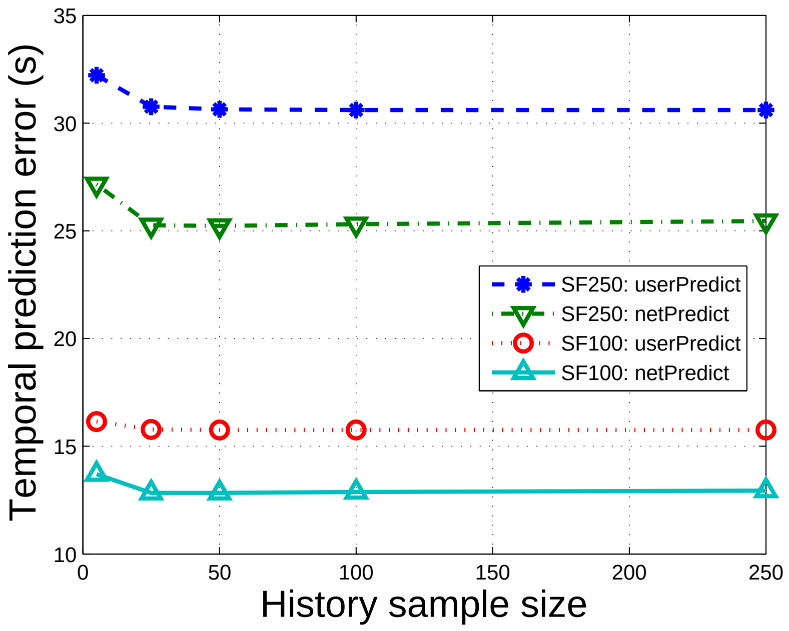


Figure 4.7: We evaluate temporal prediction error when the residence time history table uses different history sample size.

4.3.3 Hit Ratio at the Edge Buffers

We next evaluate the benefits of having caching and prefetching in the system. When a content request is satisfied at the AP's edge buffer, the multi-hop Internet path is replaced by a one hop path, the round-trip time is thus much smaller, and a higher throughput can generally be obtained. Therefore, the hit ratio at the edge buffers characterizes the performance gain of caching and prefetching regardless of different network settings (i.e., network topology, backhaul bandwidth and delay, wireless condition, etc), we thus focus on the cache hit ratio as the main metric here. Note that this metric is the "cache" hit ratio in terms of content chunks which includes hits from both the cache and the prefetch buffer.

Comparison of Four Edge Caching Strategies

We first compare the performance of four edge caching strategies: (1) *LRU*, in which we use the entire AP storage as a LRU cache, (2) *popCache*, in which we use the entire AP storage as a popularity-based cache, (3) *popCache+userPredict*, in which we

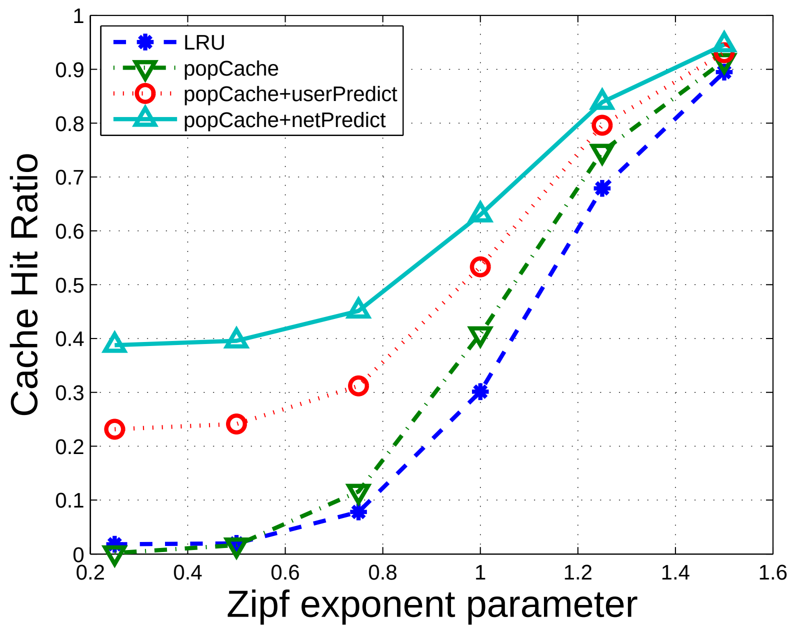


Figure 4.8: Cache hit ratio for four edge caching strategies (with SF-100 workload).

partition the storage at an AP into two parts: a popularity-based cache and a prefetch buffer based upon user-level mobility prediction, (4) *popCache+netPredict*, in which we partition the storage at an AP into two parts: a popularity-based cache and a prefetch buffer based upon AP-level mobility prediction. In EdgeBuffer, we adopt the fourth strategy, i.e., popCache+netPredict.

We report the performance of the four strategies with different Zipf exponent parameters using SF-100 workload in Figure 4.8. Further, for popCache+userPredict and popCache+netPredict, we break down the total cache hits into hits from cache and hits from prefetch buffer, and show the breakdown in Figure 4.9. In fact, we have results for both SF-100 and SF-250 workloads, but choose to only show one because the trends are very similar. From the results, we have the following observations. First, with reasonable Zipf exponent values, prefetching is more effective than caching. As Zipf exponent becomes larger, the content access popularity distribution becomes much sharper, caching becomes much more important than prefetching, and the difference between LRU cache and popularity-based cache becomes less important. As a result, all four strategies converge when we have large Zipf exponent values. Second, our strategy

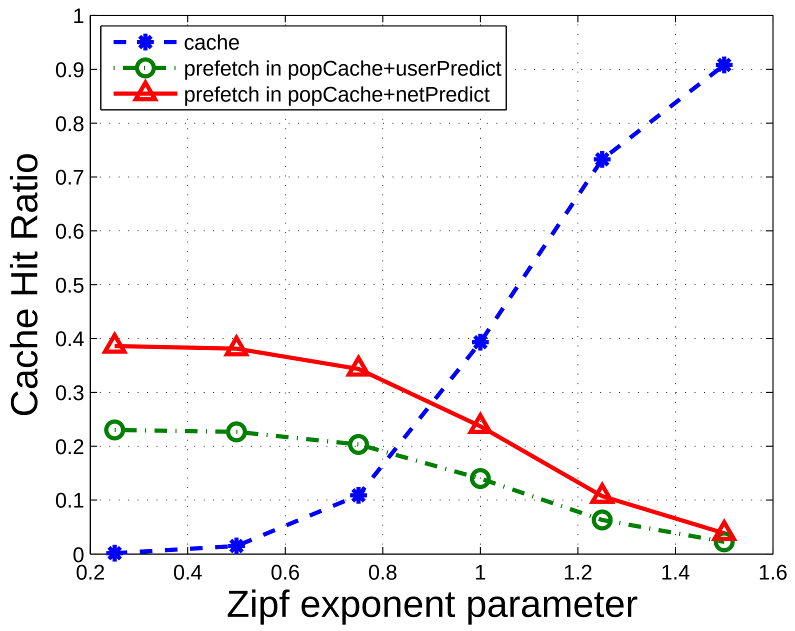


Figure 4.9: For popCache+userPredict and popCache+netPredict, we break down the total cache hits into hits from cache and hits from prefetch buffer. Since the contributions from the cache are almost identical, we only show the cache contribution for popCache+netPredict here.

always fares better than the other three in achieving local hits at the edge. The detailed breakdown results show that our AP-level prefetching is more effective than individual user based prefetching.

Next let us look at the detailed performance when the Zipf exponent parameter is 0.75, which is a common value seen in real web server access [10]. Here, the hit ratios are 0.078, 0.116, 0.312(0.109 from cache and 0.203 from prefetching), and 0.451(0.108 from cache and 0.343 from prefetching) respectively for SF-100. The results show that the average hit ratio in our scheme gains 478% improvement over the traditional LRU cache scheme. Compared with popularity cache scheme, there is 289% improvement in hit ratio, which shows the importance of prefetching. Compared to individual user based prefetching, our AP-level prediction can lead to a 45% overall improvement. If we only focus on the hits due to prefetching, our scheme improves 69% over user-based prediction, further proving that our network aggregated prediction is more effective in

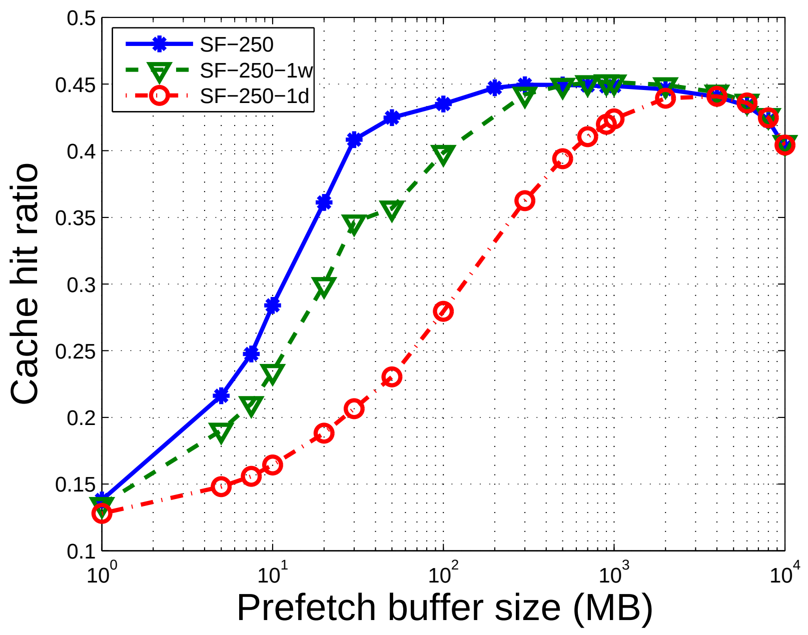


Figure 4.10: Hit ratio with different prefetch buffer sizes. Note that an AP has a total of 10GB storage.

urban settings.

Size Allocation between Cache and Prefetch Buffer

Next, we look at the impact of different cache and prefetch buffer size combinations, given that the sum of these two is fixed. In this set of simulations, we set the Zipf exponent parameter to be 0.75, and vary the load in the system. The original SF-250 workload has around 3 weeks of data with 536 taxis and 687 APs. We increased the load by compressing the total number of taxis that appeared across three weeks into one week and further into one day, thus getting workloads SF-250-1w and SF-250-1d.

