A Content-Oriented Architecture for Publish/Subscribe Systems

Dissertation

zur Erlangung des Doktorgrades
Dr. rer. nat.
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

im PhD Programme in Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von

Jiachen Chen
aus Shanghai, China

Göttingen
im Januar 2015

Betreuungsausschuss:	Prof. Dr. Xiaoming Fu, Georg-August-Universität Göttingen
	Prof. Dr. Dieter Hogrefe, Georg-August-Universität Göttingen
Prüfungskommission:	
Referent:	Prof. Dr. Xiaoming Fu, Georg-August-Universität Göttingen
Korreferenten:	Prof. Dr. Konrad Rieck, Georg-August-Universität Göttingen
	Prof. Dr. Edith Ngai, Uppsala University, Sweden
Weitere Mitglieder der Prüfungskommission:	Prof. Dr. K.K. Ramakrishnan, University of California, Riverside, USA
	Prof. Dr. Florentin Wörgötter, Georg-August-Universität Göttingen
	Prof. Dr. Carsten Damm, Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 16. März 2015

Abstract

With the user behavior of Internet shifting towards data dissemination and retrieval, *Information Centric Networking* (ICN) is proposed to suit these demands. ICN treats content as the first-class entity, with nodes exchanging information based on the names of the content instead of the IP addresses of the end points requesting or providing the content. This shift from a "location-based" network to a "content-centric" network allows more efficient data dissemination, especially when the content may be available at multiple points, or the provider or consumer is mobile.

Publish/Subscribe (pub/sub) is another important content-centric communication model where publishers and subscribers only focus on the data rather than the location of each other. However, most of the ICN proposals do not have direct support for pub/sub. This inevitably results in pub/sub being implemented via polling which leads to higher network load, higher publisher load and longer latency.

This dissertation tackles the shortcomings of existing solutions and provides a full-fledge solution for efficient pub/sub communication in ICN. By enhancing *Named Data Networking* (NDN) in the network layer, *Content Oriented Network for Publish/Subscribe Systems* (COPSS) is able to support best-effort push-based multicast that is required by pub/sub. A transport-layer protocol (*Control Protocol for Scalable and Adaptive Information Dissemination* (SAID)) is proposed to satisfy the need for reliability and congestion control during pub/sub. The protocol attempts to solve a long-standing issue of multicast congestion control by separating congestion control from reliability. In the application layer, an object resolution system (*Object Resolution Framework in Information-Centric Environment* (ORICE)) similar to a search engine is proposed to help users get identities of the data that they might need.

The efficiency of the architecture is evaluated via applications that emulate Twitter, gaming, VoD and file transfer. Results show that the architecture can outperform existing pull-based ICN solutions in terms of response latency, network load and publisher load.

Additionally, this dissertation extends the concept of content-concentricity to a common and important network management issue – service chaining. The design proves that this concept can be incorporated into many network designs and has the potential to make these systems more efficient and dynamic.

Acknowledgements

I would like to sincerely thank my supervisor Prof. Xiaoming Fu for his constant support, his courtesy to pursue my diverse research interests and the chances he allowed me to take in visiting great research laboratories and attending conferences all over the world. His efforts and guidance made this dissertation possible.

I am deeply grateful to Prof. K.K. Ramakrishnan, who also kindly agreed to be my co-supervisor. K.K. helped to continuously improve my work through constructive criticism and reviews in hours over hours of discussions in numerous meetings.

I would also like to express my gratitude to Prof. Konrad Rieck and Prof. Edith Ngai for being a reviewer of my dissertation, to Prof. Florentin Wörgötter and Prof. Carsten Damm for being a member of my examination committee.

I'm obliged to Prof. Jin Zhao and Prof. Junyu Niu for hosting me during research visits at Fudan University.

I consider it an honor to work with Dr. Mayutan Arumaithurai and Sripriya Srikant Adhatarao on this topic. The designs involved in this dissertation would not have been accomplished without their contribution and hard work.

I am grateful to my former and current colleagues at the Computer Networks Group at the University of Göttingen, especially Dr. Yang Chen, Dr. Lei Jiao, Dr. David Koll, Narisu Tao, Dr. Stephan Sigg and Dr. Xu Chen, whose feedbacks also contributed to the quality of this dissertation.

Last but definitely not least, I want to thank my parents Jianguo Chen and Lingzhen Liao for their never-ending support. Without them this dissertation would not have been written in the first place.

Contents

List of Figures

List of Tables

Acronyms

ACK *Acknowledgement*

AI *Artificial Intelligence*

AIMD *Additive Increase Multiplicative Decrease*

AoI *Area of Interest*

ARPANET *Advanced Research Projects Agency Network*

AS *Autonomous System*

BDP *Bandwidth-Delay Product*

CAPEX *Capital Expenditure*

CCN *Content Centric Networking*

CD *Content Descriptor*

CDF *Cumulative Distribution Function*

CDN *Content Delivery Network*

CI *Confidence Interval*

CNS *Content-centric Notification System*

COPSS *Content Oriented Network for Publish/Subscribe Systems*

CS *Counter-Strike*

C/S *Client/Server*

DPI *Deep Packet Inspection*

DSA *Dynamic Site Accelerator*

DHT *Distributed Hash Table*

DNS *Domain Name System*

FCFS *First Come First Served*

FCSC *Function-Centric Service Chaining*

FIB *Forwarding Information Base*

G-COPSS *Gaming over COPSS*

GFC *Generic Fairness Configuration*

HTTP *Hypertext Transfer Protocol*

ICN *Information Centric Networking*

ID *Identity*

IP *Internet Protocol*

IPC *Inter-Process Communication*

ISP *Internet Service Provider*

LRU *Least Recently Used*

LFU *Least Frequently Used*

MAC *Media Access Control*

MMORPG *Massively Multiplayer Online Role Playing Game*

MPR *Minimum Pending Request Count*

MMPR *Minimum MPR*

NAK *Negative Acknowledgement*

NAT *Network Address Translation*

NDN *Named Data Networking*

NFV *Network Function Virtualization*

ORICE *Object Resolution Framework in Information-Centric Environment*

OSI *Open Systems Interconnection*

P2P *Peer-to-Peer*

PBR *Policy-Based Routing*

PIT *Pending Interest Table*

PR *Pending Request Count*

pub/sub *Publish/Subscribe*

QoS *Quality of Service*

REM *Random Early Marking*

RP *Rendezvous Point*

RSS *Rich Site Summary*

RTT *Round Trip Time*

SAID *Control Protocol for Scalable and Adaptive Information Dissemination*

SDN *Software-Defined Networking*

ST *Subscription Table*

TCP *Transport Control Protocol*

TLV *Type-Length-Value*

TTL *Time To Live*

UDP *User Datagram Protocol*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

VoD *Video on Demand*

Chapter 1

Introduction

1.1 The Problem

Advanced Research Projects Agency Network (ARPANET), the dominant basis of today's Internet, was designed to satisfy the need for resource sharing. Its main purpose was to allow researchers to remotely use scarce devices like high-speed tape drives or even supercomputers. To enable the communication between devices, the network protocol in ARPANET used a pair of source and destination addresses in the packet header. Such *location-based* design was later adopted in the widely used *Internet Protocol* (IP).

However, the use of the network has evolved dramatically in the past several decades. Internet users increasingly desire access to *information*, ranging from news, financial issues, healthcare, to disaster relief and beyond, with disregard to their location. The mismatch between the *information-centric* user requirements and the *location-based* network design leads to wasteful network traffic and longer data retrieval latency (bad user experience).

Information Centric Networking (ICN) proposals [1–5] seek to resolve this mismatch by transforming *content* as a first-class entity. With nodes exchanging information based on the names of the content instead of the IP addresses of the end points requesting or providing the content, the network allows more efficient data dissemination, especially when the content may be available at multiple points, or when the provider or consumer is mobile. Additional performance benefits accrue with the widespread use of in-network caches.

Named Data Networking (NDN) [4], also known as *Content Centric Networking* (CCN) [5], is one of the more popular examples of ICN. NDN provides a substantial degree of flexibility for users and end-systems to request for information without regard to their location or source. By exploiting caches, NDN improves the efficiency of content delivery. Consumers can obtain data from the closest node/cache serving it. Moreover, multiple requests for the same data arriving at an NDN router can be served simultaneously by that router, oblivious to the source of the data. Additionally, the inherent per packet signature from the provider in the design ensures data integrity during dissemination.

Further, users also desire access to information irrespective of *who* published it, and often, *when* it was/will be published. *Publish/Subscribe* (pub/sub) as a very important component of the information-centric communication is particularly suited for large scale dissemination of such information, and provides the temporal separation between information generation and indication of interest. It enables users to subscribe to information of interest, without being intimately tied to who the publisher is and when that information is made available by publishers.

Intelligent end-systems and information aggregators in today's Internet (e.g., Google News, Yahoo! News, cable and satellite providers) have increasingly adapted their inter-

faces to provide such service. However, these mechanisms are built on top of a centralized server-based framework and may result in wastage of network resources as shown in [6, 7], since the IP suite is focused on end-to-end delivery of data. Furthermore, issues of *coverage* and *timeliness* still exist in such forms of dissemination, where the aggregator may be selective in what information is made available. Therefore, enabling the pub/sub capability in the network layer is highly-desired and it has the potential to overcome the aforementioned limitations.

However, most of the existing ICN proposals limit users in a pure query/response communication model which results in difficulties in supporting efficient pub/sub. The difficulties can be described as follows:

1.1.1 What to request for?

In a typical pub/sub environment, the subscribers (data consumers) only have a brief idea about what they are interested in (e.g., football, FIFA, 2014), but they usually have no means to get the exact name(s) of the data as required by ICN (e.g., `/CNN/edition/2014/-07/01/tech/web/usa-soccer-world-cup-fifa.html`).

To achieve the pub/sub functionality, a common bypass in ICN uses an *exclusion* field in the request. The exclusion field indicates what the request does not want to receive. The solutions would ask the data consumers to request for a prefix (e.g., `/CNN/edition`) and carry a list of the received data names in this exclusion field to avoid duplication. To get all the data that has been published, the consumers have to keep sending requests until no entity in the network can respond to the request (usually implemented by observing a timeout). Put aside the growing size of the request (exclusion field) every time a consumer has to send, the computation overhead on the forwarding engines (in processing these requests), and the possible data providers (in handling redundant requests) would be unnecessarily high.

1.1.2 When to request?

According to the definition of pub/sub, there is no way the subscribers know when a new data is available. Therefore, it is difficult for the client to decide when to request for the data while emulating pub/sub via a query/response mechanism.

As a common solution, the subscribers in such networks have to keep requesting for the new data periodically (polling). The frequency of polling is a tradeoff between timeliness and network/server overhead. If the publication frequency follows a uniform distribution,

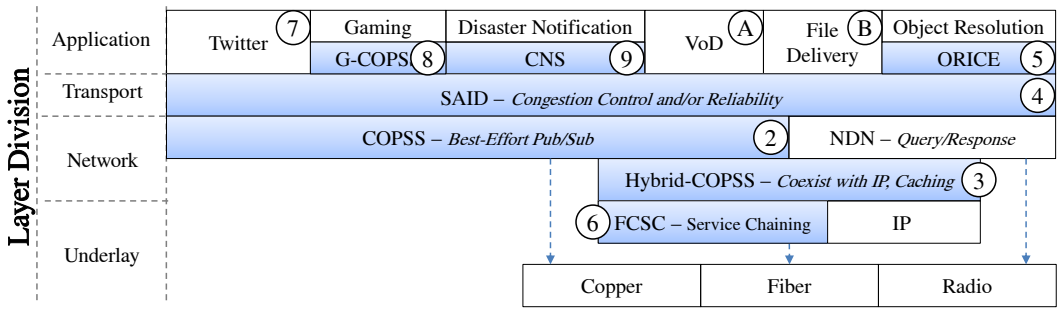


Figure 1.1: Overall Architecture of Dissertation Work.

the average message latency would be half of the polling period while the maximum could be as long as a whole polling period. More frequent polling allows higher timeliness and at the same time incurs higher network and server load. However, the frequency requirements can vary dramatically according to the application requirements – it can be as small as 10's of milliseconds (e.g., interactive games), or as large as minutes (e.g., online social networks) to hours (e.g., Email systems).

Therefore, a direct network-layer support for efficient pub/sub communications is highly desirable especially in the information-centric environment to enable the temporal separation between information generation and indication of interest.

1.2 Dissertation Contributions

This dissertation provides a full-fledge architecture (shown in Fig. 1.1) to support efficient pub/sub communications through the following contributions (following the marking on the figure):

① To provide a clear overview of the state-of-the-art pub/sub approaches, a comprehensive study of existing pub/sub and ICN solutions is conducted. The study reveals the need for a direct pub/sub support from the network layer. The lessons learned from the study are used to guide the design of the proposed architecture.

② Following the study, a basic *Content Oriented Network for Publish/Subscribe Systems* (COPSS) is proposed to enhance NDN with best-effort pub/sub support. This architecture alone can support applications like content-based Twitter, gaming, and notification during disaster (⑦-⑨ in the figure).

③ How to evolve from the existing network architecture is a key problem that every

next-generation network has to face. This work provides a hybrid-COPSS protocol that uses IP as underlay, and a deployment strategy that can help *Internet Service Provider*s (ISPs) to start providing information-centric functions step-by-step.

④ To satisfy the need for reliable publishing, the *Control Protocol for Scalable and Adaptive Information Dissemination* (SAID) is provided in the transport layer to minimize the modifications in the network layer for this extra functionality. The protocol also addresses a long-standing challenge of congestion-control in one-to-many communications. Applications like *Video on Demand* (VoD) and file delivery are used as examples and evaluations (Ⓐ, Ⓑ).

⑤ The ultimate goal of a person using the Internet is to get the data object that (s)he wants. ICN can provide better name resolution compared to IP networks, but an object resolution system is still needed to help users find proper names/*Content Descriptor*s (CDs) to fetch data. This dissertation provides *Object Resolution Framework in Information-Centric Environment* (ORICE), an object resolution framework to provide the full-fledge object retrieval solution.

⑥ This work further extends the concept of COPSS and information-centricity to provide better management in a service-chaining (IP) network. The flexibility, dynamicity, efficiency and reliability of the new solution is shown through a comparison with SDN.

1.2.1 A Comprehensive Review of *Publish/Subscribe* Approaches

As the first contribution of the dissertation, a review of pub/sub approaches are conducted before proposing yet another pub/sub architecture. Several issues are identified in the review:

- **Pull-based implementation:** Many existing pub/sub approaches are built on top of HTTP – a typical pull-based transport protocol. These solutions face timeliness issues and incur higher network and server loads.
- **Agnostic to the network topology:** Some other solutions use application-layer brokers for data dissemination. Although the application layer brokers can perform relatively sophisticated forwarding logic, they are agnostic to the underlay network topology and therefore waste more network traffic, especially inter-domain traffic.
- **Scalability:** With the number of publishers/subscribers increases, these solutions either require servers to maintain more states (e.g., connections) or consume a lot more network traffic due to the deployment of *Network Address Translation* (NAT) and the lack of IP multicast.

- **Flexibility:** Many proposed solutions require a large amount of control messages to synchronize the states on the brokers/servers. When a large number of subscribers join, leave or change subscription, the overhead of the system becomes overwhelming.

1.2.2 A Network-Layer Content-Oriented Publish/Subscribe Protocol

Based on the lessons learned in the review, this dissertation proposes *Content Oriented Network for Publish/Subscribe Systems* (COPSS), a new network-layer pub/sub protocol that addresses the aforementioned issues (② in Fig. 1.1). COPSS enhances NDN – a proposal of ICN that only supports query/response – with pub/sub functionality. The contributions of the protocol include:

- By introducing the notion of *Content Descriptor*s (CDs), COPSS allows publishers to publish data without knowing who and where the subscribers are, and subscribers subscribe to interested topics, authors, etc., with disregard to the location, publisher or the publication time.
- COPSS adds a *Subscription Table* (ST) to the NDN forwarding engine to maintain the subscription at the network layer. ST only needs to perform simple match based on the hierarchically structured CDs so that the solution is feasible in the network layer.
- COPSS adopts a *Rendezvous Point* (RP)-based communication model that enables the push-based communication similar to an IP multicast. This provides support for applications that require tight timeliness and forwarding efficiency.
- To ensure the scalability for supporting large amount of publishers, subscribers and data, COPSS provides *automatic RP balancing mechanism* that eliminates the well-known traffic-concentration issue.
- The *two-step communication* suggested in COPSS takes the subscriber interest and policy control into consideration.
- An *implementation* of COPSS over CCNx (an implementation of NDN [8]) is provided to show the feasibility of the COPSS design.

1.2.2.1 Application: Content-Based Twitter

Twitter is particularly an emblematic of a pub/sub environment. With the help of COPSS, the users can go beyond the subscription from a particular user to any content property (e.g., topic, Tweet time, location, etc.). The application is deployed and simulated using COPSS (⑦ in Fig. 1.1).

To support asynchronous data dissemination (get missing data when users are offline), brokers are provided in the application layer to store Tweets. With the help of the hierarchically structured CDs, the brokers can be distributed based on the CD hierarchy.

The application serves to show the need for a direct pub/sub support in the network. Three possible solutions for the system (NDN polling, COPSS and IP multicast) are compared with metrics of response latency, network load and publisher load. The results show that with push-based multicast support, the application using COPSS consumes least network resource, incur shortest latency and lowest publisher load.

1.2.2.2 Application: Gaming

*Massively Multiplayer Online Role Playing Game*s (MMORPGs) are also applications that use content-oriented pub/sub communication: the players publish updates to an area/object (content) without regard to who is supposed to see it; at the same time, they subscribe to the areas/objects (again, content) they can see, without knowing who else is trying to modify them. Due to the different timeliness and scalability requirements, they are always treated differently in the IP network. Many MMORPGs either face the scalability issue (they have to limit the number of players in each game instance), or face the timeliness issue (players cannot interact with the other players in a timely manner).

With the support from COPSS, MMORPGs can achieve scalability and timeliness at the same time. The hierarchical CD structure also enables a hierarchical map partitioning in games which allows more sophisticated game-world design.

The gaming application (⑧ in Fig. 1.1) is used as a stress test on the performance of COPSS, especially on the capability of automatic RP balancing, which eliminates possible traffic concentration. The results show that COPSS can perform better than the state-of-the-art solutions like NDN and IP-server approaches.

1.2.3 Evolving from Existing Network

As a next-generation network design, it is very important to provide a smooth transition from the existing network architecture. Hybrid-COPSS is proposed as an overlay solution to coexist with IP network (③ in Fig. 1.1). The hybrid solution addresses incremental deployment of ICN and elegantly combines the functionality of content-centric networks with the efficiency of IP-based forwarding including IP multicast. With proper CD to IP multicast address mapping, hybrid-COPSS can achieve all the functionality of COPSS with small amount of wasteful network traffic.

As a more general ICN solution, hybrid-COPSS also proposes an approach for incremental deployment of caches to achieve efficiency and scalability when the ISP has large amount of data providers and consumers. Detailed evaluation on a synthetic topology is used to compare the performance of different cache enabling strategies. The result shows that the top-down model enabled by RP-based query/response can achieve higher cache hit rate and lower request latency with same amount of cache size.

To overcome the lack of inter-domain IP multicast, hybrid-COPSS uses COPSS with shortcuts in the ICN overlay and intra-domain IP multicast in the underlay. The simulation results show that such solution consumes even lower inter-domain traffic compared to a pure COPSS solution.

1.2.4 A Reliable Transport-Layer Congestion Control Protocol

Many pub/sub applications need reliable delivery of publications. Whenever reliability is needed, retransmission and possible congestion collapse will occur as a consequence. As part of the contribution, the dissertation proposes *Control Protocol for Scalable and Adaptive Information Dissemination* (SAID) to provide reliability and congestion control (④ in Fig. 1.1). The protocol can also help normal query/response avoid out-of-sync issue when dealing with heterogeneous data consumers.

SAID solved the long-lasting issue of efficient reliable multicast by decoupling reliability from congestion control. By introducing a new request/subscribe model (get *any next* packet), SAID achieves fairness on each path from the publisher to the subscribers. Then, the subscribers can retrieve the missing packets (due to the lack of available link bandwidth or other issues) either from the publisher or from other subscribers. The ICN ensures the privacy (subscribers do not know each other) and data integrity.

Detailed evaluations via VoD and file delivery (Ⓐ and Ⓑ in Fig. 1.1) shows that SAID can achieve lower session completion time and lower aggregate network load compared to the NDN and reliable IP multicast solutions.

1.2.5 An Application-Layer Object Resolution System

ICNs typically require that users know the names/descriptors of the content before they send requests or subscriptions. This requirement is critical to the forwarding efficiency especially in a pub/sub environment where publishers and subscribers should know the CDs each other might use. *Object Resolution Framework in Information-Centric Environment* (ORICE) (⑤ in Fig. 1.1) is designed to support object resolution systems that meet the requirement.

ORICE chooses to put the object resolution in the application layer. Although in this case, users have to explicitly send requests to services, but such a decision allows object resolution services use advanced functionalities at the same time keep the network simple and efficient. ORICE also introduces a name-certification service that manages the ICN name space. Such design allows easier name management in ICN and also enables multiple object resolution services operate on a same name space.

From another point of view, ORICE also introduced a way for the existing object resolution systems (e.g., search engines, recommendation systems, etc.) to use ICN. With name resolution being pushed to the network layer, object resolution services can get the benefits like information diversity, scalability, data dissemination efficiency, security, etc.

A prototype of object resolution service that uses ORICE is implemented to show the feasibility and efficiency of the architectural design.

1.2.6 Extension: *Function-Centric Service Chaining* (FCSC)

The dissertation extends the concept of information-centricity to a network management requirement – service-chaining – which is currently achieved by SDN. Service-chaining is the steering of flows (usually because of administrative reasons) through the different network functions needed, before it is delivered to the destination. By introducing a *naming layer*, the solution (FCSC, ⑥ in Fig. 1.1) allows the network managers achieve: *1*) flexibility: the required functions can be easily added/removed/modified before the flow exits the network, *2*) dynamicity: the managers can dynamically instantiate/dispose function instances in the network, *3*) scalability: FCSC can support a large number of users/flows and functions, and *4*) reliability: FCSC can adapt to function failure without large amount of packet loss.

A set of simulations using real-world topology shows the benefit of FCSC compared to the SDN solutions in terms of packet latency, loss rate and number of rules (states) stored in the network.

1.3 Dissertation Overview

The remainder of this dissertation is organized as follows: in Chapter 2, a comprehensive review of related work is discussed. The basic network-layer COPSS is introduced in Chapter 3 followed by two applications in Chapter 4 (content-based Twitter) and 5 (gaming, *Gaming over COPSS* (G-COPSS)). Chapter 6 devoted to the incremental deployment of COPSS by coexisting COPSS with IP networks (hybrid-COPSS). Chapter 7 proposes

the transport-layer solution for reliable pub/sub and congestion avoidance (SAID). To provide a full object retrieval functionality, ORICE is proposed in Chapter 9. An extension of information-centricity which allows service network provide efficient and dynamic service chaining (FCSC) is presented in Chapter 8. Chapter 10 summarizes the dissertation.

Chapter 2

Background and Related Work

This chapter covers the study of state-of-the-art research on *Information Centric Networking* (ICN), *Publish/Subscribe* (pub/sub) and reliability and congestion control mechanisms. The study identifies the issues that exist in the existing solutions and reveals the need for a new pub/sub architecture.

Researches on gaming and service chaining are also studied here to provide some necessary background knowledge for G-COPSS and FCSC.

Contents

2.1 *Information Centric Networking*s (ICNs)

One major difference between ICNs and IP network is the forwarding component. IP network forwards data based on locations (IP addresses) while ICN communicates using content identities (names). This can be seen as the difference in the location of the name resolution functionality – in the application layer in IP networks while in the network layer in ICN. ICN proposals further differ in naming structures and resolution mechanisms in design which result in different efficiency and security performances.

This section covers the recent proposals related to ICN and their respective design choices.

2.1.1 Content Naming and Name Resolution

The purpose of a person using Internet is to retrieve the data object that (s)he wants. The function provided by the Internet can be seen as a data feature to data object mapping – the users express the features (e.g., key words) that they are interested in, and the network returns the data that they need. With the introduction of object resolution systems (e.g., search engines), the function is divided into two parts: data feature to data name mapping and data name to data location mapping. The object resolution systems help users get required names via a negotiation (refining) procedure with complicated *Artificial Intelligence* (AI) logic and the network provides a relatively simple and efficient way to retrieve data with the specific names.

Uniform Resource Locator (URL) is one form of content names that has been widely used in IP network. According to [9], URLs are character strings in the form of:

`scheme://domain:port/path/query_string#fragment_id`

Domain Name System (DNS) is a hierarchically distributed naming system that translates *domain names*, which can be easily memorized by humans, to the numerical *IP addresses* needed for the purpose of computer services and devices worldwide [10]. It serves as the name resolution service in the application layer. To get the location of the data represented by the URL, an explicit request is made towards a DNS server. This pre-binding between the name and the location (initiated from the object resolution service) limits the efficiency of the routing in the network – the data has to be retrieved from the original server rather than a better source nearby.

Therefore, ICNs propose to route requests/data by name. By placing the name resolution in the network layer in a hop-by-hop manner, ICNs can route the requests to any potential sources including the source, peers who serves the same data, or even caches in the network.

Different types of names are proposed to serve different needs of different proposals. They can be generally categorized into 3 types: flat names, hierarchically structured names and attribute-based names.

2.1.1.1 Flat Names

Flat names are simplest among all three name types. A flat name can be any string that identifies a piece of data. For the simplicity, flat names are usually hash tags of a data. In addition to simple strings, proposals like DONA [3], PSIRP [11], NetInf [12] and MobilityFirst [13] adopt self-certified names to achieve data integrity. Every piece of data can be verified by the name through hashing or signature.

A full-match on the names will be performed in the routers. The data consumers express interests on explicit names and the routers will forward the requests to the sources that serve the data. This kind of solution benefits the mobile networks. The parties of the communication can use a same identity for data dissemination and retrieval even if both parties are moving.

Nevertheless, such a naming strategy faces scalability issue. The routers have to maintain more states when the number of data objects increases in the network. To address this issue, explicit name resolution services (e.g., GNRS in MobilityFirst [13]) are introduced to this kind of architectures.

2.1.1.2 Hierarchically Structured Names

Hierarchically structured names are like URLs (e.g., `/dissertation/architecture.pdf`) and "/" is the delimiter between different components of a name. Such naming structure can help the aggregation and therefore achieve scalable forwarding and routing. E.g., a request to `/de/uni_goettingen/ifi/jiachen_chen/dissertation/architecture.pdf` can be routed to a server of Institute für Informatik (IFI) which serves all the files under prefix `/de/uni_goettingen/ifi`. The routers in the network only needs to maintain the aggregated states (prefixes) rather than the location of each data item. Proposals like NDN [4]/CCN [5] and TRIAD [14] use this kind of naming structures.

Longest-prefix-matching similar to the mapping of IP addresses is adopted in such naming structures. This kind of mapping can provide extra functionality compared to the full-match and has the potential to find the sources that most probably serves the data. But they are still efficient (can be accelerated by existing hardware) compared to the predicates as suggested in [15]. This allows the routers to be efficient and deployable in the network.

However, aggregation can face difficulty when the data can move around. When a popular data is served by more providers and caches in the network, explicit routing entries should be added to the routers to make use of these new sources. The scale of the routing table can be overwhelming.

2.1.1.3 Attribute-Based Names

Attribute-based names are no longer the identifiers of data objects. They can be either simple strings (like *Content Descriptor*s (CDs) [16]) or attribute-value pairs as was adopted in [2, 15]. Instead, they are more like descriptors that explains what is in the data object. A data object can have multiple such names and a name can denote a set of data. This kind of naming provides data consumers a wider range of selection. By expressing a name or a predicate on the name(s), the consumers can get all the data that meets the requirement, similar to a in-network searching (and routing).

Since some of the solutions use predicates in the intermediate nodes, they face the issues of scalability and efficiency. The amount of computational resource required for the predicate matching can be much larger than that for simple or longest-prefix matching. The amount of states stored in the network can also be huge due to the variety of predicates and their combinations.

It is believed that attribute-based naming can be a good choice for pub/sub communications since subscribers are agnostic about the exact name of the data when they are subscribing. Therefore they can only "describe" the feature of the interested data (via either predicates or descriptors).

The solution proposed in this dissertation tries to leverage the benefit provided by both hierarchically structured names and the attribute-based names to provide flexible support for pub/sub applications while avoid the complexity caused by predications. Therefore, hierarchically structured CDs are used as names in the solution.

2.1.2 Caching

To exploit the temporal locality of the data requests, proposals like [5, 12, 13], adopt in-network caching. Similar to *Content Delivery Network*s (CDNs), these proposals try to store the frequently-used data in the network to *1*) reduce inter-domain traffic which benefits ISPs, and *2*) reduce the request latency which benefits data consumers. The in-network caching in ICNs is provided with a lower granularity compared to CDNs. The proposals try

to leverage the buffer which are used for queuing in nowadays routers to serve as a temporal cache for a small amount of data.

The efficiency of the caches (cache hit rate) is heavily dependent on the cache replacement rules. A cache replacement rule decides which data can stay in the cache when cache size is not big enough to hold the entire data space. In additional to the simple *Least Recently Used* (LRU)/*Least Frequently Used* (LFU) mechanisms, many proposals try to increase the cache hit rate by exploiting the data popularity [17, 18], multipath [19], request probability [20], network coding [21], etc.

However, some reality checks on ICNs [22,23] show that the size of the in-network cache is inevitably smaller than the data space. Therefore solutions like [24] are proposed against the in-network caches. This dissertation shares the similar concerns on the cache efficiency due to its limited size. But instead of avoiding the possible benefits of caches, the solutions proposed in the dissertation try to better utilize the temporal locality of the users – via RP-based query/response (in COPSS) and a new query model for "any next" packet (in SAID).

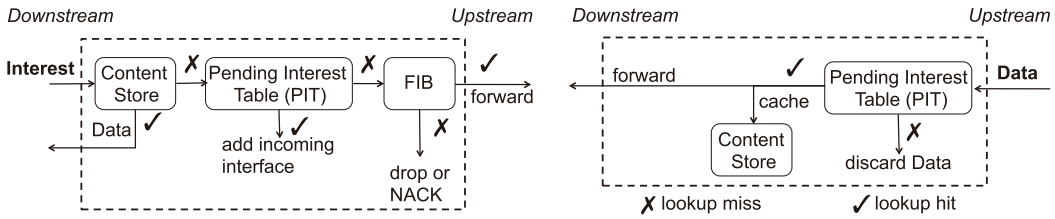
2.1.3 *Content Centric Networking* (CCN)/*Named Data Networking* (NDN)

NDN [4], also known as CCN [5] is a recent proposal aiming at delivering content efficiently without knowing the location of the content. Content sources in NDN register their availability of content by prefix (akin to a URI), and these prefixes are announced for global reachability.

There are two kinds of packets: *Interest* (i.e., request) and *Data* (i.e., content response). An Interest packet is sent by a consumer to query for data. Any data provider who receives the Interest and has matching data responds with a Data packet.

An NDN router has three data structures: the *Forwarding Information Base* (FIB) that associates content names to the next hops (termed *face*); the *Pending Interest Table* (PIT) that maps full content names with incoming face(s); and the *Content Store* that caches content from a provider upstream.

The packet processing in an NDN router is shown in Fig. 2.1. When an Interest packet arrives at a router, first the Content Store is checked to see whether the requested data is present in the local cache. If so, then this Data packet is sent out on the face that the Interest was received and that Interest is discarded. Otherwise, an exact-match lookup is done in PIT on the content name of the Interest. If the same Interest is already pending, then the incoming face of this new Interest is added to the face list of the matched entry and this new Interest is discarded. Otherwise, a longest-match is done on the content name in FIB and the Interest is stored in the PIT and a copy of it is forwarded based on the FIB entry. If there



(a) Interest processing (b) Data processing

Figure 2.1: Packet processing in an NDN forwarding engine.

is no matched entry, then the Interest is sent out on all the corresponding outgoing faces. When a Data Packet is received, the PIT is checked and a match means the Data has been solicited by Interest(s) forwarded by this node. In such a case, the Data can be validated, added to the Content Store and sent out on each face from which the Interest arrived.

2.2 *Publish/Subscribe* (pub/sub)

Publish/Subscribe systems are attractive because they relieve consumers from the strict synchronization in the time and location when the information is generated by the publishers [25]. Although meant to be content-centric in nature, current systems require the users to know the location of the publishers or the brokers (in the form of servers). In limited situations, information aggregators that collect, index and re-distribute the information in some form remove the burden from the users (and act as a rudimentary content-centric forwarder). Thus, most current pub/sub systems are built on a location-based architecture, which results in inefficiency both in data forwarding and information management at the user end. IP multicast [26–28], overlay multicast [29–31] and recently NDN [5] have been proposed to overcome some of these shortcomings, but they are not able to serve as an efficient full-fledge content-based pub/sub framework.

2.2.1 Existing *Publish/Subscribe* Systems

Existing work on pub/sub frameworks can be broadly classified into two approaches depending on how subscribers obtain data: pull-based and push-based. In a pull-based model, subscribers poll the publisher (or a broker) for any content/information update. This tends to create unnecessary overheads in server computation and network bandwidth when the update frequency is low compared to the polling frequency. Furthermore, pull-based mech-

anisms require the knowledge of the identity (domain name/IP address) of publishers (or servers acting as the broker).

In contrast, traditional push-based approaches maintain long-lived TCP connections (Elvin [1]) or notify subscribers via other means such as instant messaging (Corona [7]) or Rendezvous nodes (PSIRP [11]). These approaches face scalability issues since they require the maintenance of too many connections and states, and sometimes require that every publisher and subscriber are known to each other. The wide existence of NATs makes it impractical for every subscriber to have global visibility, thereby complicating push-based mechanisms. Overlay pub/sub approaches like Astrolabe [32] and SpiderCast [33] are agnostic of the underlying topology and therefore cause a lot of extra overhead.

To overcome the limitation of the aforementioned approaches where a subscription requires the knowledge of every content source, approaches such as ONYX [15], TERA [34], SpiderCast [33], and Sub-2-Sub [35] have been proposed as topic/content-based systems. In such proposals, users express their interest in content rather than sources (e.g., to a publisher in Twitter). COPSS adopts a CD-based approach similar to that adopted by XTreeNet [16] and SEMANDEX [36]. RSSs feeds and XMPP pub/sub [37] are used to publish frequently updated content such as news headlines, blog entries and etc., allowing users to subscribe to topics/publishers. Though both are intended as push-based applications, in reality they are essentially pull-based mechanisms that frequently poll various RSS sources or XMPP servers.

2.2.2 IP Multicast and Overlay Multicast

IP multicast [38] is a network-layer candidate solution for efficiently delivering content to multiple receivers. A sender sends data to a multicast group address that subscribers could join. Multicast routing protocols such as PIM-SM [27] construct and maintain a tree from the RP to all receivers of a multicast group. There are also well-known problems with IP multicast like limited group address space, no inter-domain multicast support, etc. Moreover the flat group address space will also cause inefficiency in forwarding. E.g., there are 2 groups of subscribers, one only interested in $football$ and the other interested in all the *sports* (including football). In IP multicast, a dissemination system can either let the second group subscribe individually to all the sports related groups, which will cause a lot more states stored in the network; or ask them to subscribe to a separate group, which will result in the publishers sending duplicate data to both the groups resulting in wasteful traffic in the network.

Overlay multicast [29–31] is another alternative for pub/sub communication, which allows data to be replicated at hosts along the dissemination structure (tree or mesh) thus

saving some of the network traffic and publisher load. Data replication, multicast routing, group management and other functions are achieved at the application layer. Thus, it enables easier deployment without the need to change the current IP infrastructure. However, overlay multicasts are agnostic to the underlying topology, likely resulting in forwarding inefficiencies.

2.2.3 Named Data Networking

NDN has limited intrinsic support for efficient pub/sub systems due to its intrinsic query/response design. ChronoSync [39] is a recent proposal in NDN that tries to synchronize the repositories of different data providers. This system can be used for file sharing and group communication. To avoid the growth of the "Exclude" field, each repository has a digest that represents the latest status. The data consumers have to explicitly request for the latest digest, compare it with the local repository, get the names of the updated files and then request for them. Although the solution provides more efficient pub/sub-like mechanism compared to the traditional solutions, it has shortcomings: *a*) the data consumers have to know all the prefixes of the providers and send requests explicitly, and *b*) polling-based mechanism causes overhead and affects timeliness.

2.3 Large Scale Reliable Data Dissemination and Congestion Control

Many protocols have been proposed to enable large scale reliable data dissemination. To maximize the utility of multicast, cyclic- and scheduled-multicasts [40, 41] have been proposed to benefit the consumers that are not starting from the same time. To deal with heterogeneous data receivers different types of congestion-control and repair mechanisms are proposed.

2.3.1 Provider-Repair Solutions

The clients in provider-repair mechanisms [42–47] send *Negative Acknowledgement*s (NAKs) to the provider (or the whole group to suppress duplicate NAKs) and the provider retransmits the missing packets specified in the NAKs. The data provider in such solutions has to align the sending rate to the slowest receiver eventually.

Rizzo [48] proposed a TCP-friendly single rate multicast congestion control mechanism that allows the data provider to align the sending rate to an *ACKer*. An ACKer is a selected

receiver that ACKs to the multicast packets. He/she is seen as the representation of all the receivers and therefore the ACK implosion can be avoided.

Due to the consideration of reliability and the friendliness to all the flows that share the dissemination tree, the slowest receiver is selected as the ACKer. In both these cases, the **overall efficiency** of the solution is dramatically affected by just a few slow receivers.

2.3.2 Peer-Assisted Solutions

Peer-assisted approaches like [49–54] group receivers in a hierarchical structure and the receivers ACK to the upper level in the tree so that the ACKs can be aggregated. By introducing such a relationship among receivers, these solutions allow receivers to perform local repair and therefore, the provider can align the sending rate to the majority of (or even the fastest) receivers based on the ACK strategy.

Unfortunately, in these proposals, the subscribers have to exchange information in a peer-to-peer manner in the IP network to perform repair as well as send ACKs. According to [55], these solutions face the problem of **privacy** and **trust**. Namely, the receivers have to reveal their identities (IP addresses) to the peers, and they have to trust the peers as there is no guarantee of data integrity during peer-repair.

2.3.3 Layered Multicast

Other than the single-rated multicast solutions, layered multicasts are proposed to deal with heterogeneous consumers. In [56, 57], the provider creates different multicast groups that transmit different resolutions of the data. The receivers can select appropriate groups according to their link capacity. These solutions are applicable to select applications. But having a reliable multicast capability for a single rate stream is still fundamental for broad-based use across all kinds of applications. Then the layered multicast solutions can be considered as orthogonal to the single-rate reliable multicast.

2.3.4 Named Data Networking

NDN provides aggregation in the data requests. When the data provider releases the data, it can be forwarded towards the consumers similar to a multicast. Several congestion-control mechanisms have been proposed for NDN like [58–62]. They all face the out-of-sync issue (see §7.2) since they rely on TCP-like mechanism. Other works such as [63] try to achieve efficient large scale data dissemination via pub/sub. But there is no clear indication of them

having an efficient mechanism to ensure reliability and avoid congestion collapse in the network.

2.4 *Massively Multiplayer Online Role Playing Games* (MMORPGs)

MMORPGs requires tight timeliness on the communication among game players. This increases the burden on both end hosts (servers and clients) and the network. Different solutions have been proposed to satisfy game requirements and they can be generally classified into 2 categories: server-based approaches and *Peer-to-Peer* (P2P) approaches.

2.4.1 Server-Based Approaches

Modern fast-paced action games run on a *Client/Server* (C/S) architecture. Due to the resource limit on the servers (including the computational resource and the network resource), these games limit the number of players who can interact simultaneously.

Feng *et al.* [64] observed that due to the limited resource, *Counter-Strike* (CS) was configured up to 22 players per game in the gaming server, and a CS server can support up to 32 simultaneous players.

Another popular MMORPG, Second Life, has multiple dedicated servers to support each of its 18,000 regions [65]. Studies such as [65–67] show that Second Life makes intensive use of network resources and that an Avatar action consumes about 20Kbps in the downlink and a movement made by the avatar could consume up to 110 Kbps on the downlink. Stenio *et al.* [66] also show that the management of a region with only 5,000 rigid-body objects requires about 72% of the server computational power.

The communication and computational overheads incurred in order to handle the players' update will increase super-linearly with the increase in number of players, limiting the maximum number of players in a game.

2.4.2 *Peer-to-Peer* (P2P) Approaches

P2P-based approaches (e.g., [68–72]) manage the virtual worlds in a distributed manner by leveraging end-user resources and therefore provide a scalable and cheap alternative to C/S gaming approaches.

Varvello *et al.* [72, 73] propose a *Distributed Hash Table* (DHT)-based architecture for Second Life to overcome the limitations of the C/S architecture. Donnybrook [71] is another system designed to handle games without server support. The shortcomings of these approaches are the high management overhead and network overhead when players move, as all actions are related to players' identities, which are agnostic to the underlying topology and require the server's involvement.

The above-mentioned designs failed to exploit the fact that in multi-player games, players are not interested in who sent the related events (e.g., update, player online/offline, action etc.) and how the information is disseminated, but the information itself. In pub/sub-based games (e.g., [74]), each player delivers a set of subscriptions describing events matching his interest. However, they did not use the predefined map partitioning information. Instead, they subscribe to arbitrary x and y ranges which is quite unrealistic in gaming scenario (there might exist mountains, walls etc., implying that players are not looking at a square of area). At the same time, it increases the computation overhead for forwarding since every node will have to compare 4 (possibly floating-point) values before it can decide where to forward.

2.5 Service Chaining

The need to perform additional processing of packets of a data flow in the network before it is delivered to the destination has become an integral element of providing Internet services. These functions include the modification of the packet header, collection of statistical information or even the modification of the payload. They are provided in the form of *Middleboxes* [75–77] for policy control, security and performance optimization. The middleboxes have to be resident on the path of a flow, which implies that the traffic has to be deviated from its "natural" IP shortest path and forced through the middleboxes. This work uses the term *Service Chaining* to describe the action of steering packets through these middleboxes.

Existing Service Chaining solutions can be broadly classified into 3 classes: indirection-based, policy-based and SDN-based.