We report the results in Figure 4.10, and have the following observations. First, there exists an "optimal" size for the prefetch buffer – after the prefetch buffer reaches a certain size, cache hit ratio starts to drop. Second, the optimal size varies with the load of the system. A lighter load requires a smaller prefetch buffer: SF-250 shows an optimal prefetch buffer at 3% of the total storage capacity, while SF-250-1d shows an optimal prefetch buffer at 40% of the total storage capacity. Third, the penally of

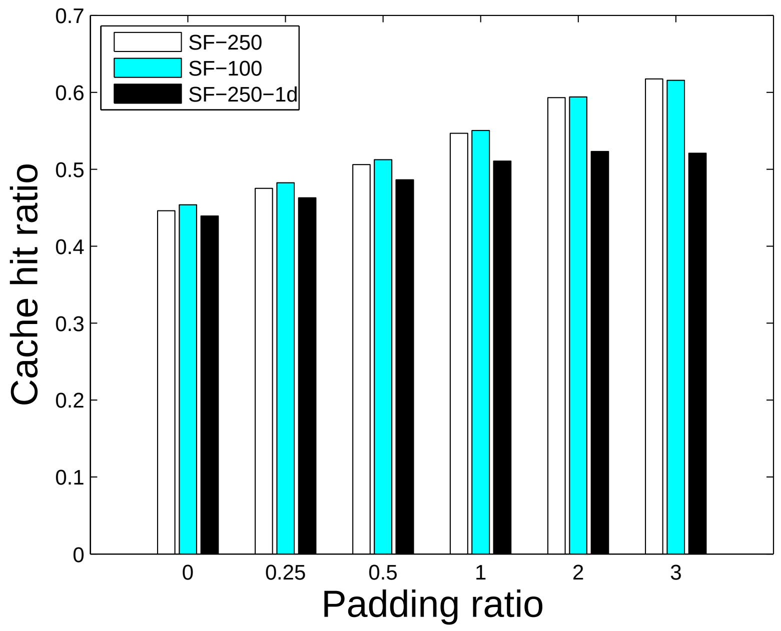


Figure 4.11: Hit ratio at different padding level.

having a prefetch buffer that is too smaller is much larger than the penalty of having a prefetch buffer that is too big. This is because when the Zipf exponent is 0.75, and the content space is reasonably large, prefetching plays a bigger role than caching.

We have also derived the guideline on how to size the prefetch buffer: $B_{pref} >= \Delta t * K * \rho * n * M$, where Δt is the interval between the time when a chunk is prefetched and the time when the chunk is requested by the client ($\Delta t = t_1 - t_0$ as in Algorithm 3), K is the number of next AP(s) conducting prefetching (2.46 in our case), ρ is the mobile client arrival rate, n is the number of chunks prefetched for an arrival client, and M is the size of a chunk. This guideline ensures the prefetch buffer is large enough so that the prefetched chunks will not be replaced before they are consumed by the client. The resulting buffer size matches our simulation results well.

Prefetching More Chunks

In this set of results, we explore the impact of prefetching more chunks than what is given by Algorithm 4. Prefetching more chunks may increase the hit ratio if our residence time estimation is erring on the small side, but it may evict useful chunks if

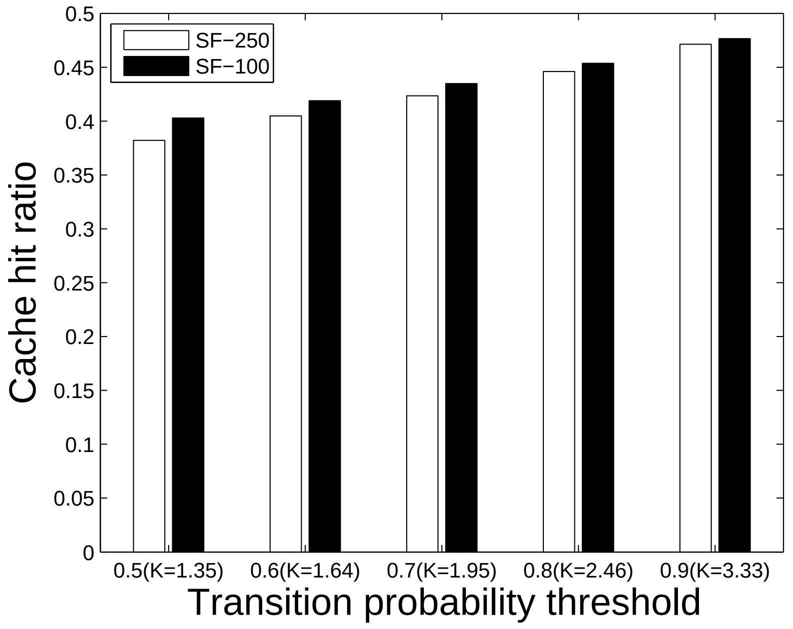


Figure 4.12: Hit ratio at different transition probability threshold. We also calculate the average value of K at each threshold.

the prefetch buffer is full[1]. Here, we introduce the parameter, *padding ratio*, to measure the amount of extra chunks that are prefetched. Assuming that we are supposed to fetch N chunks, a padding ratio of p means that $\frac{pN}{2}$ extra chunks are prefetched at both sides of the prefetch range.

We report the cache hit ratio at different padding levels in Figure 4.11, which shows the hit ratio goes up as we fetch more chunks. The hit ratio is about 38.4%, 35.7% and 18.5% higher with 300% padding ratio for SF-250, SF-100 and SF-250-1d. The main reason is that under the default simulation configuration, the prefetch buffer is large enough to accommodate prefetching extra chunks for SF-250 and SF-100. However, with heavier traffic load, prefetching too many chunks for each client will hurt the overall hit ratio, as seen from the degrading performance of SF-250-1d after padding ratio of 2.

[1]Prefetching consumes backhaul bandwidth in addition to AP's buffer storage, thus causes bandwidth waste if the chunks prefetched are not used. Such impact is not evaluated in this chapter.

Network Transition Probability Threshold

We investigate the impact of different transition probability threshold on the caching performance. The transition probability threshold determines the number of top probable next AP(s) that will conduct the prefetching, thus vastly affect the prefetching overhead and caching hit ratio.

The results are shown in Figure 4.12. The values for SF-250 are: 0.382, 0.405, 0.423, 0.446, 0.471; the values for SF-100 are: 0.403, 0.419, 0.435, 0.454, 0.477. The results show that we can achieve good performance with a small K value (like 1.35), such that the prefetch overhead is kept modest.

4.4 Related Work

4.4.1 Caching and Prefetching

Traditionally, content caching [8, 88] and CDN [26] are utilized to facilitate content distribution and retrieval. They normally measure the long-term content access pattern and cache the most frequently used or most recently used contents. However, most of such work doesn't consider about the impact of short-term user mobility on caching performance.

On the other hand, prefetching [27, 48, 78] has been proposed to proactively preload data from server instead of passively waiting on content requests. However, it is often used in the context of prefetching related contents that are expected to be used in the near future according to user's access pattern or content's structural relationship. Authors in [24, 25, 55] proposes to prefetch parts of large content object to different AP locations by utilizing mobile client's mobility pattern. While Sprinkler [24] assumes that mobile client's route is known as prerequisite, other work all requires mobility prediction. Earlier studies in [25, 55] proposes a personalized mobility model where each mobile client predicts its future connection based on its own history, while authors in [41, 80] have demonstrated to use traces of several users to build user mobility model in a college campus environment. We argue that a network-level aggregated prediction model is preferred for a complicated urban environment. Incidentally, in [55, 70, 80], authors all use a second-order Markov model to predict mobile client's future location.

4.5 Concluding Remarks

In this chapter, we present the design of EdgeBuffer, which aims to improve mobile client's content download performance by leveraging aggregated network-level prediction and prefetching as well as popularity-based caching at the network edge. Through detailed simulations, we show that such a scheme can significantly increase the fraction of content requests that can be satisfied at the edge, avoiding long latencies involved in reaching out to the original hosting server or CDN server. Our results also shed light on important issues such as how to allocate the size between cache buffer and prefetch buffer, when to prefetch more chunks, and by how much.

Chapter 5

Edge Caching and Nearest Replica Routing in Information-Centric Networking

We take a step back for caching at this chapter by investigating different caching and request forwarding schemes in general ICN architecture [85]. We present our comparison methodology, simulation results and its implications here.

5.1 Introduction

Nowadays, Internet usage is dominated by content distribution and retrieval. The idea of *information-centric networking* (ICN) [2] (e.g. MobilityFirst [49], NDN [52]) has been intensively discussed recently to better support content-oriented services from a clean-slate network architecture perspective. ICN proposes in-network caching in which the router has a cache store component in addition to the routing table. Such ICN router can cache pass-by data and store the content for the longest useful time to satisfy subsequent requests locally in its cache. The ICN network thus becomes a network of caches, instead of a simple interconnection of nodes in traditional IP network.

As essentially a cache network, ICN can be characterized by two dimensions: *(C,F)*, where C is the content caching scheme, and F is the request forwarding strategy. The content caching scheme determines where to place the cache storage, whether a cache node should cache a content, and what content to replace if the cache store is full. A service provider can place the cache storage only at the edge of the Internet, or distribute it to internal nodes (pervasive caching in extreme). When data returns, certain on-path cache nodes could cache the content following certain caching algorithm, e.g., leave copy everywhere (LCE), leave copy down (LCD) [45], select node with highest betweenness [18]. Additionally, a cache store normally uses LRU or LFU replacement policy. For request forwarding strategy, the router could forward a content request following the shortest path routing (SPR) towards the original content server while the on-path content cache is explored to serve the request; more proactively, the router could directly forward the request to a nearby content cache if the router knows the cache information, resulting in nearest replica routing (NRR).

ICN originally advocates pervasive caching and nearest replica routing. However, a debate is recently reignited around the necessity of ICN, especially the usefulness of pervasive caching. Authors in [29, 33] claim that edge caching already achieves most of the gain, which is the approach CDN normally takes (e.g. Akamai's content delivery system [56], Facebook's photo storage system [7]), while other work [64] suggests ICN caching performance could be further improved. It becomes an urgent need to better understand the pros and cons of different content caching and request forwarding design choices in ICN: whether pervasive caching and nearest replica routing could each bring significant benefits? is there an ICN architecture that achieves most of the gain and is relatively easy to deploy?

Towards this end, we summarize several design options to represent the overall design space and compare these designs through extensive simulation at both single domain network and multi-domain network scenario. From the comparison, we achieve the following observations:

- Pervasive caching may not bring about substantial benefits as ICN proposes, while a simpler edge caching configuration could already achieve most of the gain. We find that pervasive caching can achieve at most 20.6% larger network capacity in our experiments.

- Nearest replica routing ensures that a request is served from the "best" location, thus helps optimize content delivery, especially for a large network.

- For a system with basic edge caching and shortest path routing functionality, enabling nearest replica routing brings larger performance gain (e.g. 98.4% larger network capacity) than enabling pervasive caching.

Based on these observations, we propose a framework that adopts edge caching and supports nearest replica routing by having a separate cache control plane from data forwarding plane. Our framework is based upon MobilityFirst future Internet architecture [49] which utilizes a global name resolution service (GNRS) to manage the mapping between content name and its locations. Nearest replica routing is realized by querying GNRS and retrieving the update-to-date content locations efficiently. We evaluate our framework and show that it can approximate nearest replica routing quite well and is feasible to deploy at the global scale.

The rest of the paper is organized as follows. Section 5.2 talks about the background of ICN. An anaysis of the content caching and request forwarding design space is given in Section 5.3, and Section 5.4 presents the simulation results in comparing different designs. Our framework is then presented in detail in Section 5.5. Related work and concluding remarks are given in Section 5.6 and Section 5.7.