2.5.1 Indirection-based Service Chaining

Several proposals for service chaining regard *indirection* as an indispensable element for achieving high flexibility to support various scenarios including node mobility, caching and anycast. Works in [75, 78] propose an architectural modification to TCP/IP networks in

order to allow further indirection than what is supported by DNS, allowing simple integration of middleboxes into the TCP/IP architecture. [78] highlights the problem of having an IP address – location-dependent element – as the identifier of end hosts, and proposes the introduction of several levels of indirection. Other similar works include [79–84]. Unfortunately, these solutions rely on predetermined nodes that provide the service, thus becoming inflexible to react to node failure as well as new instances of middlebox functionality.

2.5.2 *Policy-Based Routing* (PBR)

Cisco's policy-based approach [85] allows the administrator to specify adjunctive rules for routing, that are selectively applied depending on the traffic characteristics (e.g., IP 5-tuple, rate, etc.). Since the rules must be manually configured on each PBR router, the solution scales poorly and cannot dynamically react to network condition changes.

2.5.3 SDN-based Service Chaining

Several solutions have been presented that leverage SDN [86–88]. The general idea is to have a logically central controller that has a comprehensive view of the administered network portion and of the networking elements present. This controller can determine the best route for each flow that traverses the network and can take into consideration the potential need for this flow to go through one or more middleboxes. To make its decision effective, the controller must add forwarding rules to the involved switches, instructing them on the new next hop for each flow that deviate from its standard IP path.

Chapter 3

A *Content Oriented Network for Publish/Subscribe Systems* (COPSS)

Due to the intrinsic query/response design, enhancements are needed for NDN to efficiently support pub/sub communications – a highly desirable information-centric feature that enables temporal separation between information generation and indication of interest.

COPSS is designed to be a content-centric architecture that meets these requirements. By introducing the notion of *Content Descriptor*s (CDs) and 2 new packet types (Subscription and Publication), COPSS enables applications to have direct pub/sub communication. COPSS leverages *Rendezvous Point* (RP)-based communication to allow publishers and subscribers to join and leave dynamically. A dynamic RP balancing mechanism is provided to eliminate the well-known issue of traffic concentration in core-based trees. Via two-step communication, users can get timeliness and efficiency provided by COPSS and also exploit the benefits provided by NDN like dynamic request forwarding, caching and access control.

The key novelties of COPSS include:

- COPSS introduces the notion of CD [16, 36] into ICN. A CD goes beyond name-based [4] and topic-based [33] content identification and allows for contextual identification of information and supports ontologies and hierarchies in specifying interests.
- COPSS supports a CD-based subscription maintenance in a decentralized fashion, relieving the publishers and subscribers from having a detailed list of one another. This facilitates a highly dynamic and large scale pub/sub environment (in which the focus is on the content that is published) and facilitates the creation of new publishers and subscribers. This is analogous to recent events in Twitter where people belonging to an affected region were able to behave as publishers.
- COPSS supports push-based multicast capability to deliver content in a timely man-

ner. Additionally COPSS leverages NDN's inherent pull-based information delivery model to provide additional features for subscribers via a 2-step delivery model that allows information publishers to exercise policy control, access control. It utilizes a snippet-based initial dissemination so that delivery of large pieces of content can be achieved in a scalable manner.

This chapter first discusses the requirements for such platform and then describes the architecture of COPSS to meet these requirements. The evaluations of COPSS will be provided with its applications, namely, content-based Twitter (§4) and gaming (§5).

Contents

3.1 Architecture Requirements

Before jumping into the description of the solution, this work first discusses the requirements of a pub/sub framework.

An efficient pub/sub framework needs to support:

- **Decoupling publishers and subscribers:** In an ideal pub/sub environment, publishers only focus on their core task of publishing while not having to maintain membership status, and subscribers receive content from a multitude of sources without having to worry about maintaining a list of publishers and frequently polling them for the availability of fresh data. Moreover, a consumer may not wish (and it may even be infeasible) to subscribe to all of the "channels" belonging to a myriad of information providers that disseminate items of interest, either on demand (such as web, twitter, blogs and social networks), or tune to a broadcast channel (e.g., television, radio, newspaper). In these cases, support should be provided to the consumer who would prefer obtaining the data based on descriptors such as keywords, tags, or other properties of the published data.
- **Push enabled dissemination:** The ability to exploit push-based delivery is a key to achieving timeliness and to avoid wasting server and network resources because of redundant polls. Therefore, an efficient pub/sub architecture must provide the capability for publishers to push information to online subscribers interested in it. Such timely dissemination is necessary in many scenarios such as disaster (e.g., Tsunami) warnings, stock market information, news and gaming.
- **Scalability:** The target architecture should be able to accommodate a large number of subscribers as well as publishers (often subscribers are also publishers as user-generated content becomes common). Therefore, it should minimize the amount of states maintained in the network, ensure the load on the publisher grows slowly (sublinearly) with the number of subscribers. The load on the subscribers should also grow slowly with the number of publishers (e.g., dealing with the burden of duplicate elimination). Importantly, the load on the network should not grow significantly with the growth in the number of publishers and subscribers. There is also a need to accommodate a very large range in the amount of information that may be disseminated, and the need for all elements of the pub/sub framework in a content-centric environment to scale in a manageable way.
- **Efficiency:** The architecture should enable a nearly unlimited amount of information being generated by publishers, allow for delivery of information related to subscriptions independent of the frequency at which that information is generated by publishers. The architecture must utilize network and server resources efficiently. It is desirable that content is not transmitted multiple times by a server or on a link. Fur-

thermore, the overhead on publisher and subscriber end-points to query unnecessarily for information must be minimized.

- **Dynamicity:** The architecture should be able to deal with the substantial churn in subscription state, allowing a large number of users to join, leave and frequently change their subscriptions. The topics of interest may change frequently as well (e.g., in a Twitter-like publishing environment, where the popular topics change frequently).

Additionally, to support a full-fledge pub/sub environment, it is desirable that the framework support the following additional features:

- **Support hierarchies and context in naming content:** It is desirable to be able to exploit both context and hierarchies in identifying content. Hierarchical naming has been recognized by NDN as well. Exploiting context enables a richer identification of content (in both subscriptions and published information), as noted in the database community (and adopted in [16]).
- **Support two-step dissemination for policy control and user interest:** There is a need for pub/sub environments to support a two-step dissemination process both for reasons of policy and access control at the publisher as well as managing delivery of large volume content. In such a scenario, the pub/sub framework would be designed to publish only a snippet of the data (containing a description of the content and the method how to obtain it) to subscribers. The subscribers then request for the content based on their interest and allowance.

3.2 COPSS Overview

To meet the requirements listed above, COPSS integrates the following designs:

- COPSS introduces the notion of CD to decouple publishers and subscribers (§3.3). The users publish and subscribe based on the CDs instead of the each others' addresses.
- COPSS introduces 2 packet types (Subscription and Publication) that use CDs as forwarding identities (§3.4).
- COPSS adopts RP-based (multicast) communication to satisfy the scalability and dynamicity requirements (§3.5). The subscribers can join/leave multicast trees rooted at RPs rather than publishers. The states in the network can be reduced because of subscription aggregation. The publications are therefore forwarded in a multicast manner and the network load is reduced.
- To achieve RP-based communication with minimum modification to existing solu-

tion, COPSS adds a *Subscription Table* (ST) to NDN forwarding engines (§3.6). This table maintains the subscriptions per outgoing face. The states stored in the table grows (sub)linearly with the number of subscribers because of the aggregation.

- To solve the well-known traffic concentration problem of core(RP)-based trees, COPSS incorporates an inherent automatic RP balancing mechanism (§3.8).
- Considering the subscribers interest and the need for policy control, two-step communication is provided with the COPSS design to reduce the wasteful traffic in the network (§3.9).
- To ensure the feasibility of COPSS design, an implementation based on NDN is provided (§3.10).

3.3 *Content Descriptor*s: Hierarchical and Context-based Names

While enhancing NDN, COPSS keeps the hourglass layering model suggested by [5] and works on the "content-chunk" layer just as NDN. Instead of identifying every piece of content by its Content Name, COPSS adopts the notion of *Content Descriptor*s (CDs), which could refer to a keyword, a tag or can be combined with a property (e.g., hierarchical structure) of the content. A CD is a human-readable, hierarchically structured string (similar to a ContentName in NDN), but a piece of data (e.g., a document) can have multiple CDs and at the same time there may exist multiple data items that are identified by a given CD. A data item with CD set CD_{data} will be delivered to the subscribers with subscription set S if:

$$\exists\, cd \in CD_{data}, s \in S, where\ s\ is\ a\ prefix\ of\ cd. \tag{3.3.1}$$

For example, a news published by CNN that is related to German football match can have CDs `/CNN` and `/sports/football/Germany`. This data item will be delivered to every subscriber who subscribes to `/CNN`, `/sports`, `/sports/football` or `/sports/football/Germany`. Of course, a subscriber can also subscribe to `/CNN` and `/sports/football` at the same time. A subscription can therefore be at different granularities by taking advantage of this hierarchy. Such naming scheme facilitates COPSS aware routers to aggregate subscription information and avoid the forwarding of duplicate content.

3.4 COPSS Packet Types

In order to support pub/sub operation directly, COPSS introduces two additional types of packets: *Subscription* and *Publication* (see Fig. 3.1). The packets use *Type-Length-Value*

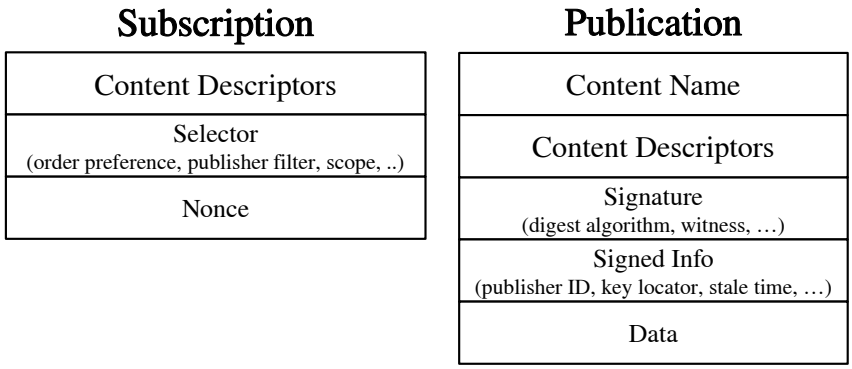


Figure 3.1: COPSS packet types.

(TLV) format for extensibility.

A Subscription packet is very much like an Interest, it contains the CDs a user wants to subscribe to, some selectors and a nonce to avoid loop. By issuing a Subscription for a CD, a subscriber will receive updates when publishers publish new contents. For the sake of efficiency, multiple CDs is allowed to appear in a single Subscription and therefore a subscriber can subscribe to a set of CDs via a single packet. A subscriber can also unsubscribe from CDs by setting a flag in the subscription.

A Publication is like a Data packet with CDs. For the efficiency on some transit Publications – the Publications that are not needed later – Content Name field can be left empty in Publication. Note that a Publication can also have several CDs so that the data provider does not have to send multiple copies for a same data object with multiple CDs.

3.5 *Rendezvous Point* (RP)-Based Communication

As a content-centric communication model, pub/sub always involves high dynamicity – the subscribers cannot know who would publish the next message they are interested in. Similarly, it is also quite difficult for the publishers to know (and maintain) the subscriber list. To meet the dynamicity requirement of pub/sub, COPSS follows the design of Protocol Independent Multicast - Sparse Mode (PIM-SM [27]), *Rendezvous Point*s (RPs) are used as the delegate for the publishers. The publishers and subscribers therefore only need to know the location/name of the RPs instead of the location/name of each other, and a join/leave of the publisher/subscriber will not affect the states maintained on the other end hosts.

In COPSS, every CD is associated with an RP and a dissemination tree rooted at the RP

is formed. The subscribers are the leaves of the tree and the intermediate routers are the internal nodes. Every router maintains the subscriptions of the next hop in the tree. While delivering a content from the publisher, the network first sends the packet (unicast) to the RP (s) that is(are) responsible for the CDs in the packet. On reaching the RP (s), the Publication is then disseminated along the trees towards the subscribers. Every intermediate router can replicate the packet if there are multiple outgoing faces that have subscribers downstream. A push-based multicast based on CDs is achieved.

Note that the RPs in COPSS are distributed modules on the routers, not the centralized servers. Each RP has a Content Name propagated to the whole network so that it can be reachable by every possible publisher and subscriber. The Content Name (FIB) propagation is left to the routing algorithms proposed in NDN like NLSR [89]. Since RP is represented by a name, it can be moved when a router fails. Multiple RPs can be created for different CDs to avoid traffic concentration and provide better scalability and efficiency (see §3.8 for detailed description).

For simpler subscription management in a dynamic network condition, e.g., mobile networks, disaster environment, and etc., COPSS uses soft-state subscription similar to PIM-SM. Every subscription contains a timeout and the clients need to subscribe to the network periodically. The timeout value can vary according to the link stability, e.g., mobile devices can have a shorter timeout period so that the subscriptions can be deleted faster after the device is disconnected suddenly; stable servers can have longer timeout period so that they will not create a lot of control messages in the network. But this soft-stated subscription can be transparent to the upper layers. The users do not have to send the refresh subscriptions explicitly. Instead, the operation systems and the 1^{st} hop routers can take the responsibility.

3.6 COPSS Forwarding Engine

While designing COPSS, this work tries to achieve significant architectural and functional improvement with minimal changes. This requires COPSS to reuse the existing packet types and router structure behaviors as much as possible. Fig. 3.2 shows the COPSS forwarding engine model. Compared with a standard NDN router, COPSS aware routers are equipped with a *Subscription Table* (ST) that maintains CD-based subscription information downstream.

An ST can be seen as a $CD \mapsto \{Face\}$ dictionary. On receiving a Publication, the router looks into its ST and find out which face(s) to forward to based on Eq. 3.3.1. The ST is very much like a PIT but the difference is Publication packets do not consume the subscriptions. The subscriptions can only be removed when there is an explicit unsubscription or timeout. Therefore, unlike the PIT which is modified on every packet arrival, a large proportion of

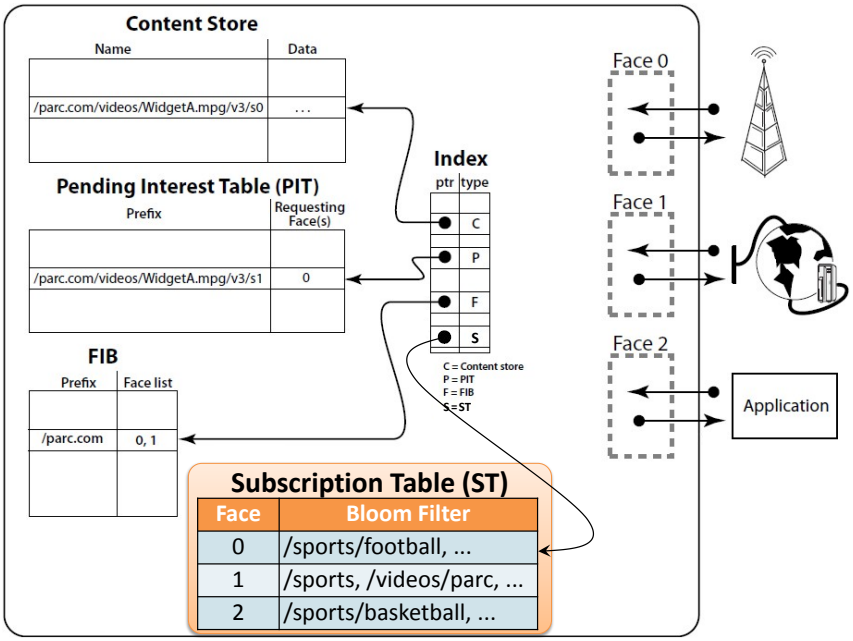


Figure 3.2: The COPSS forwarding engine – adapted from the NDN forwarding engine by adding a *Subscription Table* (ST).

operations on the ST is pure lookup. For the sake of efficiency, COPSS chooses to use a separate data structure instead of reusing the PIT. While implementing ST, a hardware/software developer can optimize the lookup speed and parallel lookups can be easily achieved when a router has more than one CPU (it is proved to be difficult to implement PIT in a similar way).

The implementation of ST is not limited in COPSS. It can be a trie-based data structure as was proposed by [90] or hash-based as was proposed by [91]. To reduce the size of ST, a BloomFilter [92] can be also used as an alternative to hashtable. While using BloomFilter, the router can create a BloomFilter for every face and the forwarding decision can be made by a BloomFilter match. Therefore, the ST would look like a $Face \mapsto BloomFilter_{<CD>}$ as is shown in Fig. 3.2. It is true that BloomFilters can introduce slight false positives in the network, it is still tolerable for the exchange of performance. These extra packets will eventually be eliminated at the 1^{st} hop router of end hosts to be friendly to the end devices with critical energy/bandwidth conditions.

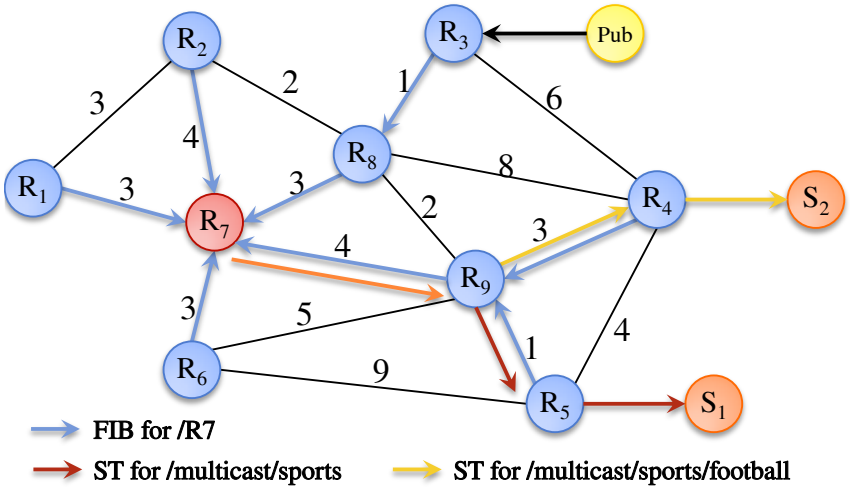


Figure 3.3: Topology for publication delivery example.

3.7 Protocol Exchange

After the description of different components in the network, the whole pub/sub procedure is described in this section via a simple example. The toy topology is shown in Fig. 3.3. The topology contains 9 COPSS routers (R_1-R_9) and the link latencies (in milliseconds) are shown in the figure. R_7 is selected as the RP for all the CDs in this example and the name of the RP is `/R7`. The routing mechanism will propagate the routing information for `/R7` throughout the network. The FIB for `/R7` is represented as a solid arrow.

3.7.1 Subscribe to CDs

To subscribe to (i.e., declare an interest in) a CD, a COPSS subscriber needs to send a Subscription packet containing the CD to the 1^{st} hop router. The router would add an ST entry (CD$\mapsto$outgoing face of that subscriber). If the router has already subscribed to this CD or a prefix of it, the router will not propagate this subscription upstream. Otherwise, the router needs to find out the name of RP that serves the CD (RP_{CD}). This mapping is described in §3.8.2. A FIB lookup on RP_{CD} is then performed to determine which face to forward the Subscription.

In the example, S_2 sends a Subscription with CD `/multicast/sports` to its 1^{st} hop router R_4. R_4 adds an ST entry (represented as a dashed arrow) for this subscription. Since there is only 1 RP that serves all the CDs, R_4 looks up FIB for `/R7` and forwards the Sub-

scription towards R_9. The intermediate router(s) does the same and eventually a dissemination path is formed from R_7 to S_2. When S_1 subscribes to `/multicast/sports/football`, the Subscription will be forwarded to R_9 through R_5. R_9 sees that there is already a subscription that is a prefix of the CD, it just adds an ST entry without forwarding this Subscription.

In the real network, there might exist multiple RPs that serve different CDs and a Subscription normally contains multiple CDs. The intermediate routers need to lookup FIB for each RP_{CD} and split the CDs in the Subscription according to the outgoing faces.

3.7.2 Publish a Data

A Publication from the publisher has to be unicasted to the RP before it can be disseminated along the multicast tree. To better reuse the existing routing mechanism in NDN, the 1^{st} hop router of the publisher encapsulates the Publication into an Interest with the Content Name of the responsible RP. The NDN routing mechanism then takes the responsibility to forward the packet to the RP. For efficiency and Interest collision avoidance, the timeout value of such Interests are set to 0 so that they will not be added to PIT. The encapsulation mechanism also reduces the computation on the intermediate routers since they do not have to perform lookups on each CD in the Publication. If a Publication packet has CDs that are associated to more than 1 RP, the 1^{st} hop router has to split the CDs and encapsulate the packet into different Interests with different RP names. It can cause redundant packet being delivered to the subscribers and these packets will be discarded by the end systems. This issue is unavoidable as long as multiple RPs are used in the network, but better partition of RP responsibility might reduce the frequency of such occurrences.

When the RP receives an encapsulated packet (Interest) with its own name, the RP decapsulates the packet and gets the original Publication. The dissemination from RP to the subscribers is relatively straightforward. All the intermediate routers lookup the ST and forward the packet to the faces that satisfy Eq. 3.3.1.

In the example, *Pub* sends a Publication with CD `/multicast/sports/football`. The 1^{st} hop router R_3 realizes that the RP with name `/R7` is responsible for this packet. R_3 then encapsulates the Publication in an Interest with Content Name `/R7`. The intermediate routers forward the packet towards R_7 according to the original NDN logic. R_7 decapsulates the Interest since the packet has the RP name it serves. The decapsulated Publication will be forwarded to R_9 based on the ST in R_7. R_9 replicates the packet since both R_4 and R_5 satisfies Eq. 3.3.1. The Publication will reach both subscribers eventually. If *Pub* sends a Publication with CD `/sports/basketball`, R_9 will only forward it to R_4 and S_1 will not receive this packet.

3.8 Handling Hot Spots and Traffic Concentration

It is common for people to get interested in the same topic thereby resulting in some CDs becoming very hot (popular), both in terms of the number of subscribers and number of publications in a short span of time. The *hot spots* in some CDs are therefore likely to be subjected to high loads due to frequent publications from many publishers. The multicast capability in COPSS permits scaling to a large number of subscribers. Nevertheless, like core-based tree approaches in IP multicast, if multiple publishers send publications to CDs served by one RP, traffic concentration will result on the links leading to that RP and could therefore also contribute to the load and queueing on the RP. Since the RPs are responsible for handling a certain number of CDs, it is difficult to predict the number of RPs required or to perform predetermined load balancing during the initial distribution of CDs. A more general solution is required.

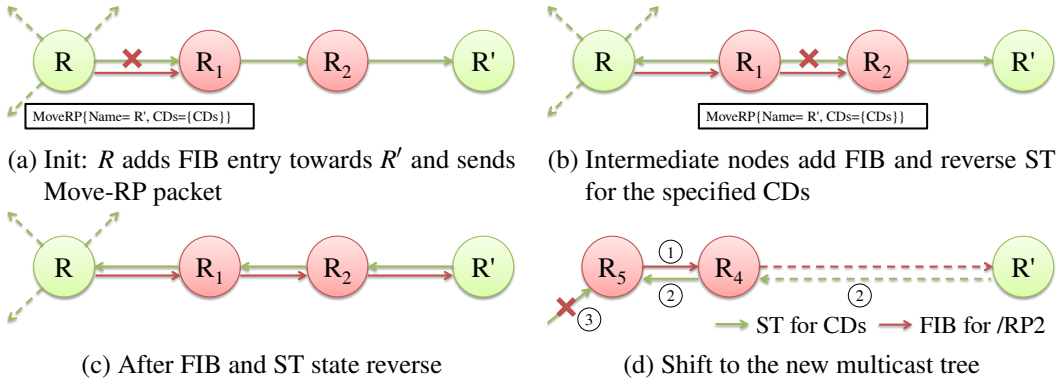
COPSS has built-in mechanisms to address hot spots including the resulting traffic concentration. It dynamically adds and deletes RPs and also reassigns CDs based on their popularity. The hierarchical map formation allows seamless decentralization and load-balancing of RPs. Since end-hosts send/receive multicast packets related to CDs (rather than the address of the RP), they are not directly affected by the addition/removal of RPs. The new RP only needs to indicate that it serves the new CDs. The other nodes will modify the FIB and ST accordingly and packets will be redirected automatically to the new RP based on a newly formed multicast tree. The solution is detailed here.

3.8.1 Automatic RP Balancing

The automatic RP balancing is performed when an RP is (nearly) overloaded. The adoption of a new RP is straightforward – by propagating the RP name and the CDs it serves. But to ensure that there is no packet loss during the shift of the RP, COPSS uses the following stages:

3.8.1.1 New RP and Related CDs Selection

To decide which CDs to move, the router monitors the traffic for each CD in a sliding window fashion of the recent N packets. Since the goal is to balance the load between the old and the new RP, the CD selection function divides the CDs into 2 groups based on the capabilities of both the RPs. The new RP selection function is similar to that in IP multicast. It may be performed by a network manager or calculated by a Network Coordinate function



(a) Init: R adds FIB entry towards R' and sends Move-RP packet

(b) Intermediate nodes add FIB and reverse ST for the specified CDs

(c) After FIB and ST state reverse

(d) Shift to the new multicast tree

Figure 3.4: Automatic RP balance procedure – ensuring no packet loss.

like [93]. It can be further optimized using prediction to ensure that addition and deletion of RPs are not performed frequently.

3.8.1.2 Reverse FIB and ST Entries between Old and New RP

After selecting the new RP (R'), the current RP (R) continues to act as the core till the complete network is aware of the new R'. An example of the initial state of RP shift is shown in Fig. 3.4a. R sends a MoveRP packet containing the name of R' and a list of CD that R' needs to handle. At the same time, R adds a FIB entry for the name of R' towards the outgoing face (R_1) and removes the ST entries for all the CDs to be moved. From that moment on, R will recapsulate the Publication packets containing the moved CDs into the name of R' and these packets will be forwarded to R' and decapsulated there.

On receiving the MoveRP packets, the intermediate routers will also reverse their ST entries so that R is in the subtree formed with R' as the root. This ensures that all subscribers who are still attached to R continue to receive messages. The intermediate routers will keep forwarding the MoveRP packet until it reaches R'. Fig. 3.4b shows that R_1 has added the new FIB entry, reversed its ST entries and the MoveRP packet is forwarded towards R_2.

Once R' receives the MoveRP packet, it will be ready to behave as the RP for those CDs. The state at the end of this stage is shown in Fig. 3.4c. Since the edge routers are not yet notified about the new RP, they keep encapsulating the related Publications using the name of R. These messages will be sent to R and R will recapsulate the message as was described above. Since the ST entries have been reversed on the path between R and R', when R' multicasts, it will be forwarded back to R and then towards the original multicast tree. A communication path between $Pub \rightarrow R \rightarrow R' \rightarrow R \rightarrow Subs$ is created.

True that at the end of this stage, the amount of data that goes through R is even higher than initial state, but a multicast tree rooted at R' is created with no packet loss. The workload reduction will be achieved in the next stage. It is easy to prove that no packets would get lost during this stage (half a *Round Trip Time* (RTT) between R and R'): The messages that arrive after the initial stage will be recapsulated using the name of R'. Since the intermediate routers follow *First Come First Served* (FCFS) rule, all these routers must have received the MoveRP packet and added FIB entry for R'. The recapsulated packets will arrive R' eventually. When R' receives the recapsulated packets and starts to multicast, an ST path from R' to R is already built. The Publications will go back to R and disseminated following the original tree.

3.8.1.3 Propagate New RP Information

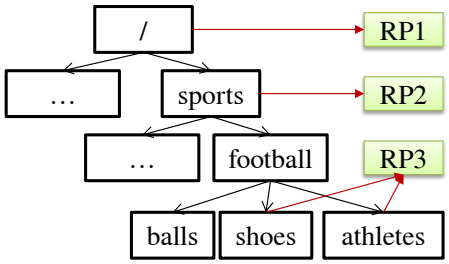
In this stage, R' propagates its serve information and becomes an RP independent of R. With more edge routers knowing the fact that R' is serving the set of CDs, more messages will be sent directly towards R'. The load on R will be gradually reduced.

R' starts this stage by propagating a packet similar to a prefix serve information in NDN. The moved CDs are carried along with this packet. The routing mechanism will forward the packet throughout the network and create a tree (for Interests) rooted at R'.

On receiving such a packet, a normal router in the network should add a FIB entry just like how it deals with a prefix serve packet. Along with that, the router should change its ST according to the new FIB direction. After the new RP notification packet is propagated throughout the network, a multicast tree (for Publications) rooted at R' can be constructed.

However, a simple mechanism as described above can cause packet loss during this stage, since the new upstream routers might not be in the same tree and the Publications will get lost before the whole tree is constructed. COPSS uses a solution similar to that adopted in [94]. Fig. 3.4d shows the actions R_5 will perform on receiving a new RP notification packet. R_5 adds a FIB entry (action ①) as described above. But instead of unsubscribing from the old face (action ③) directly, R_5 adds a subscription towards the new face first. The subscription information about the CD including both the old and new face will be stored as a *pending subscription*. During this period, multicast packets can still arrive via the original path. On receiving a join request, the new upstream routers will check their own subscription status, which can be classified into the following 3 cases:

- Not in the tree: the router will add a pending subscription and send a join packet upstream;
- Already in the tree: the router will send a *confirm* packet back; and



Storage on 1st hop routers

CD	RP
/sports/football/athletes	RP3
/sports/football/shoes	RP3
/sports	RP2
/	RP1

Figure 3.5: Example of CD to RP mapping.

- Have a same Pending Subscription: it will wait for the confirm upstream and send an Unsubscribe to the original upstream router and pass the confirm downstream.

In the example, R_4 is not in the subscription tree and therefore it adds a pending subscription and forwards the join packet upstream. After the upstream routers join the tree, R_4 will receive a confirmation. It will then turn the pending subscription into a normal subscription and forward the confirmation to R_5 (action ②). On receiving the confirmation from R_4, R_5 can then unsubscribe from the old face (action ③).

In this stage, the router does not leave the original multicast branch before it is added to a new branch. No packet will get lost in this stage.

3.8.2 Management of CD-RP Mapping

Since COPSS uses a dynamic service relation between RPs and CDs, an efficient CD-RP mapping is important so that:

- A Publication should only be sent to (at most) 1 RP according to the CD related to it;
- The size of the Subscription information stored in the network should be minimized; and
- It should be convenient to configure unambiguous mapping to avoid possible misbehavior of the network due to the mistake in the configuration.

To meet these requirements, COPSS adopts a **Longest Prefix Match** CD-RP Mapping table, which can fit well with the automatic RP balancing strategy and provides unambiguous semantics in the network. Fig. 3.5 shows a CD to RP mapping in COPSS. At the beginning, RP_1 serves every CD (`/`) in the network. Later, it finds that `/sports` becomes hot and moves it to RP_2. CDs `football/shoes` and `football/athletes` are further separated due to the workload. The CD to RP mapping table stored on the 1^{st} hop routers are shown next to the figure. When publishing, the 1^{st} hop router of the publisher will lookup

the table and find the matching RP using longest prefix match. E.g., a publication with CD `/sports/football/athelets` will be encapsulated using `/RP3` and a publication with CD `/sports/basketball` will be encapsulated using `/RP2`. While subscribing, the subscription will go to all the RPs in the hierarchical subtree and 1 level above. E.g., if a subscriber is interested in `/sports/football`, the edge router will send the subscription to both RP_2 and RP_3.

To further reduce the size of CD to RP mapping table at every edge router and for ease of management, COPSS suggests to use a broker that maintains the complete mapping while the edge routers only caches a small portion of it. When a router experiences a cache miss, it goes to the broker. Similar to the NDN design, intermediate routers can also respond to this request. The data structure for storing this table can be chosen from one of the following two options:

- The first is a traditional CD to RP dictionary. The index of CDs can be grouped into a tree (trie) structure to optimize search performance. The routers would only have to map the CD once before it sends the packet.
- The second option is a bloom filter. For every RP, there is a (counting) bloom filter storing the hash of the CDs it serves. This can compress the index space and reduce the cache miss rate. However, because of the false positives, the router will have to test all the bloom filters before it can forward the packet. This could also result in packets being sent to the wrong RPs (because of the false positives), thereby resulting in wasted network traffic and also computation overhead in the RPs to check if it indeed serves the CD in the packet.

The router designers can choose one of the options based on the size of the CD space and the number of RPs in the network. But the forwarding logic will keep the same no matter which option is adopted.

3.9 Two-Step Communication

Applications need two-step communication model due to the subscriber interest and publisher policy control. This section describes how COPSS realizes the communication model and a routing optimization in this environment is also proposed.

3.9.1 Two-step in COPSS

Two-step solution is required because of the following 2 folds of reasons:

- **Subscriber Interest**: It is quite common that a subscriber is interested in a CD, but (s)he might not be interested in every piece of data under that CD. The delivery of such kind of data can incur overhead especially when the data size is large.
- **Policy Control**: In many cases, the data items are under the policy controls (e.g., payment, security, etc.) from the publishers. A representative of such kind of communication is film delivery system – the publishers need to notify the subscribers about the new film, but the subscribers have to pay for the film before they can download it.

Therefore, the applications with relatively low subscriber interest ratio, large data size and/or policy control can use the two-step communication.

In the two-step communication, the publisher publishes a *snippet* rather than the real content to all the subscribers. This can be seen as the notification for the availability of the data but network does not differentiate it from a normal Publication in one-step communication.

A snippet is a payload carried in the Publication packet instead of the actual content. It contains the message abstract(s), pricing information and anything else that helps a subscriber decide if (s)he would like to have the whole content. Additionally, the snippet consists of the ContentName that would help the subscriber to obtain the data from publishers or other sources that are serving the same Data. The ContentName can be realized in a similar manner as in NDN and must be a unique way of identifying the served content.

Published data item can contain multiple CDs, e.g., a news article about "An injury to an American football player" could have a CD for `/news/usa` and a CD for `/sports/football`. In such a case, the snippets sent by the publisher to the different CDs could either be the same or even be different, pertaining to the taste of the subscribers. Therefore, a subscriber can receive multiple snippets pointing to a same piece of data (identified by the same Content Name), the end system can choose to combine these snippets or simply discard some of them.

On receiving a snippet, the subscriber can choose to query the real content if (s)he is interested in the data. For the query to reach the publisher that serves the data, the publisher must first register with the NDN routing on the name prefixes of the content to be published (just like the announcement of the content sources in pure query/response). Other subscribers that have already received the data could also serve the content by propagating the appropriate FIB entry to minimize the load on the publishers, especially in the case of large volume content (access control will have to be negotiated with the publisher).

In Fig. 3.6, the publisher publishes the snippet of a data item with CD `/sports/football` and Content Name `/BBC/WorldCoup/027`. On receiving this snippet, a subscriber decides to get the data by sending out an Interest with Content Name `/BBC/WorldCup/027`. To be

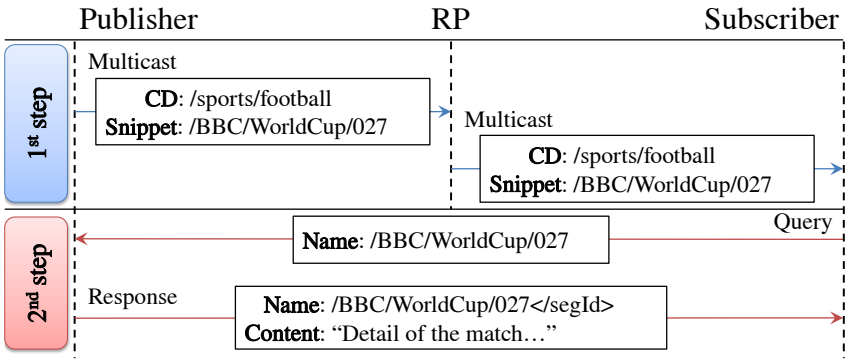


Figure 3.6: Protocol exchange for two-step communication.

able to receive the requests, the publisher needs to register to the NDN routing with a prefix of the data (e.g./BBC). The publisher can then respond with Data packets after some policy checks.

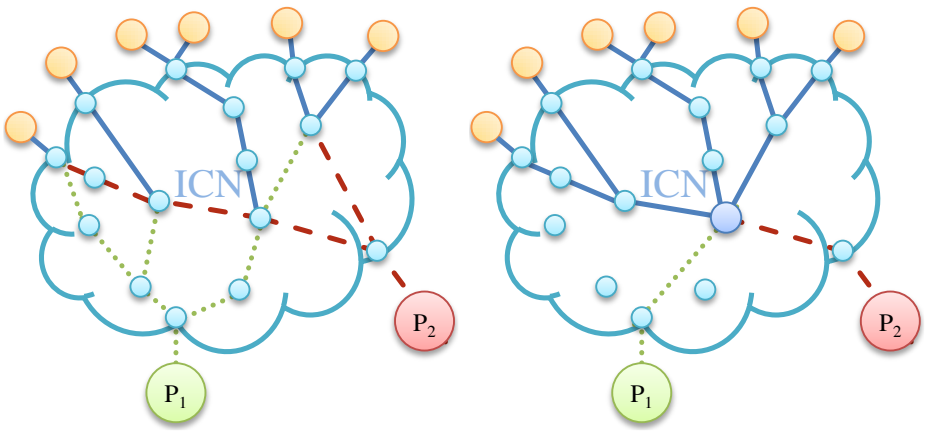
3.9.2 RP-based Query/Response

In the query/response communication, either it is the pure query/response via NDN or the second-step in COPSS, the data provider needs to register the prefix of contents (s)he serves with the routing mechanism. This work proposes a routing algorithm that creates a spanning tree rooted at the data provider with the data consumers as leaves. For example, in Fig. 3.7a, two publishers P_1 and P_2 serves prefixes /P1 and /P2 respectively. The routing algorithm creates a spanning tree for /P1 (dotted lines) and /P2 (dashed lines) rooted at the corresponding data provider. The solid lines show the shared part of the trees. The prefixes usually have to be maintained in the FIB on all the routers in the network since there might exist data consumers everywhere in the network. The total FIB entry size in the network can be calculated as:

$$Size_{FIB} = (n_{rp} + n_{pub}) \times n_r, \tag{3.9.1}$$

where n_{rp} is the number of RPs, n_r is the number of ICN-aware routers and n_{pub} is the number of publishers.

However, in COPSS environment, the RPs are already well-known to the whole network (for multicast). If the query/response tree can be aggregated at the RP, the number of FIB entries stored on every router will be reduced: only RPs need to know how to reach the publishers. In Fig. 3.7b, the solid lines from the subscribers to the RP represent the FIB entries for prefix /RP, and the forwarding information for prefixes /P1 and /P2 is only



(a) Original FIB status (b) New FIB status

Figure 3.7: FIB propagation for query/response.

maintained between RP and P_1/P_2. The new FIB entry size in the network can be:

$$Size'_{FIB} = n_{rp} \times n_r + \sum d_{rp,pub}. \tag{3.9.2}$$

where $d_{rp,pub}$ is the distance between the publisher and the responsible RP. Since $\sum d_{rp,pub} \ll n_{pub} \times n_r$, the new FIB size can be much smaller than the original (in Eq. 3.9.1).

To achieve this kind of communication, the 1^{st} hop router should add the name of RP as a prefix to the original Content Name in the Interest. It is similar to the encapsulation for a Publication. When RP receives such Interest, it "decapsulates" (remove the prefix of) the packet and forward the original request towards the provider. The RP should also take the responsibility for renaming the Data when the Data comes back from the provider.

Although this optimization is optimal for query/response in the 2 stage dissemination where the subscribers are aware of the RP, this model can also be applied to the NDN-like query/response model. In that case, new RPs should be created to serve specific name prefixes.

3.10 COPSS Implementation

This section explains the implementation of COPSS on top of an NDN forwarding engine. To ensure the seamless backward compatibility with NDN, a wrapping design is adopted while implementing COPSS.

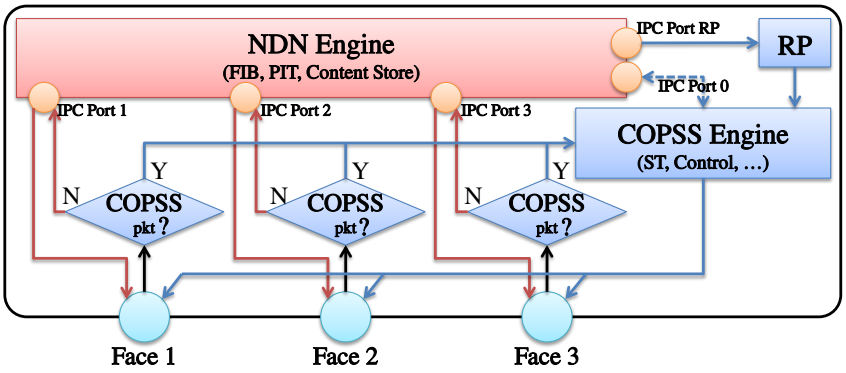
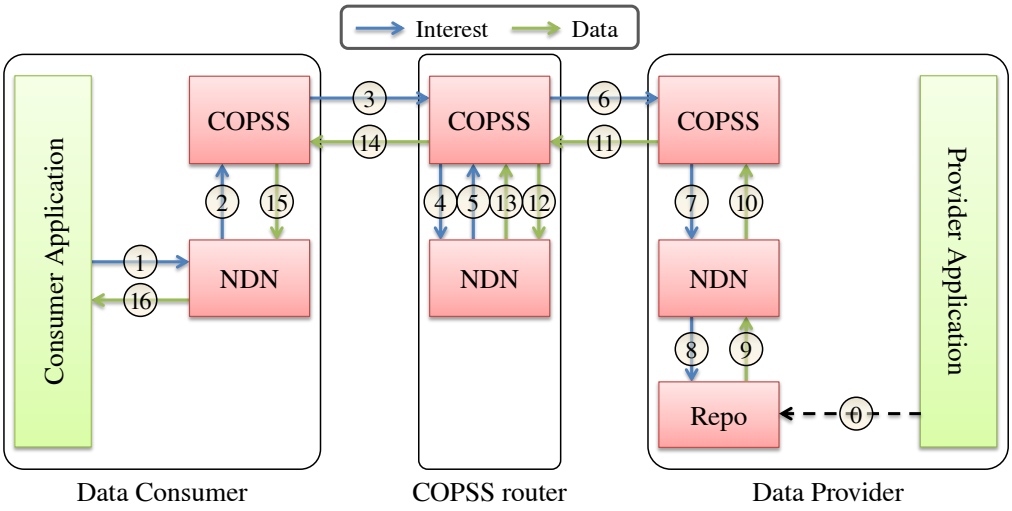


Figure 3.8: COPSS router design.