5.2 Background of ICN Caching

ICN is built upon a name-based service model: name is decoupled from locations, and network communication is based on name in contrast to today's host-centric abstraction. ICN also advocates pervasive caching where every ICN router is equipped with cache store. In addition, nearest replica routing presumably becomes natural implied by the named-based service.

However, there has been lots of discussions recently about whether the ICN design, especially pervasive caching, brings a lot more benefits than its cost [18,29,33,64]. While enabling every router as content cache and satisfying request from nearest possible cache location seems perfect ideally, such design might not well-suited for heavy-tailed request workload in reality [10], not to say about the overhead of deploying it in practice. Authors in [29] show that the maximum performance gain of a full-fledged ICN architecture is at most 9% over an edge-based caching architecture and thus propose an incrementally deployable ICN architecture, while in [64], the authors argue that the performance of ICN pervasive caching could be further improved when using a better caching algorithm, e.g. LCD instead of LCE. We thus revisit this issue and compare different designs through detailed evaluation at different network scenario, cache size, and request distribution.

5.3 Design Space For ICN Caching

As described in Section 5.1, ICN can be characterized by two orthogonal dimensions: content caching scheme and request forwarding strategy. The first dimension includes cache placement in the network and cache resource management at each cache node, while the request forwarding strategy determines how content requests are forwarded through the network, i.e., whether the request is forwarded to a nearest replica or directly forwarded towards the content source. We thus choose the following design options to represent different caching schemes in the ICN design space:

- **PERV-LCE-SPR**: this design adopts pervasive cache placement and "leave copy everywhere" caching algorithm, i.e., each node in the network has a cache store, and all nodes on path cache the content when the data returns. In addition, it assumes shortest path routing towards the original server for request forwarding.
- **PERV-LCD-SPR**: this design is the same as PERV-LCE-SPR except the caching algorithm: "leave copy down" instead of LCE, i.e., only the next node on path caches the content when the data returns.
- **PERV-LCE-NRR**: this design extends PERV-LCE with nearest replica routing in which we assume that we can always find an ideal route to the nearest replica for a content request.
- **PERV-LCD-NRR**: this design extends PERV-LCD with nearest replica routing.
- **EDGE-SPR**: in this design, we only place cache at the "edge" of the network, e.g., edge nodes of a tree topology. Each cache node has the same caching size as in the pervasive caching design.
- **EDGE-NRR**: it is the same as EDGE-SPR, but uses NRR forwarding.

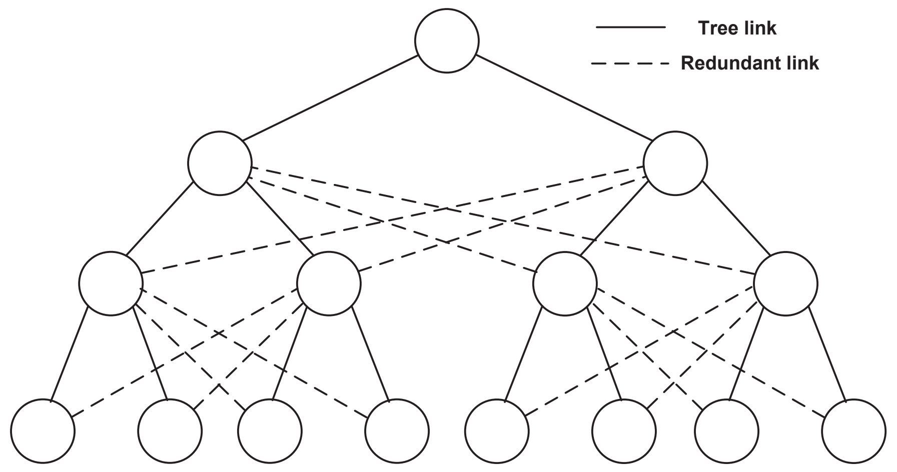


Figure 5.1: 4-level binary tree with redundant links. Dashed links are redundant links, each of which is present with probability μ.

- **EDGE-SPR-NORM**: it uses the same caching and request forwarding mechanism as EDGE-SPR. However, the total cache budget in the network is equal to the total cache in the pervasive caching design. Thus for a binary tree topology, cache size at each cache node doubles.
- **EDGE-NRR-NORM**: it shares the same scheme as EDGE-NRR, and has same total cache budget as pervasive caching.

In our discussion, all designs use LRU as the cache replacement policy.

5.4 Comparison of ICN Designs

We implement a discrete-event simulator to compare the cache performance of alternative designs described in Section 5.3.

5.4.1 Simulation setup

We consider a 6-level binary tree network with redundant links from child to aunt. Each redundant link is present with i.i.d probability $\mu \in [0, 1]$, as shown with dashed lines in Figure 5.1. No redundant link exists by default. We also investigate a large scale network topology with core nodes and expanded access tree attached to each core

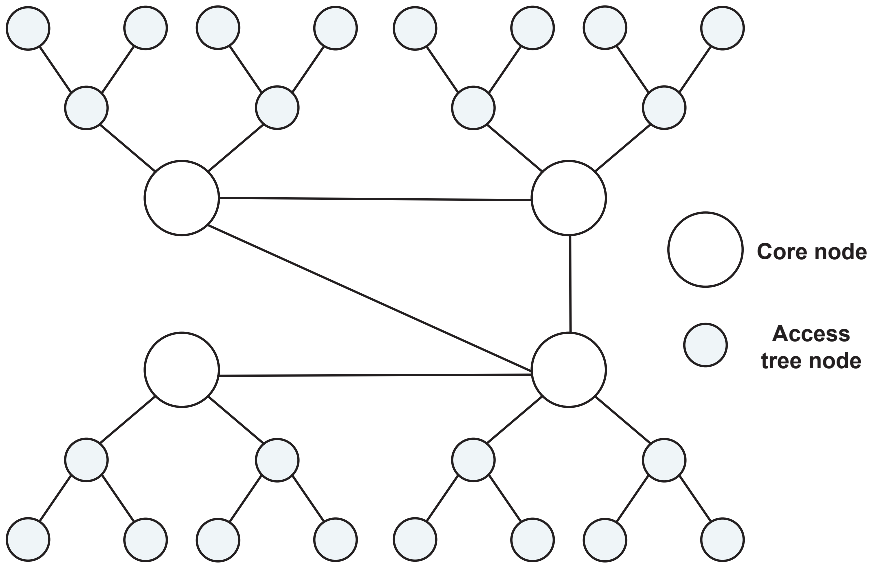
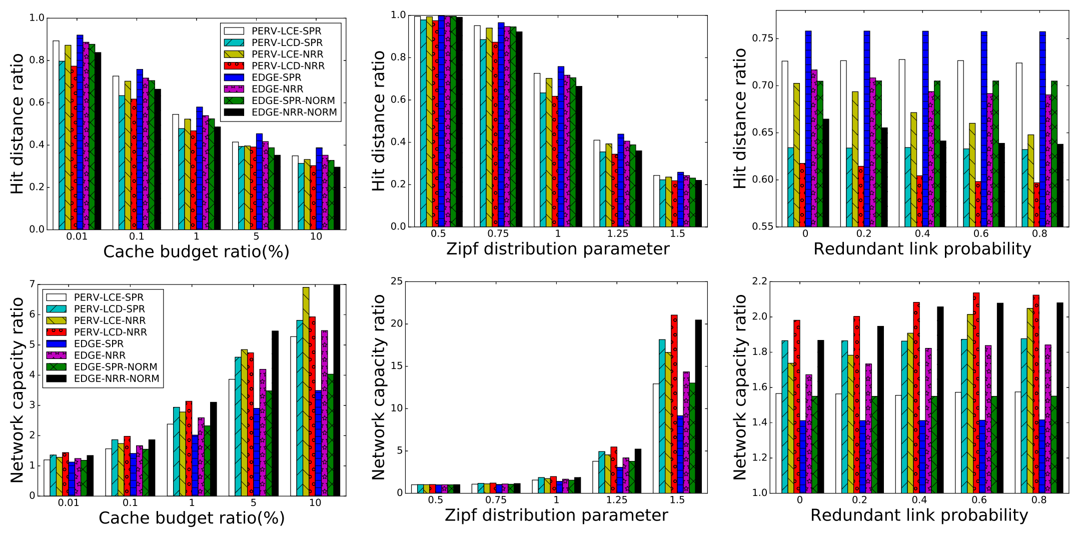


Figure 5.2: A core-tree topology example with four core nodes and their corresponding access tree.

node, as shown in Figure 5.2. All clients are connected to the edge nodes of the tree, one for each edge node.

The content catalog contains 100,000 files, 100 MB in size for each. For the tree topology, all files are originally placed at the root node; for the core-tree topology, the core nodes are the original servers and each file is independently and uniformly distributed among these nodes. We use homogeneous sized caches, with cache to catalog size ratio of 0.1% by default, i.e., each cache can store 100 content out of 100,000 catalog. For EDGE-SPR-NORM and EDGE-NRR-NORM, in order to make the total cache budget in the network equal to pervasive caching design, the cache size at each edge node is multiplied by an appropriate constant (e.g., 2 for binary tree). Each client issues content requests as a Poisson process. The popularity of the content request follows a Zipf distribution with a default parameter of 1, consistent with the observations in the Akamai CDN request traces [29].

We report data distance ratio and network capacity ratio as the evaluation results. The data distance is the average distance from the client to the location where the content is found, and the network capacity is measured as the maximum network



(a) cache to catalog size ratio (b) content popularity parameter (c) redundant link probability

Figure 5.3: Cache performance at different cache budget ratio, content popularity distribution parameter, and redundant link probability for the tree topology, with default value as 0.1%, 1, and 0, respectively. The results show that the performance difference between pervasive caching design and edge caching design is small.

throughput when each client issues request at the same rate. The data distance ratio and network capacity ratio are normalized metrics of the system compared with a system without any cache.

5.4.2 Small-scale network scenario

We use a 6-level binary tree topology with 1Gbps link bandwidth and 1ms link delay for the small-scale network scenario. We investigate the cache system performance at different cache budget ratio, content request popularity parameter, and redundant link probability. The cache budget ratio is defined as cache to catalog size ratio at a node.

Figure 5.3(a) shows the average data distance ratio and network capacity ratio over no cache design at different cache budget ratio. The network capacity is 2Gbps for no cache design. The results show that with larger cache budget, more requests are satisfied at lower level of the tree, resulting in smaller data distance and larger network capacity. Moreover, the difference of cache performance between pervasive caching design and edge caching design is relatively small, e.g. compared with EDGE-SPR-NORM, PERV-LCD-SPR reaches at most 10.1% reduction in data distance and 20.2% increase in network capacity under default cache budget ratio. Thirdly, while NRR always achieves better performance than SPR, pervasive caching might not always beat edge caching, e.g. with large cache budget, EDGE-NRR-NORM design performs even better than PERV-LCD-NRR. Lastly, EDGE-NRR-NORM has the best performance in most cases.

Figure 5.3(b) and 5.3(c) show the cache performance at different Zipf request distribution parameter and link redundancy probability respectively, from which we can reach similar observations. In particular, Figure 5.3(c) shows that the system has better cache performance when adding more redundant links, but the gain is small, implying that cache hits at edge contributes the most to the cache benefits.

These results show that the advantage of pervasive caching turns out to be rather limited, which is indeed reasonable: for either pervasive caching or edge caching, most popular contents are cached at the edge which accounts for most of the cache hits; less popular contents might be cached at intermediate nodes farther away for a pervasive caching design, however, caching less popular content at far-away locations doesn't help much, especially for heavy-tailed content distribution like Zipf; for fixed cache budget, putting all the cache at the edge could bring more benefits than spreading the budget across the whole network.

5.4.3 Large-scale network scenario

We use Abilene topology [63, 71] (shown in Figure 5.4) as core nodes, and generate 6-level binary access tree attached to each core node. We consider access tree with no redundant links in this evaluation. Each access tree could be regarded as a network domain in this scenario. We thus change the ratio of inter-domain link delay over intra-domain link delay while fixing the intra-domain link delay as 1ms, and investigate the cache network performance at different link delay ratio.

Figure 5.5 shows the average data distance ratio and network capacity ratio over

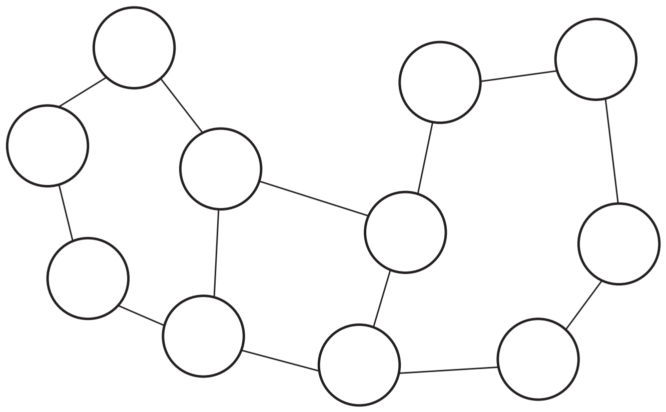


Figure 5.4: Abilene topology.

no cache design at different inter link to intra link delay ratio. The results illustrate that EDGE-NRR-NORM achieves the best performance among all designs, especially with larger inter-intra link delay ratio. The reason is that for larger inter-domain link cost, the gain of retrieving a content from a nearby domain's cache instead of from the source domain becomes more significant. Moreover, the performance gain of using NRR over SPR is much larger than the gain of using pervasive caching over edge caching. In particular, given a specific inter-intra link delay ratio, the gain of using PERV over EDGE is at most 15.9% for data distance and 20.6% for network capacity under same request forwarding strategy, while gain of using NRR over SPR is 39.7% and 119% under same caching scheme.

5.5 Approximate Nearest Replica Routing

From the evaluation in Section 5.4, we conclude that pervasive caching might not achieve significant gain over edge caching, while nearest replica routing with edge caching could bring the most benefits. We, therefore, propose a framework built upon MobilityFirst future Internet architecture that adopts edge caching and approximates nearest replica routing. Our framework could be deployed incrementally rather than requiring cache capability at all nodes.

MobilityFirst utilizes a global name resolution service (GNRS) to manage the mapping between content name and its location(s). When a content is cached at an edge node, it updates the GNRS. The GNRS thus maintains an updated list of locations that the content resides at. When a client requests a content, the request for its first chunk would cause a GNRS lookup so that a list of content locations is returned, thus a closest copy could be selected; for subsequent chunks, the request simply specifies the destination and follows the previous path. Note that when the request reaches the destination, the cache copy might be replaced, in this case another GNRS resolution is required. Utilizing GNRS as a separate control plane that manages name to location mapping makes nearest replica routing feasible.

5.5.1 Feasibility at Internet Scale

GNRS design

As a logically centralized but physically distributed infrastructure, GNRS [74, 83] is structured in multiple levels: a local name resolution service at each domain/AS, and the global-level name resolution service. The local name service contains the mappings of local objects within the AS which is composed by one or multiple central servers, while the global-level GNRS is designed as an in-network 1-hop DHT with servers distributed all across the Internet. Each server in this distributed service advertises responsibility to a portion of an orthogonal name-space, which is used to designate (using consistent hashing) a host for mappings of a particular GUID. The structuring design of GNRS is aimed to support response latencies lower than 100ms for all these operations in order to enable real-time applications for mobile hosts. Such objective is validated in one manifestation of GNRS scheme [74]. Note that at global GNRS level, the GNRS server only records the AS(es) that the object is located at, thus local updates within an AS doesn't affect global GNRS.

Update traffic overhead

When a content is cached at an edge node, it updates the GNRS, which incurs certain amount of update traffic. We have a quantitative analysis of such overhead with a focus on the global level update traffic here.

Since edge caching is used in our architecture, a content request triggers at most one GNRS update, we thus assume there is 200 global updates per AS every second and each GNRS update event contains 25 bytes traffic. Since the Internet consists of roughly 25000 ASes [68], the total Internet scale GNRS update traffic would be $25000 * 25 * 8 * 200 = 1Gb/s$. The overall Internet traffic is $88.7EB$ per month ($\approx 2.74 * 10^5 Gb/s$) in 2016 [22]. Therefore, the GNRS update traffic only takes up a minute fraction of the overall Internet traffic: $(1Gb/s)/(2.74 * 10^5 Gb/s) = 3.65 * 10^{-6}$.

5.5.2 Effectiveness

We evaluate the effectiveness of our scheme in approximating nearest replica routing by investigating cache performance of EDGE-MF compared with EDGE-NRR at a 6-level tree topology, where the root node of the tree is used as both the original content server and the GNRS server. We first look at the performance at different cache budget and Zipf request distribution parameter, and then we vary the redundant probability of the tree topology and evaluate corresponding performance. The data distance and network capacity results are shown in Figure 5.6. The results validate that MF can approximate NRR quite well. The average difference of data distance and network capacity is 0.0038ms and 0.0067Gbps, respectively.

One might concern that our scheme causes certain amount of initial delay since the network needs to query the GNRS for name resolution when receiving a request. However, it is data transmission that consumes most of the time of content delivery, and determines the throughput. In addition, the resolution delay only affects the first chunk of a content, which would lead to the discovery of the path to the closest content copy; subsequent chunks of the same content could simply follow the path. Finally, this delay can be significantly reduced when deploying multiple replica name servers in GNRS. In one manifestation of a global GNRS design [74], the median round trip GNRS query latency is limited at around 40 ms; on the other hand, an Internet content could takes up much longer time, e.g. a YouTube video is normally several minutes long ($\sim$ 4.0 minutes in average [23]), indicating that GNRS resolution takes up a tiny fraction of the overall procedure.

5.6 Related work

5.6.1 Pervasive caching and edge caching

While researchers [52, 64] advocate that making caching pervasive in the network would improve performance, lots of disputes exist since the early days of Internet. In 1999, authors in [77] found that hierarchical or cooperative caching wasnt helpful when population behind edge cache grew past a small threshold, which applied to both realistic traces and heavy-tailed object distributions. More recently, work in [29] shows that the maximum performance gain of a full-fledged ICN architecture is 9% over simple edge-caching. The reason why additional caching doesnt help lies in the content request pattern: a reasonable sized set of popular content and a very long tail of unpopular content. Moderately sized edge caches are easily sufficient to handle the popular content. Once one enters the long tail, the effectiveness of caching decreases exponentially [10].

Our work again confirms such argument. Compared with work in [29], we also find out that nearest replica routing could help optimize content delivery and propose an ICN architecture with nearest replica routing capability. Moreover, compared with the evaluation in [64] which is restricted to single domain network topology with fixed cache budget ratio and Zipf distribution parameter, we investigate much broader network settings with different network topology, cache size, and request distribution.

5.6.2 Separating control plane from data plane in ICN

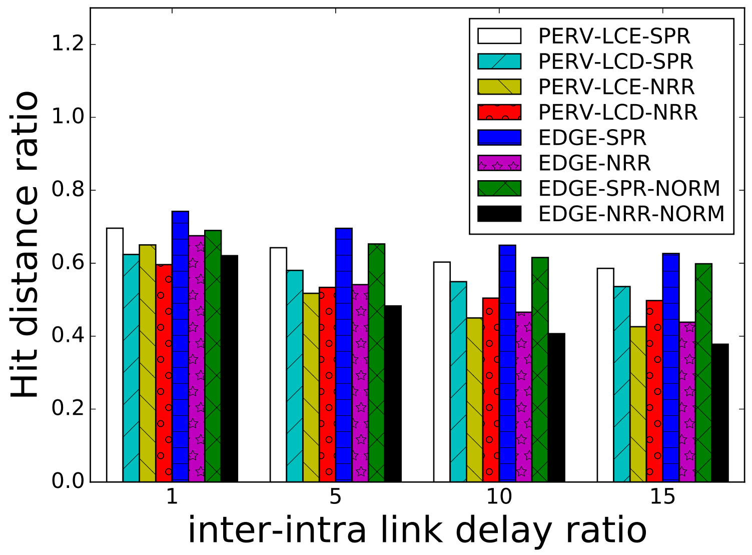
ICN architectures are generally centered around name-based service. The request forwarding process, i.e., resolution from content name to a content location, determines how such service is realized. Based on this, current ICN proposals fall into the following two categories: dynamic name resolution and name-based routing.

The name resolution approach (e.g., MobilityFirst architecture [49]) utilizes a global distributed network service as separate control plane to resolve a destination's name to its current address. The name service [67, 74] manages object name to network address mappings and provides fast responses to mapping requests and changes, such as inserts, lookups, and updates, where both end-hosts and routers can query. On the other hand, a name-based routing approach (e.g. NDN [52]) includes object name into the router's forwarding table and forwards packets directly based on the name, without using network-level addresses at all.

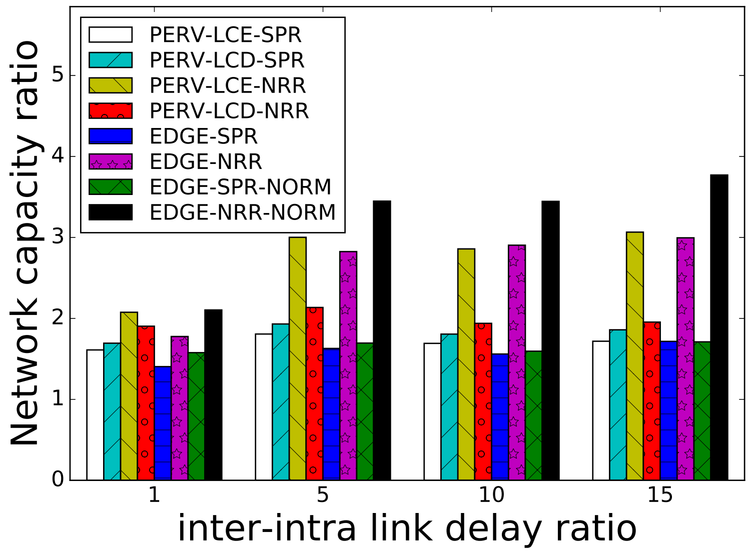
We think that having a separate control plane (e.g. name resolution service) from routing plane is beneficial in ICN [82]. First, a separate control plane limits the size of the forwarding table effectively. Second, it is better suited for mobile scenarios. With a separate control plane, the movement of a client or content simply requires an update to the control plane; however, in named-based routing, the network's routing plane needs to be updated accordingly, and the propagation of routing updates would cause substantial overhead and delays for a large network. Lastly, having a separate control plane makes nearest replica routing easy to realize, while under name-based routing approach like NDN, advertising cache information into the routing plane becomes impossible because of potential routing table explosion, thus cache copies can only be opportunistically explored.