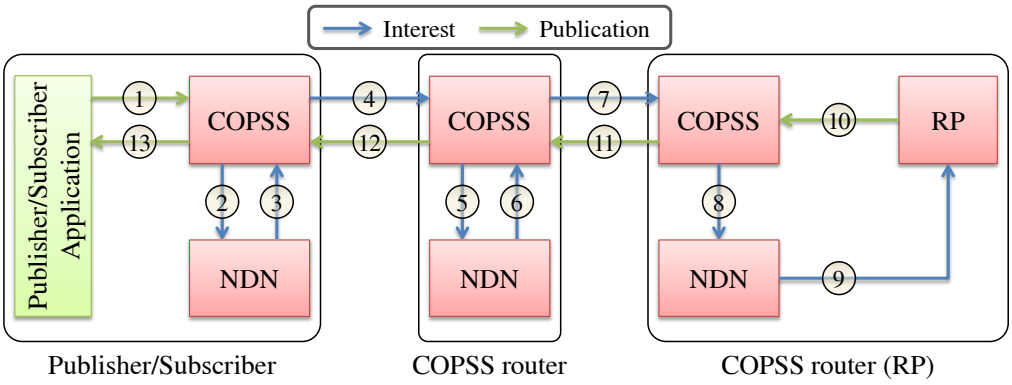
Fig. 3.8 shows the modules of a COPSS forwarding engine in implementation. COPSS creates a tunnel between each face and NDN engine via an *Inter-Process Communication* (IPC) channel. For NDN packets (Interest, Data, or any packet that is not COPSS recognizable), COPSS forwards them back and forth between the faces and the corresponding IPC ports. Similarly, when a routing module tries to add a FIB entry {`/prefix`, Face}, it will add {`/prefix`, IPC port of the Face} instead. E.g., in Fig. 3.8, when adding a FIB entry: {prefix=`/dissertation/COPSS`, face=Face 1}, the COPSS engine will add {prefix=`/dissertation/COPSS`, face=IPC Port 1} in NDN engine instead. Assume that an Interest comes from Face 2 asking for data `/dissertation/COPSS/fig1.pdf`, the packet will be forwarded to NDN engine through IPC port 2. NDN engine then decides to forward the interest through IPC port 1 based on the FIB. COPSS can forward the Interest out through Face 1 eventually.

Fig 3.9a shows the packet flow of a query/response in the implementation. After the provider stores the data object in his/her local repository (action ⓪), the consumer starts to request for the data. The request is sent directly towards the local NDN engine according to the NDN implementation. According to the FIB on the consumer, NDN engine forwards the Interest towards COPSS and COPSS forwards the packet out based on the tunnel to face mapping (actions ② and ③). The intermediate router and the provider performs the same tunnelling (COPSS) and forwarding (NDN) logic and the Interest will reach the repository (actions ④-⑧). When the repository returns a Data packet, the packet will be forwarded along the reverse path and reach the consumer application eventually.

Although the packets will be forwarded back and forth on every router, these communications are IPCs and therefore are much faster than the line speed. Such implementation turns out to be flexible (compared to patching NDN code directly) since it keeps high cohesion and low coupling between the NDN module (for query/response) and the COPSS module (for pub/sub). COPSS implementation has been adapted to CCNx [8] versions from 0.4.0 to



(a) Interest/Data packet flow in COPSS implementation



(b) Publication packet flow in COPSS implementation

Figure 3.9: Packet flow in COPSS implementation.

0.8.0, and NDN [95] versions 0.1 to 0.3 with almost no modification in the program. While designing pub/sub related communications, the designers do not have to worry about the NDN implementation either.

Some COPSS functionalities are achieved independent of the NDN engine. On receiving COPSS-recognizable packets (e.g., Subscription, Publication, etc.), the listeners on the faces will forward them to the COPSS engine instead of acting like a tunnel. COPSS engine will then modify the corresponding data structures and forward the packets based on the logic described above. A dedicated IPC tunnel 0 is used for the communication between COPSS and NDN engines.

RP is implemented as a separate module. Whenever an RP is created, such a module is instantiated on the corresponding COPSS router. The new instance of RP listens to the NDN engine directly on the name it serves. On receiving Interest packets, it will try to decapsulate the packet and forward the Publication to the local COPSS engine.

Fig. 3.9b shows the packet flow of a Publication in COPSS implementation. When the COPSS engine on the client receives a Publication packet, it encapsulates the packet into an Interest with the name of RP. Since COPSS reuses the FIB of NDN, the Interest packet is pushed to NDN engine (action ②). The packet will be forwarded as a normal Interest until it reaches the NDN engine on the RP router (actions ③-⑧). Since the RP is listening on the name it serves, NDN engine on the RP router forwards the Interest to the RP module (action ⑨). The RP module then decapsulates the packet and forwards the Publish packet to the COPSS module on the RP router (action ⑩). The Publication packet will then be forwarded according to the COPSS engine downstream (actions ⑪-⑬). The packet will not go through NDN engine any more.

3.11 Discussion

This section discusses some design considerations of COPSS.

3.11.1 The Use of NDN

As a content-centric network solution, this work chooses to design COPSS as an extension of NDN. NDN is chosen because it is one of the most recent design of ICN and there is an active research and implementation community working on different aspects of it, e.g., routing, caching, etc. The new forwarding engine model has also been implemented as a hardware by Cisco [96].

The COPSS design reuses some of the NDN functionalities like hierarchical name/CD processing, unicast routing (from the publisher to the RP), and the query/response in the two-stage communication. But the author argues that COPSS can be built on any existing ICN solutions like NetInf, MobilityFirst, etc., since the support that is needed by COPSS is provided by most of the ICN solutions. COPSS can even be a stand-alone network architecture that provides pure pub/sub capability. The forwarding engines would then need two data structures – FIB and ST.

3.11.2 COPSS as an Overlay

As a next generation network design, it is natural for COPSS to be deployed as an overlay – similar to IP network implemented as an overlay of the telephone networks in the early ages.

This provides an opportunity for the ISPs to use the existing facility to provide basic ICN functionality. As more users turn into the new network, the ISPs can then gradually deploy more ICN forwarding engines for scalability and efficiency. The incremental deployment solution of COPSS is described in §6.

The author argues that COPSS is designed to work on any network protocol including UDP, IP, Ethernet, regardless of the layer in the original *Open Systems Interconnection* (OSI) model. It can also work directly over the physical layer like copper, fiber, radio, etc. This enables COPSS to link more networks that use different protocols, e.g., an IP network with an Ethernet (that is not using IP) can exchange data via the COPSS "overlay".

3.12 Chapter Summary

This chapter presented the requirements and the design of COPSS – an efficient pub/sub architecture. COPSS enhances a popular representation of ICN (NDN) to provide pub/sub functionality with minimal changes. It can achieve timeliness by push-based communication, achieve efficiency by using multicast, and can also scale well to accommodate a large number of publishers, subscribers and CDs. COPSS users can also exploit the efficient information dissemination provided by NDN in two-step communication. To prove the feasibility of COPSS design, a possible implementation is also provided via a wrapping design on the existing open source NDN implementation.

Chapter 4

Application: Content-Based Twitter

Twitter is particularly an emblematic of a pub/sub environment. With the help of COPSS, the subscribers can go beyond the subscription from a particular individual. They can also subscribe based on the properties of the contents (e.g., keywords, publication time, etc.).

To show the efficiency and scalability, COPSS is evaluated using trace-driven simulations. The simulator is parameterized using the results of careful microbenchmarking of the open source NDN implementation and of standard IP based forwarding. The evaluations show that COPSS provides considerable performance improvements in terms of aggregate network load, publisher load and subscriber experience compared to that of a traditional IP network and pure NDN.

Contents

4.1 Communication Design

Since the content-based Twitter follows a typical pub/sub communication model, the network-layer communication design is relatively straightforward. The application groups the categories of contents into a hierarchical structure which follows the natural ontology, and each category is represented by a CD. The publishers and subscribers include related CDs in the publication and subscription. The network will then take care of the delivery of the data to subscribers who expressed the interests to it.

Two types of data transmissions are considered in this application: *1*) short data dissemination: when a publisher tries to publish a short data (140 characters in a typical Tweet), one-step communication (described in §3.7) can be used directly to provide timeliness and network efficiency, and *2*) large or policy-controlled data dissemination: if the data is big in size or the publisher wants to have a policy control on the data item, the publisher can use two-step communication (described in §3.9). The network efficiency and provider load will be compared when the publishers are using these two communication models.

4.2 Asynchronous Data Dissemination

It is natural for a subscriber to get offline (turn off the end-system or move to a different location). The application should enable the user to receive messages that were missed when they are offline (the asynchronous data dissemination). Furthermore, while trying to get the missed data, the user should not have to know who the publishers were, or even whether they are still connected to the network. It is quite reasonable because some sensors as publishers might be disabled/destroyed shortly after transmitting a warning.

Existing pub/sub paradigm usually introduces a third-party entity in the application layer (i.e., *brokers*), to store all the published messages. The brokers get all the messages from the publishers and deliver the messages to subscribers either by pushing or wait for the subscribers to pull. Whenever and wherever a subscriber joins the system, (s)he queries the brokers to retrieve the missed messages when (s)he was offline. Similar approach is adopted in the application but the pushing is left to COPSS.

In order to be scalable to a large amount of data, the solution enables the broker to be a set of collaborating, distributed servers (i.e., a *broker cloud*). Thanks to the name-based routing provided by ICN, each broker can serve a subset of the CD space and register prefix with the served CD in the name in the form of `/brokerName/CD`.

To receive all the messages in a served CD, a broker subscribes to that CD just like a

normal subscriber. It can then obtain a copy whenever a publisher publishes new messages. While storage space is a concern, and issues such as the content replacement policy on the broker are relevant, their solutions are likely to be similar to what has been adopted in a non content-centric, server-oriented information infrastructure.

4.2.1 Querying for Missing Messages

In order to query for missed messages, the solution requires a subscriber to record the ContentName (identity) of the last message (s)he received when (s)he was online. The broker cloud should also order all received messages based on their arrival using its local time. When a subscriber rejoins the system, (s)he queries the broker cloud with two pieces of information: *1*) the CD (s)he subscribed to as part of the ContentName in the Interest (request), and *2*) the identity of the last message received either in the request content or encapsulated in the ContentName. The ICN routers will forward the request based on the name to the dedicated broker. The broker has to look up in the storage for the required messages and send all the matched messages to the subscriber.

The following subsections describe how the solution addresses the scalability and reliability issues.

4.2.2 Scalability: Retrieving Missing Content

A subscriber may have subscribed to a very large number of CDs. The popularity of the CDs can vary dramatically from "disaster warning" (with very few messages) to "sports" (with huge amount of messages). When dealing with asynchrony, a subscriber coming back online would have to send a query for every CD that (s)he has subscribed to since it is impossible to predict which groups have had new content. With the magnitude of subscribers and CDs envisioned, such a pull-based approach for information every time a subscriber comes back online (or moves to a different point in the network) could result in a lot of traffic.

To reduce the query load that a subscriber generates and the corresponding processing overhead at the broker, the solution seeks to aggregate processing by grouping content across multiple CDs. Instead of querying for content related to individual CDs, the subscriber queries for a whole group. The tradeoff is that the subscriber might receive extra messages (false positives), which have to be eliminated at the receiving end-point. One possible aggregation function could be traditional hashing schemes (e.g., based on the CD string), but it has the disadvantage that their decision does not take the semantics related to CDs into consideration.

The hierarchical structure of CDs in COPSS allows subscribers and/or brokers to exploit the benefit of aggregation. It is believed that such hierarchy will help to minimize the retrieval overhead and in reducing false positives. E.g., a subscriber who subscribed to a large number of disaster warning CDs could request using a higher-level CD in the hierarchy (e.g., `/disaster`). It is particularly attractive when these updates are infrequent and the portion of false positive is small.

4.2.3 Scalability: Message Delivery

The scale of the subscribers envisioned is likely to lead to many offline users becoming online in a burst at peak periods (e.g., in the morning), resulting in a large burst of query traffic. This provides opportunities to optimize network traffic. The application tries to use markers in the message sequence to allow batch responses. E.g., assume a subscriber of a CD requests for content that (s)he missed since 9 pm and another subscriber of the same CD requests content that (s)he missed since 11 pm. Using a marker (i.e., 9pm, 11pm) to delineate the message sequence into batches allows the broker to multicast the overlapping message sequence between the subscribers. This approach can reduce both network and broker load.

4.2.4 Reliability: Possible Loss of Sequence

COPSS multicast messages may arrive at the subscribers and the broker cloud in different order. This can be caused by different latencies from different publishers. Thus, if the broker simply provides the subscribers with the messages received after the matched message in the log, some published information may be missed.

This problem is solved by requiring the broker cloud to group all the messages in its log into different *windows*. Let the size of this window be n, indicating a set of consecutively received messages at the broker. Upon recept of a query, the broker finds the *target window* that contains the matched message. Then messages related to the queried CD belonging to the same window are sent to the subscriber. By sending these extra messages within the *target window*, messages that may have been received out-of-order can be included and delivered to the subscriber. The (application of the) subscriber would be responsible for duplicate elimination. The subscriber would also maintain a local *window* that stores the messages that the subscriber received as soon as (s)he came online. This ensures that once the broker starts sending messages that the subscriber has already received, the subscriber could stop sending the query. Note the parameter n has to be tuned to make a good tradeoff of delivering unnecessary information to subscribers and network load versus the success

Table 4.1: Forwarding performance of COPSS, NDN and IP (95% CI).

(in μs)	**COPSS**	**NDN**	**UDP**
Subscription/Interest/200B	2,679.1(575.13)	2,295.6(1106.42)	37.4(8.21)
Publication/Data/4096B	82.8(5.60)	2,135.4(876.04)	75.2(16.31)

probability of recovering all the messages when the subscriber is offline, depending on the message frequency and user online/offline pattern.

4.3 Evaluation

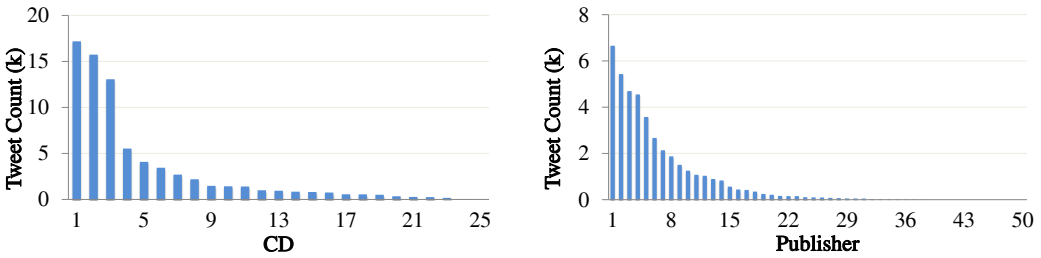
This section shows how COPSS can achieve improved performance compared to a pull-based NDN implementation as well as IP multicast. To enable a large scale evaluation, this work builds an event-driven simulator written in *C#*. A RocketFuel topology and traces collected from Twitter [97] are fed to the simulator to represent the real-world environment. The forwarding latencies of COPSS, NDN and IP are taken from a careful micro-benchmarking.

Multiple scenarios are used to evaluate the performance of the solutions: *1*) forwarding performance with both one-step and two-step communication, and *2*) efficiency of subscribers retrieving missing messages

4.3.1 Microbenchmarking

Microbenchmaring is performed to study the processing overhead of NDN and COPSS compared to a pure IP-based forwarding (albeit recognizing that the functionality offered by an ICN node is significantly different). CCNx [8] v0.4.0 (as a version of NDN engine), COPSS implementation and a user-level IP/UDP forwarding are compared in the benchmark. The measurements were performed on a Linux 2.6.31.9 machine (3.0 GHz Intel E8400 CPU, 4GB Memory). Note that NDN and COPSS are currently implemented as a user-space overlay, using UDP or TCP encapsulation for exchanging packets between different nodes. To have a reasonably fair comparison, a user-level UDP forwarding is used in the comparison. The evaluation also takes the packet size into consideration: 200-byte UDP packets compared to NDN Interests and COPSS Subscriptions; 4096-byte UDP packets compared to NDN Data packets and COPSS Publications. The time between the incoming and outgoing instants of each packet is measured using Wireshark [98].

The result of the benchmark is shown in Table 4.1. To achieve the functionality of name-



(a) # of tweets per CD

(b) # of tweets per pub

Figure 4.1: Dataset information.

based routing, NDN routers are about 50 times slower than UDP forwarding. COPSS routers need to use FIB lookup in NDN engine for Subscriptions and therefore is as slow as NDN for Subscription forwarding. But Publication forwarding in COPSS only involves ST matching and therefore it is faster compared to Subscription forwarding. Although a hardware-level implementation would be able to accelerate the forwarding performance, the requirements placed on an NDN router is still higher than that placed on IP router and therefore, the forwarding latency will surely be longer.

4.3.2 Large Scale Trace-Driven Experiments

4.3.2.1 Data Trace and Experimental Setup

This work uses a Twitter [97] data trace on technical topics obtained from the public Internet during a one-week period in 2010, which totaled 68,695 tweets sent by 38,481 users.

25 hot keywords such as `iphone`, `ipad`, `blackberry`, `smartphone` are selected as CDs. The tweets without any of these 25 keywords are filtered out and the remaining 41,613 tweets from 22,987 users (4.13 tweets/minute, up to 48 tweets posted at the same time) are used as the simulation input. Fig. 4.1a shows the number of tweets associated with each CD respectively. Additional hot keywords from the the subset are identified to obtain sub-CDs for the selected 25 CDs. Each 1^{st}-level CD has 1–25 sub-CDs and a total of 407 distinct CDs are eventually selected.

To make the publishers of the system tweet more frequently, the 22,987 users are assigned to 50 publishers by a power-law based hashing. Fig. 4.1b shows the number of tweets per publisher.

Without the means of inferring CD-Subscriber relationship from the data trace, this work

Figure 4.2: RocketFuel topology (Exodus, AS-3967).

assumes that popular CDs have more subscribers (based on [99]). Thus, subscriptions to CDs are generated following the CD-Tweet distribution. Subscribers can subscribe to multiple CDs. But a subscriber who has subscribed to a higher level CD will not subscribe to its sub-CDs. To simplify the simulation, in two-step communication, subscribers query the data as soon as they get Publication.

Rocketfuel [100] backbone topology (Exodus, AS-3967, shown in Fig. 4.2) is used as the core routers in the simulation. Besides the 79 core routers, 200 edge routers are added with each core router having 1-3 edge router(s). The simulations randomly distributed 50 publishers, and uniformly distributed the subscribers (varying from 200 to 600) on the edge routers. The link weights between the core routers were obtained from the topology and interpreted as delays (*ms*). The delay between each edge router and its associated core router is set to $5ms$; the delay between each the host and its associated edge router is set to $10ms$.

To study the impact of different solutions on the network, *aggregate network load* is used as an important metrics to show the amount of traffic in the network. It is calculated as:

$$\sum_{i=1}^{packetCount} packetSize_i \times hopCount_i. \tag{4.3.1}$$

I.e., when a packet with size $1kB$ is sent from host A through router R to host B, $2kB$ is added to the aggregate network load. For a fair comparison, COPSS packets are encapsulated into UDP packets when transmitted over an IP underlay. The encapsulation overhead is therefore the same as in NDN.

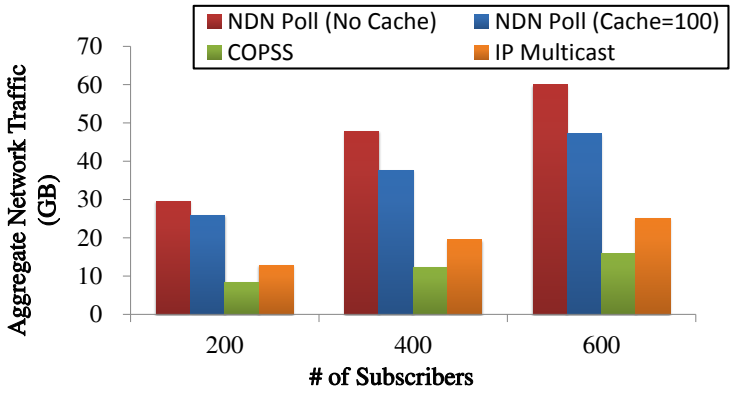


Figure 4.3: Aggregate network load (of NDN vs. COPSS one-step and IP Multicast).

4.3.2.2 Performance of COPSS One-Step, NDN and IP Multicast

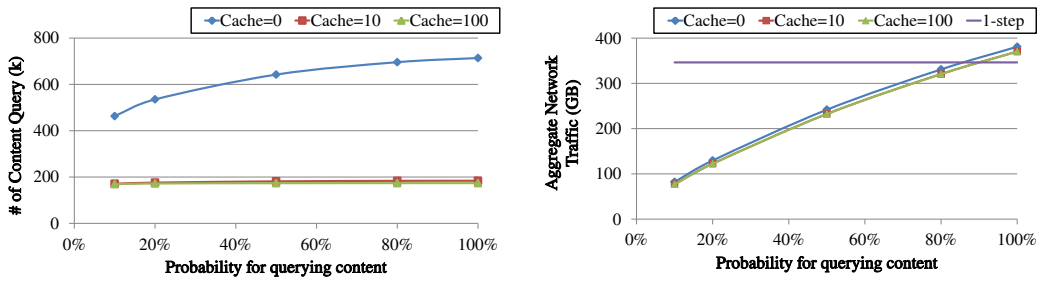
Fig. 4.3 illustrates the aggregate network load driven by the application over the standard pull-based NDN (both without a cache as well as a cache of 100 packets), using COPSS one-step communication as well as native IP multicast.

In the result, COPSS consumes less network load because of its design of a content-centric push mechanism that improves upon NDN by adding the multicast capability. In addition: *1*) COPSS clients do not need to poll. With NDN, subscribers have to poll the server once every 30 minutes, which introduces additional network overhead. *2*) Network traffic is further reduced because of multicast with COPSS even compared to the use of caches in NDN. And *3*) The notification latency is much shorter in COPSS ($\sim$80*ms*) compared to NDN ($\sim$15*min*, half of a polling period).

Moreover, COPSS performs better than native IP multicast, because of the hierarchical CDs. This results in fewer messages being transmitted to reach the subscribers of different CDs. Note that the aggregate network load due to NDN and IP multicast increases linearly with the number of subscribers. With COPSS, the increase in subscribers results in only a marginal increase in the aggregate network load since the data is only duplicated very close to the subscriber (in the optimal case at the subscriber's first hop router).

4.3.2.3 Performance of COPSS Two-Step Communication

This simulation evaluates the performance of COPSS in transferring large volume content Large contents that are 128 times larger than the original Tweets are created from the trace.



(a) Publisher Load (b) Aggregate Network Traffic

Figure 4.4: COPSS in two-step mode.

Table 4.2: Broker load vs. router cache size.

Cache Size	**0**	**10**	**100**
Broker load (# of query)	895.953	892.714	778.270

The load on the publisher is evaluated (see Fig.4.4a) by counting the number of queries for data that reach the publisher for varying cache sizes (0, 10, 100). COPSS is able to leverage the benefits of NDN, by first pushing snippets to subscribers who then immediately query the publisher if they are interested in the content. A cache size of 10 packets is sufficient for the simulation to reduce the load on the publisher significantly.

Fig. 4.4b illustrates the aggregate network load when a varying portion of subscribers request for the full content on receiving the snippet. The delivery using COPSS one-step communication for the whole content is also shown as a reference. When a small percentage of users request for the content, substantial network resources are saved by adopting the two-step mode. However, when the number of subscribers requesting for the content reaches more than 85%, the two-step mode is more expensive because the extra snippets in the first step.

4.3.2.4 Performance of COPSS Asynchronous Dissemination

Based on the preliminary analysis on the twitter data trace available, the simulation synthesizes the users' offline/online pattern as follows: everyday, about 75% of all subscribers randomly go offline between 2-3am, and then go back online randomly between 10-11am. Each of the remaining users randomly chooses offline period and time with an average offline duration of 20 minutes. The behavior is created to study the impact of a large portion of the subscribers coming online at nearly the same time. Table 4.2 shows the broker load with different cache sizes. As the cache size increases, the number of queries reaching the broker reduces. The cache hit rate is boosted by the division of content based on the markup message and the grouping of subscribers into higher level CDs when appropriate.

Table 4.3: The FIB entry created due to Server/RP and publishers of the specific content.

	NDN-Pull		**COPSS (1-step)**		**COPSS (2-step)**	
Node type	Pub	Server	Pub	RP	Pub	RP
FIB entries	13,950	279	0	278	13,950	278

4.3.2.5 Additional Observations

Table 4.3 shows that in the case of NDN-pull and COPSS (2-step), the publishers need to be visible and therefore have to propagate their entry throughout the network. In the case of COPSS (1-step), since it is an RP based multicast, the publishers need not propagate their entry and only the RP propagates the entry to all the (278) COPSS enabled routers. It shows that COPSS (2-step) behaves in a manner similar to NDN-pull with regards to FIB propagation whereas in the case of COPSS (1-step), the size of the FIB in the network is considerably lower.

4.4 Chapter Summary

This chapter presents and evaluates a content-based Twitter which requires a typical pub/sub environment. To accommodate the users getting offline, the application uses dedicated servers (brokers) to store the messages for users. With the help of NDN, the brokers can be scalable and efficient.

Trace-driven simulations are used to evaluate the benefit of COPSS in such applications. COPSS reduces the aggregate network load and the publisher load significantly. The additional ICN layer processing with COPSS compared to IP multicast is relatively small, achieved by considerable efficiency in avoiding duplicate and unnecessary delivery of content to subscribers. Because of the inherent pub/sub capability, COPSS provides better timeliness, scalability and efficiency compared to NDN.

Chapter 5

Application: Gaming

Information-Centric Networking provides substantial flexibility for users to obtain information without knowing the source of information or its current location. With users increasingly focused on an online world, an emerging challenge for the network infrastructure is to support *Massively Multiplayer Online Role Playing Game*s (MMORPGs). Currently, MMORPGs are built on IP infrastructure with the primary responsibility resting on servers for disseminating control messages and predicting/retrieving objects belonging to each player's view. Scale and timeliness are major challenges of such a server-oriented gaming architecture. Limited server resources significantly impair the user's interactive experience, requiring game implementations to limit the number of players in a single game instance.

This chapter describes a distributed communication infrastructure G-COPSS with the help of COPSS to enable efficient decentralized information dissemination in MMORPGs, jointly exploiting the network and end-systems for player management and information dissemination. G-COPSS aims to scale well in the number of players in a single game, while still meeting users' response time requirements.

A microbenchmarking on the processing involved in managing game dynamics is carefully performed using a simple game with a hierarchical map. The same game using NDN and IP server infrastructure is also evaluated as a comparison. This work then simulates an application that is particularly emblematic of MMORPG – Counter-Strike – but one in which all players share a hierarchically structured map. Trace-driven simulations are used to evaluate the timeliness and scalability of G-COPSS. The results show that G-COPSS provides orders of magnitude improvement in update latency and a factor of two reduction in aggregate network load compared to a server-based implementation.

The key contribution of G-COPSS is to provide an efficient, distributed communication infrastructure for MMORPGs. In more detail, the contributions include:

- G-COPSS is designed as a decentralized gaming platform that leverages the content-centric pub/sub multicast capability provided by COPSS, with added features that both exploit the knowledge of and optimize for the gaming environment.
- Current games use map partitioning to help scene rendering and update dissemination. G-COPSS enhances this with a *multi-layer hierarchical map* functionality. Players can have different levels of visibility of the game environment based on their position and altitude. This allows them to only send/receive updates pertaining to that vision.
- G-COPSS provides additional features to enhance the experience for players moving between regions in the game map and for offline players that come online.
- The performance of G-COPSS is evaluated by a micro-benchmark and compared to an NDN and an IP-server based solution. Moreover, large-scale trace-driven simulations adapted from Counter-Strike is used to demonstrate the performance gains in terms of aggregate network load, server load and improved player experience.

Contents

5.1 Motivation

*Massively Multiplayer Online Role Playing Game*s (MMORPGs)[1] are increasingly popular. This is not only because of their attractive structuring and creative scenarios, but also because they allow for a large number of players to participate in the same game. World of Warcraft, Counter-Strike and Second Life are examples of such games. Supporting them at scale, however, is a significant challenge. These games have high interactivity (and therefore need very low network latency), since every action an individual player performs needs to be communicated. A player also needs to be informed of all the related players and their positions/actions. Players react based on the 'current' environment and the cumulative actions of all the players.

These multi-player games require a persistent view of the world and are usually managed by a dedicated server (e.g., one that is hosted by the game publisher). The game environment in many such server-based MMORPG is such that it is divided into regions with different groups of players having varying amounts of visibility. Players publish their actions to a (centralized) server which then forwards the updates to the relevant players based on the player's visibility region. The load on the server and communication needs for player management can be significant. Processing and I/O at the servers as well as the network bandwidth can be a bottleneck. The communication structure for these games requires the flexibility of supporting a very dynamically changing set of participants. A player potentially needs to be able to send to, and receive from a set of participants that it does not even explicitly know of. Distributed approaches that seek to overcome the performance bottlenecks in a server-based MMORPG need to accommodate these needs. Although P2P-solutions seek to relieve the servers from the heavy computation workload, the need to provide the flexible communication framework of sending and receiving to a dynamic (and possibly unknown set) of participants poses difficult challenges even for a P2P-oriented environment.

This work realizes that the game-communication is eminently satisfied by the content-centric pub/sub – players publish updates to an area/object (content) without regard to who is supposed to see it; at the same time, players subscribe to the areas/objects (again, content) they can see, without knowing who else is trying to modify them. The fundamental capability of disseminating information based on content - without the need of knowing who to send it to or who to query for information - makes the content-centric communication fabric very suitable for gaming applications. With the use of an appropriate interface, users can select and filter the information desired, irrespective of the publisher of this information. G-COPSS is therefore proposed to fine-tune and optimize COPSS to support the specific needs of a game environment.

[1] `http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game`

5.2 Overview

G-COPSS is designed as a decentralized content-centric communication framework on COPSS to satisfy the specific needs of a typical MMORPG environment. It utilizes predominantly push-based multicast provided by COPSS to ensure that players receive timely updates. Using such a content-centric solution overcomes many of the limitations of a server-based or a P2P-based solution, in terms of scalability, responsiveness as well as the need to know the identities of the players and objects that an individual player has to communicate with.

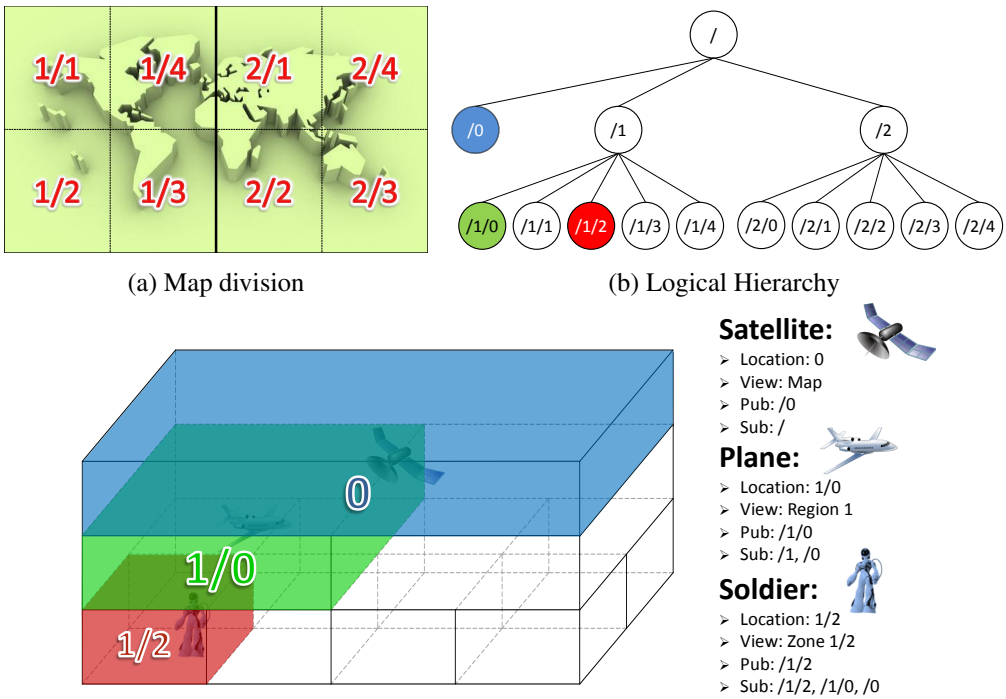
A key challenge to information-centric communication is a content naming structure that is sufficiently general to accommodate diverse needs. However, in the case of gaming environments, the solution can take advantage of the typical schema and the natural partitioning of a game map. This work generalizes the game map by considering the notion of *hierarchical game maps*. A naming hierarchy is formulated suitable for even complex games that may include a hierarchical game map partitioned in arbitrary ways.

A game specific optimization that helps players get information when moving from one partition to another is also provided.

5.3 Gaming Hierarchy and Nomenclature

There are multiple reasons behind game designers partitioning the online game map into distinct regions. One is to have the natural representation of the game environment and to manage/limit the visibility of the players. Also, for reasons of implementation, such as efficient broadcast of updates and load distribution on the servers, the game map may be partitioned. G-COPSS takes that optimization a step further by considering a multi-layer hierarchical relationship between the various areas in the environment, as shown in Fig. 5.1, where the game map is broken up with the *map* layer above the *region* layer at the top and the *zone* layer at the bottom of the figure. Figure 5.1a shows a world map which is first divided into 2 regions (marked 1 and 2) and each region is further divided into 4 zones (marked $1/1 - 2/4$).

Optimization tools for map partitioning can be modified and used for partitioning the maps. The game designer, on finishing the map definition and creating-assigning objects to maps, can use such optimization tools to calculate the hierarchical partitions based on the physical features and the object heat level in each partition. The final output of such tools can be used directly by the user programs.



(a) Map division

(b) Logical Hierarchy

(c) Hierarchical subscription

Figure 5.1: Hierarchical map partitioning.

To match the information generated by a producer to the consumers interested, it is important to have an easily understandable nomenclature. Here, G-COPSS uses the CDs provided by COPSS. The schema used for the partitioning of the map forms the schema for the naming hierarchy as well, and the zones/areas in the game map are mapped to hierarchical CDs.

Players subscribe to the groups that represent the areas that they are currently involved or located in and are therefore able to publish and subscribe to those areas. The name hierarchy created for the map is shown in Fig. 5.1b. Therefore a player who is flying above region 1 will have subscribed to the CD `/1` and will therefore be able to see the updates sent from the players below in all the zones $1/1 - 1/4$. G-COPSS in fact allows map designers to divide the map into arbitrary layers, but for simplicity, only 3 layers are used as examples in this part. CDs form the naming framework corresponds to the hierarchical location map in G-COPSS.

When a player at the lower zone layer wants to see the updates sent by a person flying over the region above, it would result in high overhead to subscribe to CD `/1` since (s)he

would then receive updates from all the players belonging to the zone-layer of `/1`. To address this issue, G-COPSS introduces the concept of having every zone/area represented also as a leaf node. This allows finer-grained publication and subscription. To perform this, G-COPSS creates a `/0` sub-CD for every non-leaf CD in the hierarchy, i.e., `/0` for the top map layer (`/`) and `/1/0` for the region `/1`. These `/0`s are used to represent the areas above, e.g., where planes are flying (shown in the 3D partition in Fig. 5.1c) so that every area in the game world is represented by a leaf node in the logical hierarchy. E.g., the green area above $1/1$ to $1/4$ is represented by CD `/1/0`, and the blue area above $1/0$ and $2/0$ is represented by `/0`. This therefore allows a player to subscribe to `/1/0` and receive updates sent by the player flying above, like the plane.

G-COPSS makes the basic assumption that all players have access to the common game-map via the game client that was downloaded apriori and are therefore aware of the naming convention and the grouping of the zones/areas.

5.4 Communication Design

The hierarchical CDs described above form the basis of the multicast-based communication model wherein virtual RPs are assigned to act as the core of the multicast trees towards the subscribers. Updates generated by players are forwarded along the subscription tree to all players subscribed to receive content in the same region/zone. The RPs handle one or more CDs based on the expected load and COPSS is able to dynamically add and delete RPs based on the demand (as described in § 3.8). This section describes how G-COPSS adapts COPSS to provide basic gaming data transmission.

5.4.1 Update Dissemination in Gaming

According to the naming hierarchy, a player can send an update using a leaf CD representing the area that contains an object (including himself/herself) (s)he has modified. The player can also subscribe to the leaf CDs that represents the areas (s)he has visibility into (*Area of Interest* (AoI)). Furthermore, subscriptions to CDs can be aggregated to a higher level in the hierarchy. According to [64], almost all of the packets in a gaming application are under 200 bytes. Therefore the one-step model of COPSS, where the data is directly pushed to the subscribers, is used by G-COPSS to disseminate update/control messages. Moreover, in a game, a player is a publisher as well as a subscriber, and therefore G-COPSS allows players to publish and subscribe using different CDs according to the game semantics.

5.4.1.1 Hierarchical Publishing

Players need to publish the updates they make to all interested-players (interested-players are those that view the same acAoI/objects) using a core-based multicast tree structure, the RPs being responsible for the groups associated with the corresponding CDs. E.g., if a player moves a satellite in the blue area in Fig. 5.1c, (s)he will send the message to the RP serving the CD `/0`; if (s)he shoots at a plane in the green area, (s)he will disseminate the information using the CD `/1/0`; and if a soldier is moving in the red area, (s)he will disseminate it using the associated CD `/1/2`.

5.4.1.2 Hierarchical Subscriptions

Assume that according to the semantics of the hierarchical map, players are able to see all the updates below and vice versa. Therefore, a player will subscribe to the area (s)he is in and all the `/0`s along the hierarchy. E.g., a player standing on $1/2$ should subscribe to `/0`, `/1/0` (the `/0`s along the hierarchy) and `/1/2` (the area (s)he is in). This allows him/her to see the units standing on $1/2$, the planes flying over zone 1, and the satellite at top (so that (s)he will not be shot without knowing who did that). Likewise, a player flying over 1 will see the units standing on $1/1 - 1/4$, those flying over 1 (area $1/0$) and also those on top (area 0). Note that the CDs of `/1/1` to `/1/4` and `/1/0` could be aggregated to `/1` implying that the player can therefore subscribe to `/0` (the `/0`s along the hierarchy) and `/1` (the area (s)he is in).

5.5 Player Moving Support

It is natural for a player to move from one sub-world to another as well as from being offline to coming online. Similar to the offline-user support provided by content-based Twitter (§4.2), G-COPSS provides the following additional facility to support effective dissemination for player movement.

When a player enters (or approaches) a new sub-world, (s)he should be able to see the current status i.e., the snapshot of the sub-world. G-COPSS achieves this with the help of a decentralized set of servers that behave as brokers and maintain an up-to-date snapshot of the various zones that they are responsible for by subscribing to them. The broker is only required to manage the snapshot of the sub-world instead of all the existing updates in the sub-world. Therefore, it only subscribes to the leaf CDs representing its serving area and calculates snapshots on receiving updates.

G-COPSS provides several options for a player to retrieve the latest snapshot of the area (s)he is interested in.

5.5.1 Query from Broker

This approach exploits the query/response functionality provided by NDN. When a player moves/teleports to another sub-world, (s)he queries for the snapshot of that sub-world (but only the part (s)he hasn't seen prior to moving) according to the name of that area, e.g., `/snapshot/1/3`. NDN will forward the query to the responsible broker(s) which in turn return the snapshot.

5.5.2 Peer Assisted Request

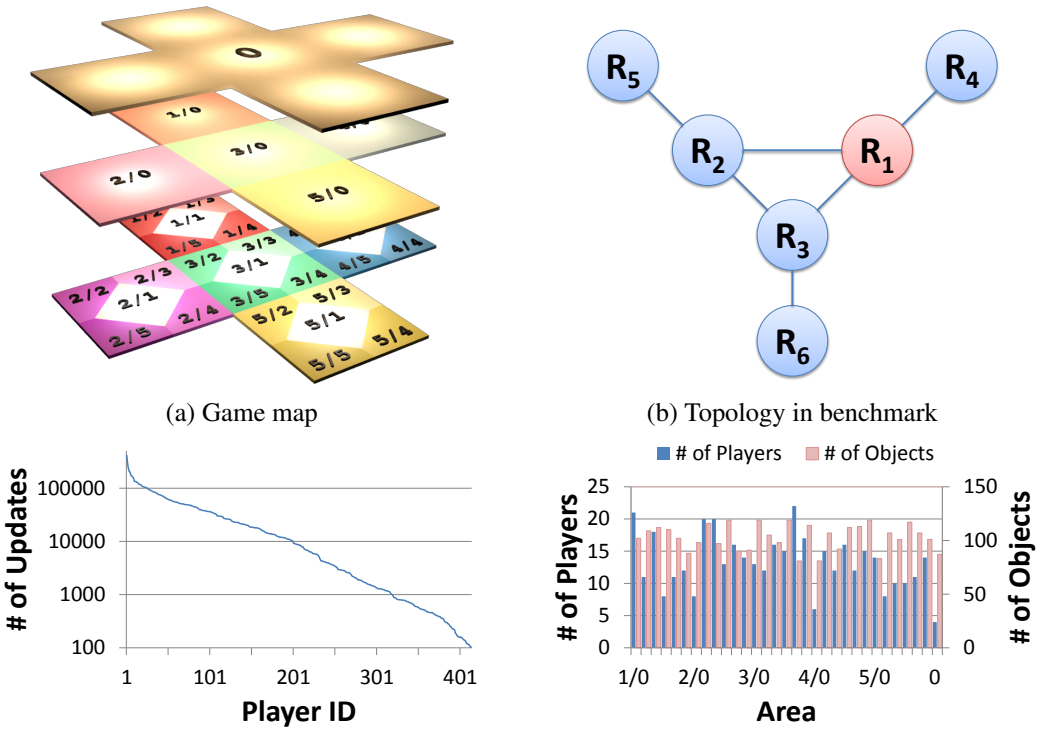
To prevent the broker from being the bottleneck of the dissemination of snapshots, nearby players who happen to be in the target sub-world can also help. To provide such help, the player (T) can register a game-specific FIB entry (e.g., `/snapshot/1/3`) to the nearby routers (using *Time To Live* (TTL)). When requesting for a snapshot, the moving user's (M) requests will be forwarded towards nearby users by NDN directly.