5.7 Conclusion

In this paper, we analyze the design space for ICN content caching and request forwarding scheme, and compare alternative design options that adopt different combination of solutions in the space. The simulation results reveal that pervasive caching might not bring significant performance gain as ICN normally claims while having nearest replica routing capability could bring even more benefits. We thus propose a scheme based upon MobilityFirst architecture that adopts edge caching and approximates nearest replica routing (NRR) through a global name resolution service. Detailed analysis and simulation results show that our scheme can approximate ideal NRR quite well and is feasible to deploy at a global Internet scale.

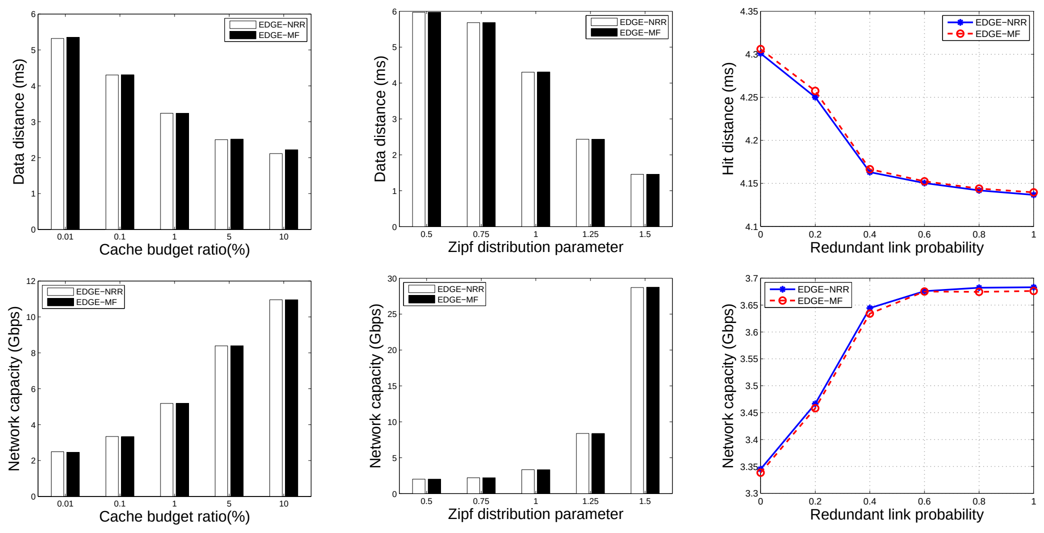


(a) data distance ratio



(b) network capacity ratio

Figure 5.5: Cache performance at different inter link to intra link delay ratio for the large-scale core-tree topology. The cache budget ratio and Zipf distribution parameter is 0.1% and 1. EDGE-NRR-NORM performs the best among all designs.



(a) cache to catalog size ratio

(b) content popularity parameter

(c) redundant link probability

Figure 5.6: The effectiveness of our edge cached MF scheme compared with ideal EDGE-NRR at different cache budget ratio, Zipf distribution parameter, and redundant link probability, with default value as 0.1%, 1, and 0, respectively. The results show that our scheme performs very close to ideal EDGE-NRR.

Chapter 6

System Prototype and Field Trial of the MobilityFirst Future Internet Architecture

In this chapter, we talk about the prototype and field trial effort for the MobilityFirst architecture. We have implemented the whole MobilityFirst architecture and developed a video distribution system using the prototype. The system utilizes edge cache to improve content delivery, and is integrated with satellite broadcast to facilitate large scale video distribution. We deploy the whole system on ORBIT testbed and validates its performance.

6.1 MobilityFirst Prototype Overview

We have implemented a proof-of-concept prototype of the MobilityFirst (MF) architecture. The prototype includes a Click-based router [42], the distributed name resolution service (GNRS), and the host-side protocol stack and network API.

The router is implemented as a set of routing and forwarding elements within the Click modular router. It implements reliable hop-by-hop data transport in which each data chunk is transferred reliably at each hop, and has the interface to GNRS for dynamic name resolution. It's very easy to extend the software implementation with additional logic modules to support programmable network services. For example, content caching is implemented as a plug-in module to the MF router to support inherent in-network content caching.

The name resolution service (GNRS) is implemented as a DHT based distributed key-value store based on the DMAP scheme [74] to manage the mappings between GUID and its location(s). It is implemented in Java which is hardware and OS independent.

The implementation uses Apache MINA as the network socket library to develop high performance network application, and utilizes BerkeleyDB to provide both in-memory and on-disk storage for GUID bindings.

The host protocol stack is implemented on both the linux and Android platform as a user-level event-based data processing engine. It uses the pcap library to intercept and inject packets and interacts with MobilityFirst routers over Ethernet, WiFi, and WiMAX links. The network API is implemented as a user-library, which provides GUID-based *send* and *receive* messaging interfaces, and *publish* and *get* interfaces to publish, locate and retrieve content. These APIs can set the service ID (e.g., multicast routing, anycast routing, real-time streaming, etc) in the packet header, which would determine how the routers process and forward the packet. These network APIs are used to develop MobilityFirst apps. A MF app communicates with local host stack through UNIX domain sockets, and is then connected to the MF networks.

6.2 Video Distribution System over Satellite

Video streaming has become the dominant form of traffic consumed over the Internet. The ever-increasing demand of video consumption has resulted in heightened levels of congestion within today's terrestrial networks. The inherent unicast characteristics of terrestrial networks imply that multiple point-to-point sessions need to be created in order to transfer a content to multiple different sites. This will become challenging as the number of users and the size of the contents increase. As a broadcast medium, satellite networking [11] has its own distinct advantages over terrestrial networks in distributing delay tolerant, high-bandwidth content over a large geographical area simultaneously, which comes as a rescue. In addition, edge caching is an effective approach to offload content request from original content provider. Therefore, we develop a video distribution system over satellite for the field trial of MobilityFirst, in which the content provider utilizes satellite overlay to efficiently distribute popular contents to the edge cache service and significantly alleviates network congestion. Our system combines the benefits of both networks: the advantage of terrestrial network in handling individual request instantaneously, and the capability of satellite in serving multiple,

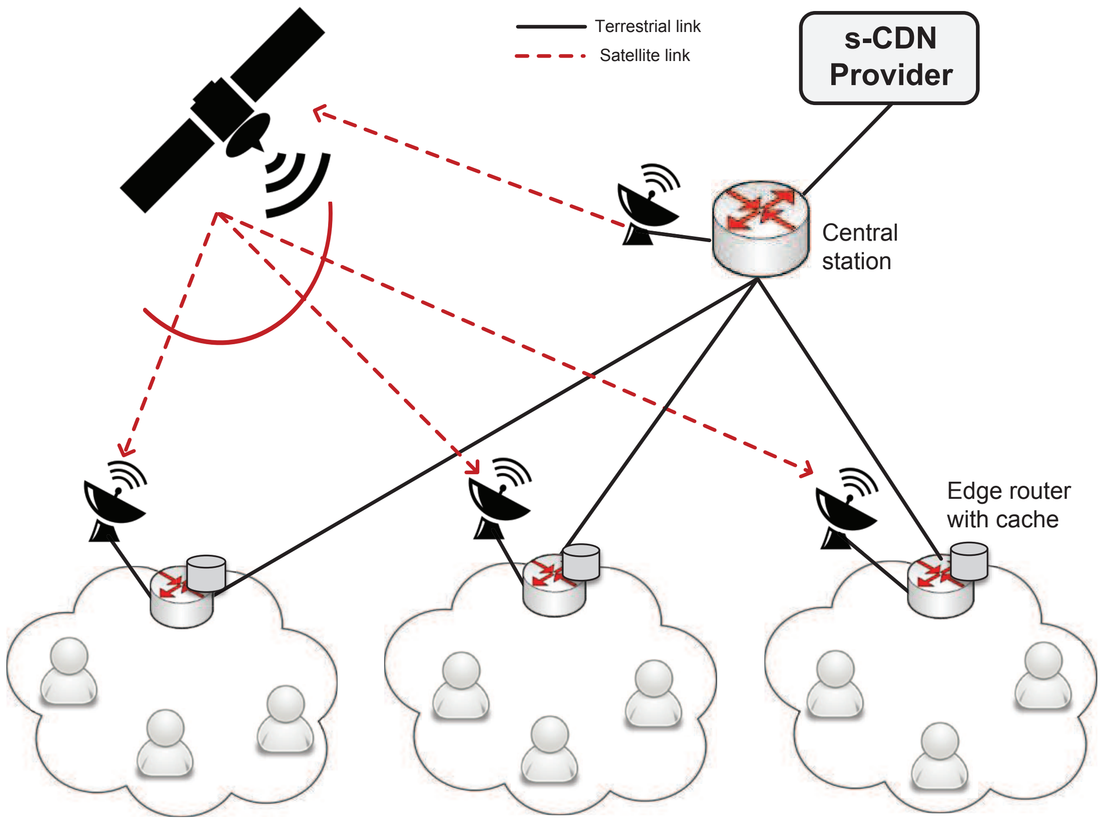


Figure 6.1: Video distribution system with satellite overlay. The central station is connected to the satellite CDN (s-CDN) provider, and has both satellite medium and terrestrial medium to deliver content.

delay-insensitive requests simultaneously.

Figure 6.1 depicts the overall architecture of the satellite overlay to video distribution networks from a high level. The satellite CDN (s-CDN) provider is conceptually the content provider, e.g., the s-CDN provider has agreement with Netflix and Youtube for serving their content. The central station is connected to the s-CDN provider and is able to distribute the content through either satellite network or terrestrial network. Each edge network has an injection point of satellite at the edge router. The edge router is also equipped with cache storage to serve requests locally.

The central station serves as the coordinator between s-CDN provider and the rest of the content distribution network. It receives data from s-CDN provider and determines how to transmit it. The s-CDN provider receives content requests that were

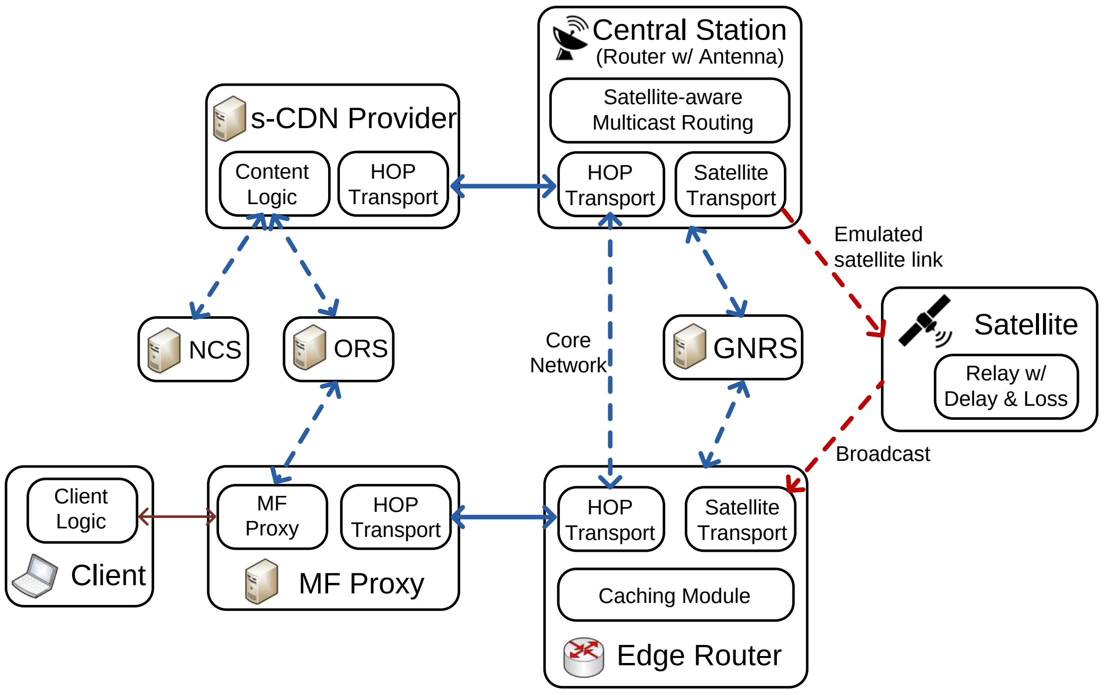


Figure 6.2: Prototype of the video distribution system using MobilityFirst. Reliable transport is the hop-by-hop chunk transfer protocol in MobilityFirst, while unreliable transport is specifically designed for the satellite medium.

missed at the edge caches and forwards the content to central station. In addition, s-CDN provider could pro-actively push content to edge caches, thus actively distributes popular contents in advance.

6.3 Core Components and System Prototype

We implement the video distribution system based on MobilityFirst prototype, as shown in Figure 6.2, and conduct the field trial on ORBIT testbed [57]. The additional components added to MobilityFirst prototype includes: content naming service that generates content GUID and provides search function over the content catalog, multicast-aware routing which determines whether data is transmitted through satellite or terrestrial network, satellite transport protocol that is adapted to satellite medium, cache service that brings content closer to clients, etc. We talk about the design and prototype of these core components in detail here.

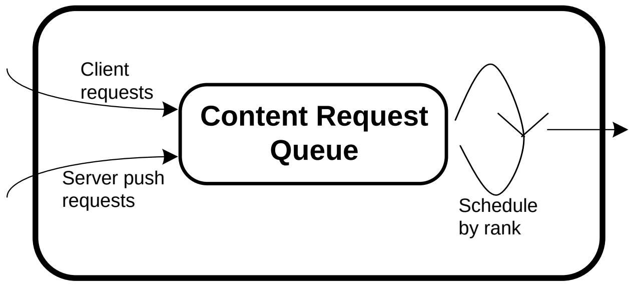


Figure 6.3: s-CDN request scheduling engine. The content request with highest rank is scheduled to be processed next.

6.3.1 Content naming service

The naming service provides the functionality of GUID generation and search which includes the name certification service (NCS) and the object resolution service (ORS). NCS, analogical to ICANN, is a domain-specific naming authority which assigns GUID from human-readable name. For example, content publishers such as Netflix and YouTube can maintain their own NCS that contains a directory of available GUID space, and assigns a GUID when publisher/provider requests with content's metadata. ORS provides the search functionality over the content catalog: when given the keywords, ORS returns relevant content GUID(s) with metadata snippet, and the user could then pick the desired content from returned results. In our satellite scenario, s-CDN provider requests NCS to generate GUIDs for its contents, and publishes these contents with their GUIDs to ORS; ORS indexes all content metadata information and serves search queries over its catelog from users (e.g., MF proxy). We use Apache Lucene as the search engine library to implement ORS.

6.3.2 s-CDN provider and client

In our video distribution system, s-CDN provider provides the video contents, while the client is the final consumer. In addition to serve content for client requests, s-CDN provider could push popular content to cache service (i.e., a group of edge cache routers) based on its content placement logic so as to pro-actively distribute content to the edge. One example placement logic is to periodically pre-position content based on estimated demand of each video [6]. The s-CDN provider thus has an aggregated queue of requests including client request and server push request, and schedules to process each request based on its rank, as shown in Figure 6.3. The rank of request for content i is defined as follow:

$$R_i = w_i * P_i / S_i \tag{6.1}$$

where P_i and S_i is the estimated popularity and the size of the content, and w_i is the urgency factor of the request. When a content is finished transmitting, the next request to be processed is the request with the highest rank in the request queue. The scheduler favors request which is more urgent and the corresponding content is smaller and more popular.

Before s-CDN provider sends out the content, it sets the destination GUID and service ID of the data packet in its packet header. For a client request, the destination GUID is simply the client GUID; for server push request, the destination GUID is the multicast group GUID of the edge cache routers where the server decides to pre-position the content. In addition, s-CDN provider could set the service ID of delay-tolerant, high bandwidth content as satellite service.

In our prototype, we use Dynamic Adaptive Streaming over HTTP (DASH) [72] as the standard for video streaming. In DASH, a complete video content is broken into a sequence of small HTTP-based file segments, each segment containing a short interval of video playback time. A media presentation description (MPD) is used to describe segment information and its URL. To utilize DASH for MobilityFirst network, we assign a GUID for the MPD file and each video segment, and the segment URL in MPD is replaced by its corresponding GUID.

We use the MobilityFirst host stack and API to implement the modified DASH system. A basic DASH video server is implemented using MF network API, which handles requests for content (i.e., video segments and meta files) addressed by their GUIDs. We use VLC player with the DASH plugin [51] as the video client. In order to make the VLC player work with the MF network, a MF proxy is built that translates

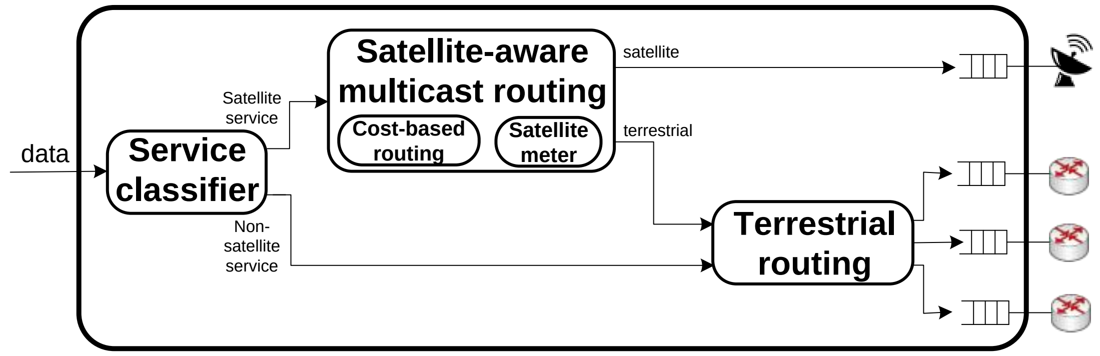


Figure 6.4: Data processing flow at the central station. Satellite-aware multicast routing is used to determine whether satellite medium or terrestrial medium is used for forwarding.

HTTP requests from VLC player to MF content requests, and vice versa. Mappings from video segments URLs to GUIDs is supported through the content naming service - ORS.

6.3.3 Satellite-aware multicast routing

When the central station receives data identified for satellite service, it needs to determine whether it would use the satellite medium or the terrestrial medium to transfer the data, as shown in Figure 6.4.

We propose a cost-based routing logic for the satellite-aware multicast routing, i.e., choosing the medium with smaller cost. Assume that the cost to transmit 1 byte over satellite broadcast medium is c_s, and the cost to transmit 1 byte to edge network i along terrestrial medium is c_{t_i}. For a content chunk with size S that needs to be forwarded to a set of edge networks N, the cost to transmit the data over satellite is $C_s = S * c_s$, while the cost to transmit over terrestrial is $C_t = S * \sum_{i \in N} c_{t_i}$. The satellite-aware multicast routing thus decides on the appropriate transmitting medium by comparing C_s and C_t. If satellite medium is picked, a further check on satellite meter is conducted to avoid overusing the satellite network and congesting it, since the bandwidth of satellite is constrained. The satellite meter is implemented as a token bucket, in which token is added at a rate equal to satellite transmission bandwidth. If the meter has enough tokens for a chunk with size S, it removes S tokens and passes the data to satellite

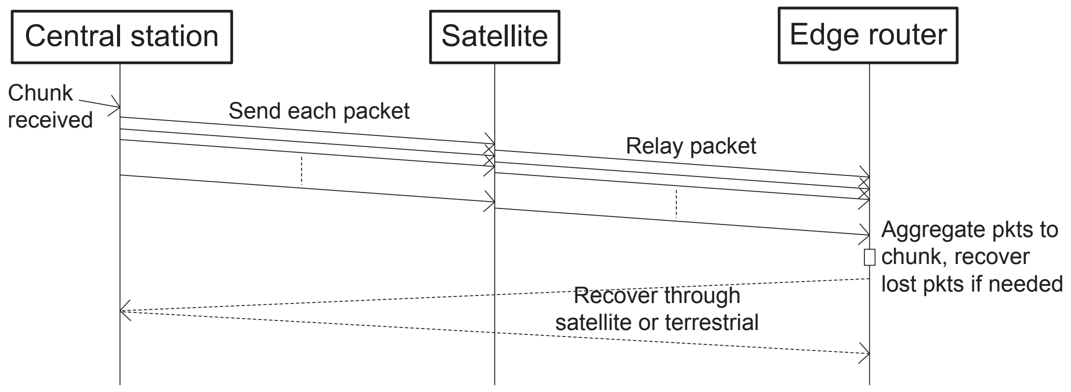

Central station
Satellite
Edge router
Chunk received
Send each packet
Relay packet
Aggregate pkts to chunk, recover lost pkts if needed
Recover through satellite or terrestrial

Figure 6.5: Sequence diagram of the satellite transport protocol.

interface; otherwise, the data is still sent through terrestrial.