One concern of such peer-assisted request is data integrity. How can M ensure that the snapshot (s)he got from T is not a cheat? To address this concern, G-COPSS suggests every packet from the broker should be signed by the game provider. The integrity of each packet can be verified. Addition to that, T should not send the latest snapshot which is calculated by its own game client (and not signed). Instead, T should send the latest snapshot (s)he received from the broker (signed), and all the updates happened in between (which are also sent and signed by the broker).

5.5.3 Cyclic-Multicast

Instead of querying for the latest snapshot, the new player subscribes to a multicast group that disseminates the snapshot in a cyclic manner. The responsible broker is the only publisher of this group. It starts multicasting on receiving the first Subscription packet and stops when there is no subscriber any more.

This alternative will be helpful when players move in a group, i.e., several players move into a new area at the same time. However, the multicast packets sent after the last player receives the snapshot (between when the Unsubscribe packet is sent from the player to when it is received by the broker) will be wasted. In many game scenarios, it is quite common for



(a) Game map

(b) Topology in benchmark

(c) # of updates per player

(d) # of players/objects per area

Figure 5.2: Experiment Setup.

a team or group of players to move at roughly the same time to a different area resulting in a flash or burst of requests for snapshots. In such cases, the cyclic multicast is very effective since it does not introduce additional query overhead and also ensures that the broker does not become a bottleneck.

5.6 Evaluation

The performance of G-COPSS is evaluated and compared with that of the traditional IP server based solution and the NDN solution in both test-bed and simulator. The test-bed microbenchmarks real G-COPSS implementation on the computation overhead and queuing effect. The effects of bandwidth and congestion related latency issues are not considered, since its effect would be experienced by all the candidate solutions and will be studied in §7. The simulator, which is parameterized by microbenchmark results, uses real-world topology and larger trace to demonstrate the capability of G-COPSS in a much larger environment.

In both tests, players share a global game-map shown in Fig. 5.2a. The game map is converted to a hierarchical map as follows: first it is divided into 5 regions (marked 1–5); second each region is further divided into 5 zones ($1/1$–$5/5$). This results in 31 leaf CDs in the hierarchy: 25 in the bottom layer (*zones*) marked $1/1$–$5/5$, 5 in the middle layer (*regions*) marked $1/0$–$5/0$ and 1 in the the top layer (*world*) marked 0. Each area is further divided into objects representing things that the players could change. A player is able to see and modify these objects depending on his/her location in the game and the hierarchy of the area (s)he belongs to.

5.6.1 Microbenchmarking

G-COPSS is implemented and evaluated on a lab test-bed and compared with the other two approaches. The clients of the game uses the map depicted in Fig. 5.2a. 62 players participated in the game with 2 players in each area. They can modify any object in their AoI and all the updates are translated into the respective CDs before being sent out. To make a fair comparison, a 1-minute trace is generated beforehand which is composed of publish records like $\{time, playerName, CD, Content\}$. In this time frame, the publishing frequency of every player ranged from once every $100ms$ to $500ms$ which resulted in a total of $12,404$ publish events. The publication size ranged from 50 to 350 bytes.

The lab test-bed has 6 Optiplex 755 SF systems as routers, running Ubuntu 10.10 and using CCNx v0.4.0 and COPSS implementation. One PowerEdge T300 system running Ubuntu Server 11.10 is used to emulate all the 62 clients and the server in an IP-server scenario. The network topology is shown in Fig. 5.2b. R_1 serves as the RP in G-COPSS. The server is also linked to R_1 in IP-server test. Players are uniformly distributed across the routers in the network.

NDN solution uses the method described in VoCCN [101] and assumes that players are managed using the system proposed in ACT [102], so that players know each other and their current position. Every player queries all the possible players for the updates in the AoI. Two optimizations are used: pipelining and accumulated updates. For pipelining, the game logic allows a player to have a set of at most N ($N = 3$ in the benchmark) queries outstanding at any time. For update accumulation, instead of sending a response for each update/action, the game clients send a response every t ms. All the updates within the t ms will be put into one packet. There is a tradeoff: t is large enough, more updates are included which saves some bandwidth, but the update latency will be longer. If t becomes too small, players can see the updates immediately but incur a lot of overhead.

In the server-based solution, the game client creates data packets containing the following fields: source address, destination address and payload. All the routers use an application-

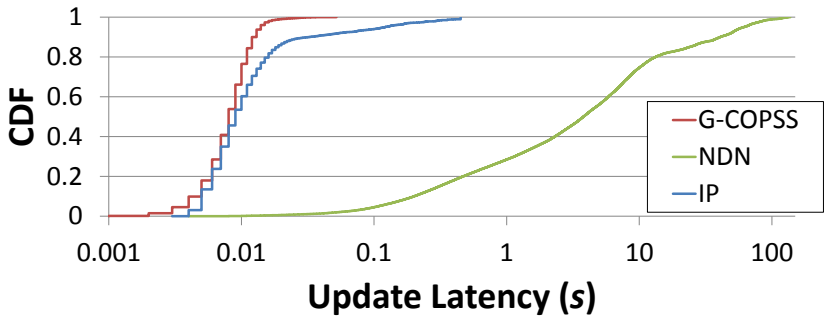


Figure 5.3: Update latency CDF of G-COPSS, NDN and IP Server.

level forwarding engine for a fair comparison, forwarding packets based on the destination address. All players send updates to a server. On receiving an update, the server decides whom to send it to, and sends the update using unicast.

5.6.1.1 Microbenchmark Result

The update latency of different solutions is shown in Fig. 5.3. In the evaluation, the average update latency experienced by players in G-COPSS is 8.51*ms*, compared to 25.52*ms* in the IP server-based solution. The latency here is smaller than the real world situation because the test-bed machines do not incur much latency on the wire; almost all the latency caused is due to the packet processing at the routers. In G-COPSS, all players experience an update latency smaller than 55*ms* whereas about 8% of players experience an update latency over 55*ms* in the IP server case. The server becomes the bottleneck of message dissemination although IP routers are more efficient than the G-COPSS routers according to the benchmark in the content-based Twitter (§4.3.1).

For the NDN solution, even with the optimizations provided, the average update latency is over 12*s* and the largest latency experienced reaches 150*s*. This high latency is due to the heavy workload on NDN routers caused by huge amount of queries initiated by every player trying to request for the next possible update in AoI (including the refreshing of the queries). The packet loss caused by the workload worsens the situation. Therefore, this work conjectures that the VoCCN based NDN solution might not work for large-scale pub/sub systems like games.

5.6.2 Large Scale Trace-Driven Experiments

To further evaluate the performance of G-COPSS in large scale scenarios, a simulator is developed with a game model derived from the data trace of the CS game [103]. Since

the query/response-based NDN (which is already optimized based on the state of the art [101,102]) still experiences large delays (as shown in §5.6.1), the NDN solution is omitted in the rest of the evaluation and G-COPSS is compared directly to the IP server-based solution.

5.6.2.1 Data Trace and Experimental Setup

The simulation uses a CS data trace obtained during the peak period of one day [103]. It consists of a Wireshark [98] trace collected on a busy CS server in a $7h05m25s$ period, which totaled 20 million packets sent to and from the server by $32,765$ different addresses ($59,294$ different address:port pairs). Since the packet headers captured did not contain game-related information, and is for a server based game, following operations are performed to filter out packets that represent a decentralized game: *1*) discarded all packets sent from the server since the simulation only takes the user updates, *2*) discarded packets with address:port that send less than 100 packets to obtain the trace of established connections (for clients who are really playing the game), excluding the many connection attempts which were used for clients to measure the RTT to the server, and *3*) each unique address is represented to represent a unique player. This resulted in a data trace consisting of 414 unique players and $10,686,950$ packets (updates) in the same period of time as the original trace. Fig. 5.2c shows the number of updates performed by the different players. The 414 unique players are in the same game-world shown in Fig. 5.2a with each area containing between 4 and 20 players. Furthermore, as shown in Fig. 5.2d, each area contains 80 to 120 objects. Note that this setting is more complicated than the original CS since it has more than 400 players in a same game world while CS allows only 22 players.

The Rocketfuel [100] backbone topology (AS 3967) is used as the core routers. 200 edge routers are added to the 79 core routers, and each core router is connected to 1-3 edge router(s). 414 players are uniformly distributed on the edge routers. The link weights between the core routers were obtained from the topology and interpreted as the delay (ms). The delay between each edge router and its associated core router is set to $5ms$; the delay between each of the host and its associated edge router is set to $10ms$.

5.6.2.2 Update Message Dissemination for Online players

This section evaluates the performance of G-COPSS in disseminating game updates compared to the traditional approach of using an IP network with servers. For simplicity, players in this simulation remain in their starting area and the events (from the trace) generated are related to the objects within their AoI.

The first 100,000 update packets from the event trace is used to evaluate the average

Table 5.1: Performance of G-COPSS and IP-server with different # of RPs/Servers.

	# of RP/Server	Update Latency (*ms*)	Network Load (*GB*)
G-COPSS	1	47,680.29	5.59
	2	558.18	5.66
	3	94.89	5.58
	Auto	106.76	5.61
IP Server	1	249,679.70	9.74
	2	71,991.92	10.00
	3	21,448.17	9.62

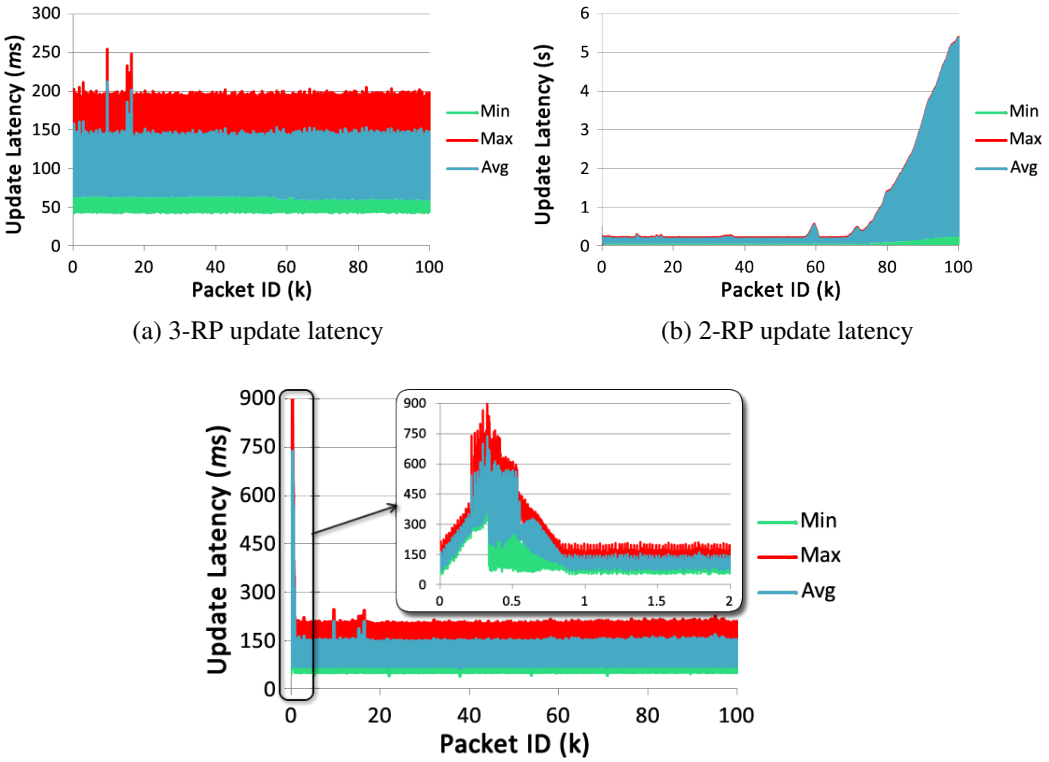
latency incurred in delivering an update from the publisher (player performing the update) to all the other subscribers (players subscribed to the CD) for different number of RPs and servers. The average update frequency observed in the event trace is about 2.40*ms*. In the simulations, an RP's processing time (including FIB lookup, packet decapsulation and ST lookup) is set to 3.3*ms* (based on benchmark measurements), and the server processing time is around 6*ms* (also based on the microbenchmark result, factoring in some additional processing for other game related function like location translation and collision detection.)

Table 5.1 shows that the update latency for both the server and the G-COPSS approach. The automatic RP balancing functionality in COPSS is disabled at first. In the case of 1 or 2 RPs, the latency is high due congestion at the RPs. Adding another RP mitigated the congestion: using 3 or more RPs, no congestion is observed. Fig. 5.4a shows the minimum, maximum and average update latencies of every update in the 3-RP case. The update latency is below 200*ms*, well below the generally considered acceptable latency for such games (i.e., between 300*ms* and 1 second [104]).

Compared to the G-COPSS approach, the server based approach is much worse, resulting in a very significant, unacceptable update latency. It is also partly due to the need for IP servers to disseminate the information via unicast to all the individual subscribers whereas G-COPSS is able to perform a hierarchical CD based multicast. Moreover, multicast in G-COPSS reduces the network traffic significantly (almost by half).

The work then tries to understand the detail for the congested cases. With 2-RPs, the latency of G-COPSS (shown in Fig. 5.4b) indicates that the RPs see congestion after $70,000$ packets and the latency increases dramatically. This is caused by the traffic concentration and queueing at the RPs.

With 3 RPs or 3 Servers the scalability is evaluated with varying the number of players in the network from 50, 100, ..., 400. Fig 5.5 shows the response latency and aggregate network load. Note that 3 servers are sufficient to support about 200 players playing at the



(a) 3-RP update latency

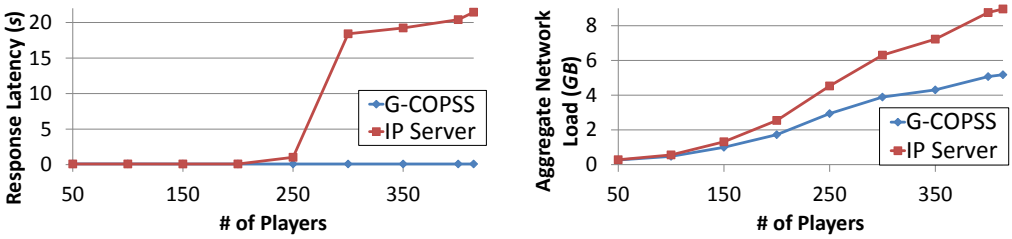
(b) 2-RP update latency

(c) Automatic-balancing update latency

Figure 5.4: Traffic concentration elimination.

same time. Fig 5.5a shows that the response latency observed in the G-COPSS remains low regardless of the increase in players, reflecting the advantage of a multicast based solution. However, in the case of the IP-server solution, the response latency increases rapidly beyond a threshold of the number of players (approximately 250 players) as the IP servers become a bottleneck.

However, even with G-COPSS, it is difficult to predict how many RPs would be required. A hot spot in the game (e.g., a lot of players in one area or a lot of player activity) can result in traffic concentration and queueing. Automatic RP balancing is evaluated in this scenario. Fig. 5.4c shows the detail of the update latency in a 414 player simulation. The latency for the first 2,000 packets are enlarged to show the further detail. The COPSS routers divided and moved the CDs to additional RPs twice when queuing was detected. The automated RP balancing algorithm chose 3 RPs, and as shown in Table 5.1, the performance of the automatic RP balancing is close to the manually selected 3-RP solution. With an increasing number of players, automatic RP balancing can further split the CDs to avoid congestion.



(a) Response Latency

(b) Aggregate Network Load

Figure 5.5: Performance of G-COPSS and IP server with different # of players.

Table 5.2: Performance of IP-server (6 servers), G-COPSS (6 RPs).

	Update Latency (*ms*)	**Network Load** (*GB*)
IP Server	90.17	1,046.62
G-COPSS	104.41	594.09

The whole event trace is then used to compare the performance of the IP-Server (with 6 servers) and G-COPSS (6 RPs) in this simulation. Table 5.2 shows that with the use of hierarchical CD based multicast, G-COPSS consumes the less network load due to the content-centric multicast.

5.6.2.3 Message Dissemination for Players Moving

This section evaluates the convergence time required for a player who has moved from another area to obtain the current snapshot at the new location. Due to the lack of a real gaming environment that uses G-COPSS features, this work tries to emulate the activity that is likely to be generated in a real game. The following operations are performed to the data trace: *1* every player moves after an interval ranging from 5*min* to 35*min*, and *2* for every movement, the player has a 10% chance of moving up, 10% chance for moving down if possible and 80%-90% chance of moving in the same level.

To depict the results more precisely, the player movements are further categorized into the following 6 types:

- To a lower layer: E.g. $1/0 \rightarrow 1/1$ (plane landing) or $0 \rightarrow 1/0$ (satellite come back into atmosphere). No download is required since (s)he has the view already.
- From a zone to a region: E.g. $1/1 \rightarrow 1/0$ (plane take off). Snapshots of 4 leaf CDs (`/1/2` to `/1/5`) need to be downloaded.

Table 5.3: Convergence time for different types of movement in different player moving solutions.

Move Type		**Move Count**	**# of Leaf CDs**	**Convergence time** (95% confidence interval) (*ms*)		
				QR,*window* = 5	**QR**, *window* = 15	**Cyclic-Multicast**
Vertical	To lower layer	302	0	0 (0.00)	0 (0.00)	0 (0.00)
	Zone → region	297	4	9326.89 (266.97)	3441.83 (97.67)	1130.05 (54.92)
	Region → world	166	24	28346.41 (853.15)	11544.09 (248.84)	3191.00 (186.78)
Lateral	To a different zone [same region]	2,407	1	2454.54 (27.92)	1010.07 (11.28)	403.89 (16.75)
	To a different zone [diff. region]	501	2	4690.62 (102.76)	1812.98 (40.17)	637.49 (29.70)
	To a different region	1,297	6	14192.77 (192.40)	5124.74 (67.02)	1600.04 (31.32)
Total		4,970	3.3	6869.55 (191.74)	2600.58 (71.78)	851.53 (23.68)

- A region to a world: E.g. $1/0 \rightarrow 0$ (launching a satellite). Snapshots of 24 leaf CDs (all leaf CDs except `/1/0` to `/1/5` and `/0`) need to be downloaded.
- To a different zone in the same region: E.g., $1/1 \rightarrow 1/2$ (soldier moving within the country). Snapshot of a leaf CD (`/1/2`) needs to be downloaded.
- To a different zone in a different region: E.g., $2/3 \rightarrow 3/2$ (e.g.,soldier moving across country border). Snapshot of 2 leaf CDs (`/3/0`,`/3/2`) need to be downloaded.
- One region to another region: E.g. $1/0 \rightarrow 2/0$ (e.g.,plane moving across country border). Snapshots of 6 leaf CDs (`/2/0` to `/2/5`) need to be downloaded.

Updates of players to the objects they can see are uniformly assigned at the time the update is performed. The $3,197$ objects share a different update rate based on their location. E.g., the number of changes observed on the 87 top-layer objects ranges from $27,742$ to $28,587$ (since every player can see and modify them), whereas the number of changes observed on the 483 middle-layer objects ranges from $4,445$ to $8,460$, and ranges from $1,070$ to $4,073$ on the $2,627$ bottom layer objects. When a player moves, the brokers need to send him/her a snapshot of the area (s)he has moved into. To model the size of the area (since the objects in the area could have been updated), this work assumes that the size of the objects in an area changes according to the updates. Since the original object (version 0) is downloaded with the map, the broker does not send anything if the object has not changed. The object size at version n is calculated by:

$$size(obj_{vn}) = \sum_{i=1}^{n} \Delta^{i-1} \times size(upd_i), \tag{5.6.1}$$

where $size(upd_i)$ stands for the size of ith update packet. Δ is set to 0.95 in the simulation. Therefore, the size of objects at the last version (at the end of the simulation) ranges from 579 to $1,074$ bytes.

Table 5.3 shows the efficiency of two solutions proposed snapshot dissemination. The simulation measures the convergence time as the time it takes for the player to receive an

update after (s)he has moved into the area. 3 brokers are deployed in the network to manage the snapshots.

The players in the cyclic-multicast solution see an average of $851ms$ whereas the players using the query to broker (QR) solutions with a pipelining window size of 15 face an average of $2,600ms$. Pipelining in QR solution optimizes the performance when the window size is increased from 5 to 15. However, there is no further benefit for a higher window size beyond 15, as it has approached the size of the pipe. Moreover, the broker might be the bottleneck in a QR solution, as the number of players moving increases.

The convergence time grows (sub)-linearly with the CD count. In cyclic-multicast, a player can get the snapshot within 4 seconds even if (s)he is moving to the top layer. But note that during this period, if the objects are disseminated in an intelligent manner, the most important objects could begin to show first and the finer details could be downloaded after that.

The total number of objects sent by the broker via cyclic multicast was $1,689,939$, whereas in the query response (QR) approach, the broker sent around $1,700,000$ objects since it received those many queries (queries can be aggregated at the routers). Though the total number of objects sent out in both the solutions were roughly the same, the QR solution consumes more than $26GB$ aggregate network traffic compared with that of cyclic-multicast's $14GB$. This is due to the QR solution incurring more control overhead and the fact that the cache ages out quickly in a gaming scenario. The frequent update also implies that a packet goes to no more than 3 clients.

5.7 Chapter Summary

This chapter presents G-COPSS that functions as an efficient decentralized communication infrastructure for a gaming environment by leveraging the advantages of a content-centric network. With a new hierarchial map partitioning, the designers can provide a more complicated game world while still keep the game efficient.