6.3.4 Satellite transport protocol

Satellite link is substantially different from both the terrestrial and traditional wireless link [3]: it is asymmetric, error-prone (1 error per 10 million bits), and has high delay ($10ms{\sim}300ms$ depending on satellite altitude) and limited bandwidth ($100Kbps{\sim}20Mbps$). The hop-by-hop reliable chunk transfer in MF (Section 2.2) doesn't work for satellite medium. We thus propose a UDP-style transport protocol for the satellite.

Figure 6.5 illustrates the sequence diagram of the satellite transport. When the central station receives a chunk and decides to transmit it through satellite, it simply sends out all packets of the chunk without asking for acknowledgement from the satellite node, and the satellite broadcasts these packets to each edge router. The edge router then tries to aggregate these packets into the original chunk, and sends out a special recovery request with lost packets information to upstream central station through terrestrial link. After receiving all recovery requests from edge routers, the central station retransmits lost packets.

6.3.5 Edge cache

The edge router has built-in cache capability to serve content requests locally. Since s-CDN provider keeps track of the aggregate requests across the entire coverage area of the CDN, it has a network-wide estimate of relative popularity of each content, i.e., global content popularity. The s-CDN provider periodically updates edge cache routers with these values so that they have up-to-date, global view of content popularity. In addition, the edge cache router monitors content requests in its local network, and could estimate the local popularity of each content. We thus propose a hybrid-popularity based caching scheme in which both local and global popularity is considered for cache replacement at an edge cache router. Note that both the local and global popularity is either measured over some reasonable window or filtered through an aging function, to factor in temporal shifts in popularity.

When an edge cache router receives a new content, it needs to decide whether to stream the content and whether to cache it. If there are outstanding requests from one or more clients for this content in its local network, the edge router will stream the content to corresponding clients. The router also keeps track of the content requests information for local content popularity estimates. In addition, the router will cache the content if there is enough remaining cache space. Otherwise, the router needs to determine whether cache replacement should take place. A hybrid-popularity metric [11] is used for such purpose which weighs both the local and global content popularity.

For content i, denote the global popularity as G_i, and the local popularity at edge cache router m as $L_{m,i}$. Let $w \in [0, 1]$ be the global weighting factor, and M be the total number of cache routers. The hybrid popularity metric $P_{m,i}$ is defined as follows:

$$P_{m,i} = (1 - w) * L_{m,i} + w * \frac{G_i - G_{m,i}}{M - 1} \tag{6.2}$$

In the equation, the term on the left is scaled local popularity, and the term on the right is a measure of average requests from edge networks other than the current one, scaled by w. The hybrid-popularity metric captures the spatial locality of content requests, and offsets the small request sample size in local estimate by combining global estimate which could otherwise be inaccurate.

The cache service is implemented as an external service that communicates with the MobilityFirst Click-based router through network socket as shown in Figure 6.6. Such design provides certain level of isolation between the cache service and MF router,

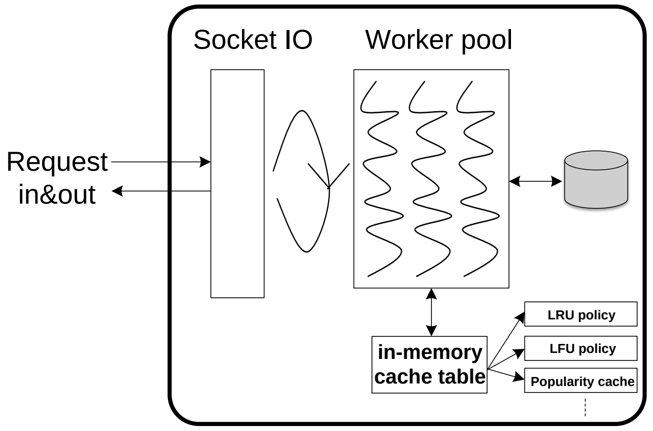


Figure 6.6: Cache service prototype in MobilityFirst.

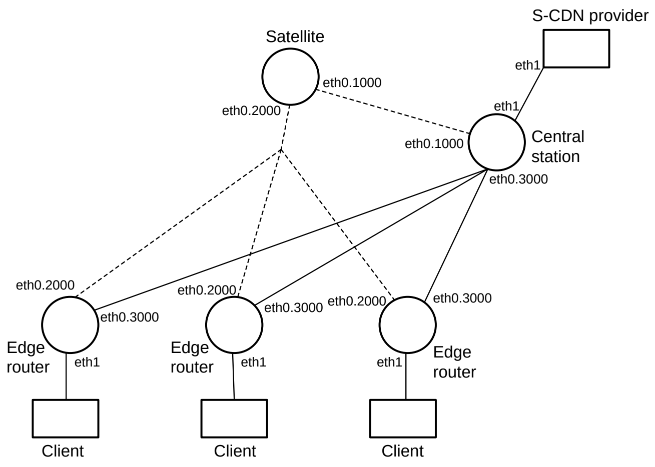


Figure 6.7: Field trial experiment scenario on ORBIT testbed.

ensures that IO-heavy cache service doesn't impact the core performance of the router logic, and makes cache service more flexible and scalable. An abstract base class of cache operations is defined which includes $try\_cache(Data)$, $check\_cache(request)$, and $get\_cache(request)$, and is very easy to be extended with different caching policy, e.g., LRU caching policy, LFU caching policy, popularity-based caching, etc.

6.4 Field Trial Experiment and Future Work

We use the ORBIT testbed for our field trial experiment, with each node running Ubuntu 12.04 and Linux kernel version 3.2. In terms of networking capability, each node is equipped with two Gigabit-Ethernet interface, eth0 and eth1. Both interfaces on all the ORBIT nodes are connected, each forming a Gigabit LAN. We use VLAN tags to create desired topology to isolate satellite traffic from terrestrial traffic, and the link emulation tool netem [54] to configure the link bandwidth and latency.

We conduct a full emulation of the satellite-based video distribution system on ORBIT under a relatively small topology as seen in Figure 6.7, and validate the capability of MobilityFirst architecture in tackling real-world use case. Future work of the field trial includes system emulation on national wide network - GENI [34] with larger topology, and detailed system evaluation and performance analysis.

Chapter 7

Conclusions

The evolution of 5G [28, 39] promises to provide richer content and service access for mobile users, including HD video, virtual reality, M2M communication, etc. The trend, on the one hand, is leading to the convergence of cellular network and the Internet since both industries are serving the same mobile end-users and providing the same content and service; on the other hand, it is shifting the network towards a platform for information-centric usage with mobility as the norm. Information-centric network (ICN) is aimed to address such new challenges from clean-slate network architecture perspective. The dissertation investigates content delivery schemes under two ICN architectures, CCN and MobilityFirst. In particular, we make the following contributions:

- We analyze the characteristics of content delivery in CCN: it is receiver-driven with hop-by-hop transport fashion; it has no explicit content source resulting in multi-source, multi-path transfer. We propose a transport protocol, CHoPCoP/pCHoPCoP, that utilizes explicit congestion control to deal with congestion in such multi-source, multi-path transfer scenario. Our protocol also supports multi-homing at the client by distributing Interests across different network interfaces and aggregating their bandwidth.

- We propose EdgeBuffer framework under MobilityFirst context that moves beyond traditional caching. The framework advocates pushing caching towards the edge of the Internet, and separates the edge storage into cache buffer and prefetch buffer for popularity-based content caching and mobility prediction-based prefetching, thus capturing both long-term aggregated content access pattern and short-term individual user access pattern. In addition, we propose a network-level mobility prediction model to guide prefetching. We show that the framework is a promising approach in improving cache hit ratio and supporting mobile content delivery.

- We take a step back for caching, and investigates the pros and cons of different caching and request forwarding schemes in ICN. In particular, we compare pervasive caching with edge caching, and nearest replica routing with shortest path routing. Our simulation results show that pervasive caching is not fundamentally better than edge caching; compared to pervasive caching, nearest replica routing brings more benefits, especially in a large network. Therefore, we propose a caching system based on the MobilityFirst architecture that adopts edge caching and nearest replica routing.
- We implement a proof-of-concept prototype of the whole MobilityFirst architecture, and develop a video distribution system running on ORBIT testbed based on the prototype. In addition, the developed system utilizes satellite broadcast to efficiently distribute content to large area, and uses edge caching to effectively offload content requests. The prototype and field trial efforts validate the feasibility of applying ICN concept into practical use case.

Although the dissertation explores some crucial components of mobile content delivery (in particular, transport control and content caching) and deepens our understanding of the field, it poses more questions than it answers. Hopefully, the trend towards information-centric network will continue to emerge, and we could see its concept and architectural model applied to the real world in the near future.

References

[1] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone. Design considerations for a network of information. In *ACM CoNEXT*, 2008.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.

[3] M. Allman, D. R. Glover, and L. A. Sanchez. Enhancing tcp over satellite channels using standard mechanisms. *RFC 2488, January*, 1999.

[4] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (aip). In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 339–350. ACM, 2008.

[5] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *ACM SIGCOMM*, 2004.

[6] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan. Optimal content placement for a large-scale vod system. In *Proceedings of the 6th International COnference*, page 4. ACM, 2010.

[7] D. Beaver, S. Kumar, H. C. Li, J. Sobel, P. Vajgel, et al. Finding a needle in haystack: Facebook's photo storage. In *OSDI*, volume 10, pages 1–8, 2010.

[8] C. Bernardini, T. Silverston, and O. Festor. Mpc: Popularity-based caching strategy for content centric networks. In *ICC*. IEEE, 2013.

[9] G. D. Bhanage, Y. Zhang, and I. Seskar. On topology creation for an indoor wireless grid. In *ACM WiNTECH*, 2008.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *IEEE INFOCOM*, Mar 1999.

[11] C. Brinton, E. Aryafar, S. Corda, S. Russo, R. Reinoso, and M. Chiang. An intelligent satellite multicast and caching overlay for cdns to improve performance in video applications. In *31st AIAA International Communications Satellite Systems Conference*, page 5664, 2013.

[12] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. Rofl: routing on flat labels. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 363–374. ACM, 2006.

[13] G. Carofiglio, M. Gallo, and L. Muscariello. Icp: Design and evaluation of an interest control protocol for content-centric networking. In *IEEE INFOCOM WKSHPS*, 2012.

[14] G. Carofiglio, M. Gallo, and L. Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *ACM ICN*, 2012.

[15] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini. Multipath congestion control in content-centric networks. In *IEEE INFOCOM WKSHPS*, 2013.

[16] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *IEEE ICNP*, 2013.

[17] CCNx open source project. `https://www.ccnx.org/`.

[18] W. K. Chai, D. He, I. Psaras, and G. Pavlou. Cache less for more in information-centric networks. In *NETWORKING 2012*, pages 27–40. Springer, 2012.

[19] A. Chandra, J. Weissman, and B. Heintz. Decentralized edge clouds. *Internet Computing, IEEE*, 17(5):70–73, 2013.

[20] Y. Chen, B. Liu, Y. Chen, A. Li, X. Yang, and J. Bi. Packetcloud: an open platform for elastic in-network services. In *Proc. of ACM MobiArch*, pages 17–22. ACM, 2013.

[21] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 1989.

[22] Cisco Systems Inc. White paper: Cisco vni forecast and methodology, 2015-2020.

[23] comScore Inc. comscore releases february 2014 u.s. online video rankings.

[24] S. K. Dandapat, S. Pradhan, N. Ganguly, and R. Roy Choudhury. Sprinkler: distributed content storage for just-in-time streaming. In *Proc. of CellNet*, pages 19–24. ACM, 2013.

[25] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das. Predictive methods for improved vehicular wifi access. In *ACM MobiSys*, pages 263–276, New York, NY, USA, 2009.

[26] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *Internet Computing, IEEE*, 6(5):50–58, Sep 2002.

[27] D. Duchamp et al. Prefetching hyperlinks. In *USENIX Symposium on Internet Technologies and Systems*, pages 12–23, 1999.

[28] Ericsson. 5g radio access-reasearch and vision. *White Paper*, 2013.

[29] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable icn. In *ACM SIGCOMM*, pages 147–158, 2013.

[30] S. Floyd. Tcp and explicit congestion notification. *ACM SIGCOMM Computer Communication Review*, 1994.

[31] S. Floyd. Recommendation on using the "gentle_" variant of red. 2000.

[32] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1993.

[33] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[34] Global environment for networking innovations (GENI). `http://www.geni.net`.

[35] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *USITS*, volume 1, pages 4–4, 2001.

[36] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, et al. Xia: Efficient support for evolvable internetworking. In *USENIX NSDI*, 2012.

[37] H.-Y. Hsieh, K.-H. Kim, Y. Zhu, and R. Sivakumar. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 1–15. ACM, 2003.

[38] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1-2):99–114, 2005.

[39] Huawei. 5g: A technology vision. *White Paper*, 2013.

[40] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[41] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *IEEE INFOCOM*, 2006.

[42] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, August 2000.

[43] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM*, 2007.

[44] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *ACM SIGMETRICS*, 2005.

[45] N. Laoutaris, H. Che, and I. Stavrakakis. The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634, 2006.

[46] M. Li, D. Agrawal, D. Ganesan, A. Venkataramani, and H. Agrawal. Block-switched networks: A new paradigm for wireless transport. In *NSDI*, volume 9, pages 423–436, 2009.

[47] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Network Protocols, 2001. Ninth International Conference on*, pages 165–171. IEEE, 2001.

[48] E. P. Markatos, C. E. Chronaki, et al. A top-10 approach to prefetching on the web. In *Proceedings of INET*, volume 98, pages 276–290, 1998.

[49] MobilityFirst project. `http://mobilityfirst.winlab.rutgers.edu/`.

[50] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol. *RFC 5201, April*, 2008.

[51] C. Müller and C. Timmerer. A vlc media player plugin enabling dynamic adaptive streaming over http. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 723–726. ACM, 2011.

[52] Named data networking project. `http://www.named-data.net/`.

[53] S. C. Nelson, G. Bhanage, and D. Raychaudhuri. Gstar: generalized storage-aware routing for mobilityfirst in the future mobile internet. In *Proceedings of the sixth international workshop on MobiArch*, pages 19–24. ACM, 2011.

[54] netem: network emulation tool. `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`.

[55] A. J. Nicholson and B. D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *ACM MobiCom*, 2008.

[56] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[57] ORBIT testbed. `http://www.orbit-lab.org/`.

[58] S. Oueslati, J. Roberts, and N. Sbihi. Flow-aware traffic control for a content-centric network. In *IEEE INFOCOM*, 2012.

[59] C. Perkins. IP mobility support for ipv4, revised. *IETF Internet Standard, RFC 5944*, Nov 2010.

[60] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/, Feb. 2009.

[61] I. Psaras and V. Tsaoussidis. Why tcp timers (still) don't work well. *Computer Networks*, 2007.

[62] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012.

[63] D. Rossi and G. Rossini. Caching performance of content centric networks under multi-path routing (and more). *Relatório técnico, Telecom ParisTech*, 2011.

[64] G. Rossini and D. Rossi. Coupling caching and forwarding: Benefits, analysis, and implementation. In *Proceedings of the 1st international conference on Information-centric networking*, pages 127–136. ACM, 2014.

[65] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-layer issues in information centric networks. In *ACM ICN*, 2012.

[66] J. Saltzer. On the naming and binding of network destinations. *RFC 1498, August*, 1993.

[67] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav. A global name service for a highly mobile internetwork. Technical report, 2014.

[68] Y. Shavitt and E. Shir. Dimes - letting the internet measure itself. `http://www.netdimes.ogr/`.

[69] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. In *ACM SIGCOMM*, 1995.

[70] L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive wi-fi mobility data. In *IEEE INFOCOM*, 2004.

[71] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 133–145. ACM, 2002.

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

[73] A. Venkataramani, A. Sharma, X. Tie, H. Uppal, D. Westbrook, J. Kurose, and D. Raychaudhuri. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *IEEE COMSNETS*, 2013.

[74] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. In *IEEE ICDCS*, 2012.

[75] WiGLE - Wireless Geographic Logging Engine. `https://wigle.net/`.

[76] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *USENIX NSDI*, 2011.

[77] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. In *ACM SIGOPS Operating Systems Review*, volume 33, pages 16–31. ACM, 1999.

[78] J. Xu, J. Liu, B. Li, and X. Jia. Caching and prefetching for web content distribution. *Computing in science & engineering*, 6(4), 2004.

[79] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Elsevier Computer Communications*, 2013.

[80] J. Yoon, B. D. Noble, M. Liu, and M. Kim. Building realistic mobility models from coarse-grained traces. In *ACM MobiSys*, 2006.

[81] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *ACM SIGSPATIAL GIS*, pages 99–108, 2010.

[82] F. Zhang. Comparing alternative approaches for mobile content delivery in information-centric networking. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–2. IEEE, 2015.

[83] F. Zhang, K. Nagaraja, Y. Zhang, and D. Raychaudhuri. Content delivery in the mobilityfirst future internet architecture. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pages 1–5. IEEE, 2012.

[84] F. Zhang, C. Xu, Y. Zhang, K. Ramakrishnan, S. Mukherjee, R. Yates, and T. Nguyen. Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–9. IEEE, 2015.

[85] F. Zhang, Y. Zhang, and D. Raychaudhuri. Edge caching and nearest replica routing in information-centric networking. In *Sarnoff Symposium (SARNOFF), 2016 37th IEEE*. IEEE, 2016.

[86] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu. A transport protocol for content-centric networking with explicit congestion control. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE, 2014.

[87] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu. Providing explicit congestion control and multi-homing support for content-centric networking transport. *Computer Communications*, 2015.

[88] J. Zhang, R. Izmailov, D. Reininger, and M. Ott. Web caching framework: analytical models and beyond. In *IEEE Workshop on Internet Applications*, 1999.

[89] L. Zhang. Why tcp timers don't work well. In *ACM SIGCOMM*, 1986.

Vita

Feixiong Zhang

Education

- Ph.D., Electrical and Computer Engineering, Rutgers University, 2016
- B.S., Electrical Engineering, Tsinghua University, Beijing, China, 2010

Work Experience

- **Graduate Research Assistant**, WINLAB, Rutgers University, September 2011 to May 2016
- **Software Engineering PhD Intern**, Google Inc, May 2015 to August 2015
- **Research Intern**, Bell Labs, Alcatel-Lucent, June 2013 to August 2013
- **Research Intern**, Corporate *R&D*, InterDigital Inc., May 2011 to August 2011
- **Teaching assistant**, Department of Electrical and Computer Engineering, Rutgers University, September 2010 to May 2011

Publications

- **Feixiong Zhang**, Yanyong Zhang and Dipankar Raychaudhuri. Edge Caching and Nearest Replica Routing in Information-Centric Networking. In *37th IEEE Sarnoff Symposium*, 2016.
- Matteo Varvello, Rafael Laufer, **Feixiong Zhang**, T.V. Lakshman. Multi-Layer Packet Classification with Graphics Processing Units. To appear in IEEE/ACM Transactions on Networking (ToN).
- Yi Hu, **Feixiong Zhang**, K. K. Ramakrishnan, Dipankar Raychaudhuri. GeoTopo: A PoP-level Topology Generator for Evaluation of Future Internet Architectures. In *IEEE ICNP 2015.*
- **Feixiong Zhang**, Yanyong Zhang, Alex Reznik, Hang Liu, Chen Qian and Chenren Xu. Providing Explicit Congestion Control and Multi-Homing Support for Content-Centric Networking Transport. Elservier Computer Communications (COMCOM) 2015.

- **Feixiong Zhang**. Comparing Alternative Approaches for Mobile Content Delivery in Information-Centric Networking. In *IEEE WoWMoM 2015 - PhD Forum.*

- **Feixiong Zhang**, Chenren Xu, Yanyong Zhang, K. K. Ramakrishnan, Shreyasee Mukherjee, Roy Yates, Thu Nguyen. EdgeBuffer: Caching and Prefetching Content at the Edge in the MobilityFirst Future Internet Architecture. In *IEEE WoWMoM 2015.*

- Matteo Varvello, Rafael Laufer, **Feixiong Zhang**, T.V. Lakshman. Multi-Layer Packet Classification with Graphics Processing Units. In *ACM CoNEXT 2014.*

- **Feixiong Zhang**, Yi Hu, Yanyong Zhang, Dipankar Raychaudhuri. Enabling Mobile Content-Oriented Networking in the MobilityFirst Future Internet Architecture. In *ACM MobiHoc 2014.*

- **Feixiong Zhang**, Yanyong Zhang, Alex Reznik, Hang Liu, Chen Qian and Chenren Xu. A Transport Protocol for Content-Centric Networking with Explicit Congestion Control. In *IEEE ICCCN 2014.* (Selected for Elsevier Computer Communications fast track.)

- **Feixiong Zhang**, Alex Reznik, Hang Liu, Chenren Xu, Yanyong Zhang and Ivan Seskar. Using ORBIT for evaluation wireless content-centric network transport. In *ACM WiNTECH 2013.*

- Chenren Xu, Bernhard Firner, Robert S. Moore, Yanyong Zhang, Wade Trappe, Richard Howard, **Feixiong Zhang** and Ning An. SCPL: Indoor Device-Free Multi-Subject Counting and Localization Using Radio Signal Strength. In *ACM IPSN 2013.*

- **Feixiong Zhang**, Kiran Ngaraja, Yanyong Zhang and Dipankar Raychaudhuri. Content Delivery in the MobilityFirst Future Internet Architecture. In *35th IEEE Sarnoff Symposium*, 2012.

- Jian Chen, Hewu Li, **Feixiong Zhang**, and Jianping Wu. Research on switching mechanism for MIMO mode of 802. 11 n wireless networks. Journal of Convergence Information Technology 6, no. 7 ,2011.

- Jian Chen, Hewu Li, **Feixiong Zhang**, and Jianping Wu. MIMO Mode Switching Scheme for Rate Adaptation in 802.11n Wireless Networks. In *IEEE GLOBECOM 2011.*