G-COPSS is implemented both on a test-bed and in a simulator. A microbenchmark is performed to compare the performance of G-COPSS (over 60 players on the test-bed) to that of an IP-server and NDN based approaches. Using trace driven simulations for a system (over 400 players), this work shows that G-COPSS is able to outperform a pure IP-server based gaming environment in regards to aggregated network traffic and update latency. G-COPSS also outperforms the NDN approach, which depends on a query-response model for such a gaming application.

The automatic RP balancing is also evaluated in the simulation. The result shows that the balancing mechanism eliminates the traffic concentration in gaming and there is no packet loss during the RP move period.

Chapter 6

Enhancement: Evolving from Existing Networks

COPSS seeks to meet the *content-centric* user requirements. As a next-generation network design, it is important to provide a smooth transition from the existing network architecture. This chapter describes hybrid-COPSS, a hybrid version of COPSS that addresses incremental deployment of ICN and elegantly combines the functionality of content-centric networks with the efficiency of IP-based forwarding including IP multicast. Furthermore, this work proposes an approach for incremental deployment of caches in generic query/response ICN environments that optimizes latency and network load. To overcome the lack of inter-domain IP multicast, hybrid-COPSS uses COPSS multicast with shortcuts in the ICN overlay. The hybrid approach would also be applicable to the NDN framework.

To demonstrate the benefits of hybrid-COPSS, this work uses a multiplayer online gaming trace (as described in §5) in the lab test-bed and microbenchmark the forwarding performance and queuing effect for both COPSS and hybrid-COPSS. A large scale trace-driven simulation (parameterized by the microbenchmark) on a representative ISP topology is used to evaluate the response latency and aggregate network load. The results show that hybrid-COPSS performs better in terms of response latency in a single domain. In a multi-domain environment, hybrid-COPSS significantly reduces update latency and inter-domain traffic. The contribution of hybrid-COPSS include:

- A detailed design of hybrid-COPSS with both pub/sub and query/response features in a hybrid (IP + ICN) scenario. This work addresses the inter-working of COPSS with IP multicast to achieve both incremental deployment and forwarding efficiency of hash based multicast (similar to IP multicast). This work also presents how COPSS routers seamlessly integrate an IP network infrastructure and content-centric end hosts. Moreover, this work shows how ICN nodes with caches could be placed to optimize query/response functionality.

- A solution to the challenges of inter-domain multicast, through the use of ICN overlay nodes at individual administrative domain edges. Moreover the hybrid-COPSS design provides maximum freedom to individual domains with the capability of distributing and managing their limited IP multicast space, while ensuring that global connectivity is maintained.
- An approach to map the very large (potentially unbounded) address space that a large scale ICN may use, onto a limited IP multicast group address space. The choice of the address mapping has a direct impact on network efficiency. This work provides an efficient mapping schema that reduces wasteful network traffic.
- A study of FIB size and cache hit rate in a pub/sub system using incremental ICN deployment. This work proposes an incremental deployment framework for ISPs and a FIB propagation mechanism to increase the cache hit rate and reduce the FIB size in such environment.
- An evaluation of the performance of hybrid-COPSS against pure COPSS and IP multicast in an experimental test-bed and a trace-driven simulation on a representative ISP topology.

Contents

6.1 Introduction

Incremental deployment is important to all the next-generation network designs. All the designers have to provide a mechanism for ISPs to smoothly transit from the existing infrastructure. Additionally, efficiency concerns on a pure ICN is another motivation for a hybrid version of ICN and IP network.

A major component of NDN is the extensive use of caching at every hop of the network. Specifically, NDN requires every NDN router to process the content request (rather than header-based forwarding) and store the named content to respond to subsequent requests from the cache, in order to achieve better performance than the case if the request for content was just forwarded by the network to ultimately be served from the publisher/source of the content. While the performance penalty of hop-by-hop processing of the content request and response is mitigated by caching and generating the response from the point where the first cache hit takes place, the NDN solution still requires every NDN router to do the complex parsing of the request to forward the request or respond to it. Having a cache at every NDN router is also expensive. As demonstrated in §4.3.1, supporting content centricity entails significant additional processing in the forwarding engine. The excitement of the new content centric network architecture has to be tempered by the performance, cost and complexity consequences of the architecture.

This chapter examines how to evolve from an IP infrastructure to an ICN by co-existing with the IP network. Hybrid-COPSS attempts to support all the *functionality* a COPSS-enhanced ICN provides (both Query/Response and Publish/Subscribe) and provides users with name-oriented/content-oriented access to information. However, the network exploits the cheaper IP forwarding capability where appropriate. Cache hits from the key ICN nodes enable fast response to content requests, but this needs to be balanced against the cost of having a large number of complex ICN nodes. Therefore, additionally, by a judicious choice of placing a limited number of full-fledged ICN nodes that can also cache content at key points in combination with a larger number of hash-based forwarding (similar to IP forwarding), the work addresses the problem of efficient migration to an Information Centric future. The NDN implementation treats ICN as an overlay using TCP/UDP between ICN overlay nodes. However, a tightly integrated approach is believed to be able to provide the best of both worlds – with COPSS routers at the edge and at selected points, and the core routers in the network only performing IP forwarding.

6.2 Hybrid Publish/Subscribe

This section describes the multicast-based delivery model of hybrid-COPSS for incremental deployment and leveraging the efficiency in IP network. The solution retains the content-

centric functionality from both the user's and the end-system's perspective. In [4, 16], the authors proposed that content centric network could be built as an overlay to achieve the ICN functionality. NDN [4] proposes to use UDP packets to encapsulate NDN packets (Interest and Data) or TCP to transfer NDN protocol messages over an IP network. This links NDN forwarding engines via faces (address:port) and forwards packets on a hop-by-hop basis across the IP underlay. While the COPSS architecture can also be implemented as an overlay, this work explores an integrated approach. Hybrid-COPSS tries to provide content oriented functionality that is integrated with the routing and forwarding functionality of an IP network.

To achieve forwarding efficiency for multicast (overlay or IP)-based information dissemination, the solution seeks to reduce the time required for name resolution and complex protocol exchange at every hop in the overlay. Therefore, it is desirable for the heavy-weight COPSS forwarding function to be present only at critical positions and leave intermediate routers to focus only on forwarding. Note that the needs of a query/response system could be different from that of a multicast system. Therefore the solution does not place strict requirements on where the pure COPSS or pure IP routers need to be placed. In fact, the COPSS enabled nodes can be used either with their full ICN overlay functionality or with the more limited, but efficient, functionality (consisting of only multicast and IP like forwarding). This design allows a query/response application to utilize the ICN capability of intermediate nodes when and where needed. The overlay-underlay design of the nodes implies that where needed, there are ICN routers that provide query aggregation and caches (thereby the benefit of cache hits).

6.2.1 Packet Forwarding in Hybrid-COPSS

Hybrid-COPSS seeks to exploit the forwarding functionality of IP in the core of the network to achieve efficiency. The full-fledged functionality of COPSS is present on edge routers (routers directly linked to publishers and subscribers) and at the RPs. The edge routers are directly linked to the RPs on the overlay. The edge routers still maintain CD to RP mapping table. But since the FIB stores the {`address:port`} pair as an outgoing face, edge routers can find the address of the RPs seamlessly. Here, let RA_{CD} and RN_{CD} be the address and the name of the RP module that serves CD respectively. The RA_{CD} can be calculated by $FIB(RN_{CD})$ because the RPs and the edge routers are directly linked at the ICN overlay. Then, this packet will be forwarded to the RP through the underlay. CD to IP multicast group address ($Group_{CD}$) mapping is maintained in the *Group_Table* on the RP ($Group_{CD}$ = $Group\_Table(CD)$). Note that in a multi-domain scenario, the routers at the borders of the domains could also have COPSS functionality. This will be briefly addressed in Section 6.4.

On receiving a Subscription packet from an end host that seeks to subscribe to a CD, the

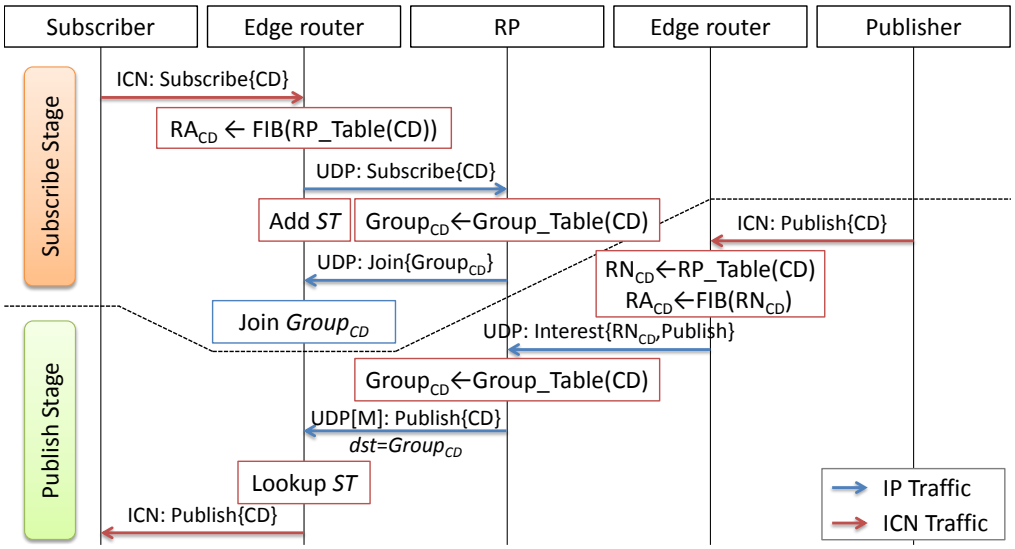


Figure 6.1: Protocol exchange of pub/sub in hybrid-COPSS.

edge router will first modify its ST and then forward it to the RP using the address RA_{CD}. The protocol exchange is shown as the "Subscribe Stage" in Fig. 6.1. When the COPSS RP receives this packet, it will assign an IP multicast group address to the CD if there is no group for the specified CD. It then sends a group join invitation to the edge router to which the edge router responds by joining the specified IP multicast group. In the IP network, an RP-based tree or source-based tree will be formed according to the IP multicast protocol (e.g., PIM-SM [27] or SSM [28]).

To publish content, the publisher sends a Publication packet to the edge router (the user behavior is kept same as COPSS). The protocol exchange is shown as the "Publish Stage" in Fig. 6.1. The edge router encapsulates the packet using an Interest with prefix RN_{CD} and sends it to RA_{CD} (by looking up FIB). When the RP receives this packet, it will decapsulate it and forward it based on the Group_Table, instead of the ST. RP can also use the ST to maintain the Group_Table by replacing the face with the group address. This packet will be delivered to all the edge routers that subscribe to $Group_{CD}$. The edge routers check the CD in the Publication packet and forward it based on their own ST. Since CDs are used to represent content, the sheer volume of CDs could be an order (or multiple orders) of magnitudes greater than IP multicast group addresses. The mapping of the very large, hierarchical ICN namespace onto a bounded, flat IP multicast group address space will naturally result in wasteful traffic being sent on the network, which will have to be discarded by the edge router.

To a significant extent ISPs support IP multicast within their domain. The primary chal-

lenge is in supporting IP multicast across domains. However, in those cases where IP multicast is not supported within a domain, users can rely on a pure COPSS overlay. The network then exploits an overlay multicast tree to minimize the number of copies sent from the RP and on each overlay hop.

6.3 Hybrid Query/Response

In addition to pub/sub mechanisms, query/response is another essential part of content delivery. It could be initiated by an end-host that requests a particular content or in response to receiving a snippet via the pub/sub delivery mechanism. The latter approach (called the second-stage dissemination) is performed when the end-host is interested in receiving the whole video after watching a trailer that was sent by the pub/sub mechanism.

The second-stage dissemination helps reduce the bandwidth used for data delivery since not every subscriber is interested in each piece of data published with the CD (s)he subscribed to. At the same time, this strategy helps publishers with policy control in responding to subscriber requests. For example, the snippet can be seen as an advertisement and the publishers can have access control on the actual complete content. For the first stage, COPSS is used to minimize the announcement latency, and for the second stage, query/response can be used to get the best use of in-network cache, as is typically performed in NDN.

This part focuses on adapting the query/response component to efficiently function in the envisioned hybrid scenario where it co-exists with IP as well as COPSS (multicast based dissemination) nodes.

As mentioned earlier, the needs of a query/response system could be different from that of a pub/sub system and therefore this work does not place strict requirements on where ICN-aware or pure IP routers need to be in the network. Since hybrid-COPSS utilizes IP multicast, it is sufficient for edge routers and RPs to be ICN aware in the basic case.

The simplest approach to enable query/response functionality is to enable them on the COPSS nodes only. But for achieving improved overall efficiency, having more ICN-aware routers can help because of the in-network cache and FIB aggregation capability. However, for the dissemination of the content, traversing a larger number of ICN-aware routers involves more processing at each of the hops and thus contributes to higher latency. Therefore, with the goal of achieving an incrementally deployable architecture and having an optimized delivery mechanism that reverts to hashing based forwarding whenever possible, this work explores optimization strategies for placement of ICN caching nodes versus nodes that forward based only on IP.

6.3.1 RP-based Query/Response

To provide content-oriented query/response in an incremental way, hybrid-COPSS suggests to use the RP-based query/response communication proposed in §3.9.2 that allows for reducing the size of the FIB while increasing the cache hit rate. This mechanism only needs the RPs to know the respective publishers/providers rather than every ICN router. While requesting data, the subscribers adds the RP name as a prefix. The packet will be forwarded to the RP and redirected to the provider.

Different from Eq.3.9.2, in hybrid-COPSS, the publishers/providers can be directly linked to the RPs in the overlay, the number of FIB entries can be calculated as:

$$Size'_{FIB} = n_{rp} \times n_r + n_{pub} \times n_{rp}. \tag{6.3.1}$$

The size is even smaller than the one in Eq.3.9.2. The benefits of such an architecture for the query/response is that it limits the FIB sizes that are being propagated in the network. The rest of the section details the effect of placing ICN aware nodes on this model.

6.3.1.1 FIB Propagation from Consumers

In §6.2, hybrid-COPSS requires the RPs and all the edge routers to be ICN-aware. In the overlay, FIB entries are created from the edge routers directly to the RPs since there are no ICN-aware routers in between. When there are more ICN-aware routers deployed in the network to increase the cache-hit rate for query/response interactions, hybrid-COPSS slightly modify the "Subscribe Stage" to allow intermediate ICN-aware routers to be a part of the query/response tree as well. But, in the "Publish Stage", the packet flow remains the same.

Fig. 6.2 will be used as the example of the following description. In Fig. 6.2, IR_1 is an IP router. CR_{0-6} are ICN-aware routers. $CR_{2,4,6}$ are edge routers. S_1, S_2 and P_1 are linked to CR_2, CR_4 and CR_6 respectively. In the overlay, ICN-aware routers have an ICN module each. The edge routers have ICN modules with additional functionality (e.g., encapsulating, decapsulating and ST for end-hosts). The RPs are designed as a logically separate module for multicast and renaming and therefore an ICN-aware router can have one or more RPs deployed. On router CR_0, there exists both ICN module and RP module. When an RP is setup on a router, the RP module registers a FIB entry on the router's ICN module: {name=`/RP`, face=$R_{x:RP}$}. $CR_{x:ICN}$ is used to denote the {`address:port`} pair of the ICN/edge module on CR_x and $CR_{x:RP}$ for RP module on CR_x.

On receiving a Subscription packet from the subscriber (S_1), the edge router ($CR_{2:ICN}$) will add ST and forward it using UDP packet whose destination is $CR_{0:ICN}$ according to

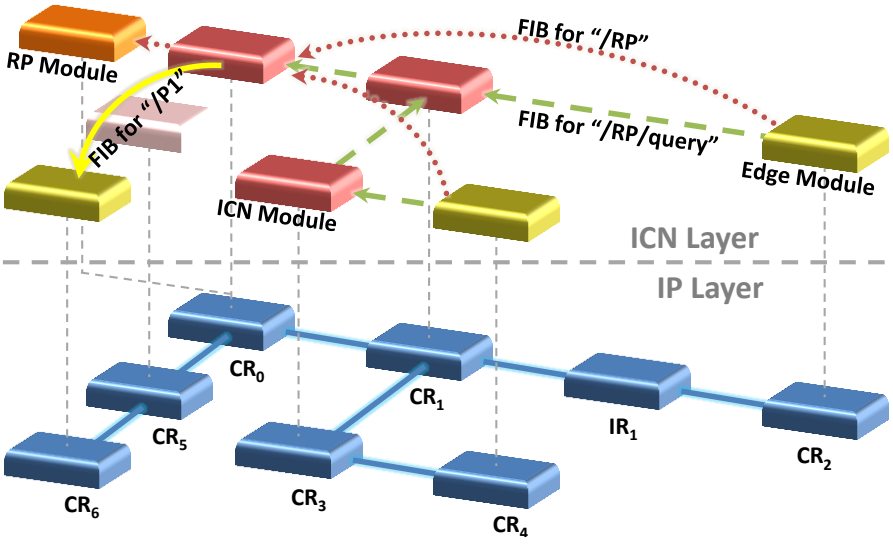


Figure 6.2: Overlay for query/response.

FIB(RP_Table(CD)). Here, hybrid-COPSS uses a flag bit in the UDP packet to mark the special message type. When IR_1 receives the packet, since it is a normal IP router, it forwards the packet directly to CR_1. But CR_1 knows about the special flag bit and instead of forwarding this packet towards CR_0, it redirects the packet to $CR_{1:ICN}$. Its ICN module treats the Subscription packet similar to an Interest packet. An entry {name=/RP/sub, face=$CR_{2:ICN}$} will be added into the PIT that contains the requests that are yet to be served. When $CR_{1:ICN}$ sends the UDP packet out, the *source* field of the UDP packet is changed from $CR_{2:ICN}$ to $CR_{1:ICN}$. When $CR_{0:ICN}$ receives the Subscription packet, it does the same as $CR_{1:ICN}$, but the face of the PIT entry is $CR_{1:ICN}$. The packet is forwarded to $CR_{0:RP}$. $CR_{0:RP}$ responds with a Join packet which contains CD, RP and $Group_{CD}$ (the same behavior as described in §6.2.1). The ICN module treats this packet similar to Data packets with extra FIB add action. $CR_{0:ICN}$ adds $FIB(/RP/query) = CR_{0:RP}$, $CS(/RP/sub) = Join$. Subsequently, this Join packet will consume $PIT(/RP/sub)$ and be forwarded to $CR_{1:ICN}$. $FIB(/RP/query)$ will be created along the path to $CR_{2:ICN}$ and Content Store entry for /RP/sub will be stored in the intermediate routers. When edge module ($CR_{2:ICN}$) receives the Join packet, it checks if there is an end-host subscribing to CD. If so, it will join the IP multicast group specified by the Join packet (the same behavior as described in §6.2.1). When another subscriber S_2 also tries to join CD, $CR_{1:ICN}$ can respond directly instead of going all the way to $CR_{0:RP}$ since it already has $CS(/RP/sub) = Join$. If S_2 tries to subscribe to a sub entry of CD (e.g., CD/x), and the RP serves CD, $R_{4:ICN}$ will add $ST(CD/x) = S_2$ but subscribe to the CD upstream, so as to still get hit on $CR_{1:ICN}$. Fig. 6.2 shows the FIB for /RP (in red dotted lines, for multicast) and FIB for /RP/query (in green dashed lines, for query/response).

6.3.1.2 FIB Propagation from Providers

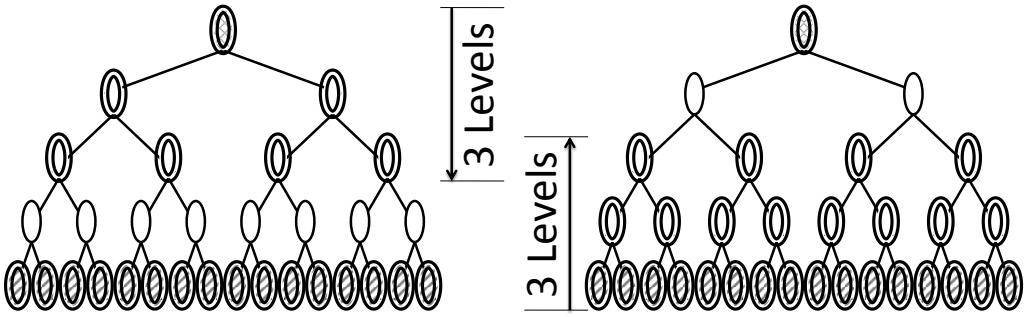
Since every end-host in the network can be a possible publisher, and it is not necessary for some of the publishers to be known by the whole network (they only use single-stage pub/sub), setting up a FIB entry for every possible publisher on the RPs is not advisable in the pub/sub environment. Hybrid-COPSS therefore require FIB creation information to be piggybacked with the first Publication packet. If a publisher needs to serve a prefix ((s)he wants the subscribers to issue a query for the whole data), (s)he will encapsulate the prefix in a Publish packet. In Fig. 6.2, P_1 will encapsulate `/P1` in the Publication packet. On seeing the piggybacked information in the Publication packet, the $CR_{6:ICN}$ will setup $FIB(/P1) = P_1$ and forward the packet to $CR_{0:ICN}$ ($CR_{5:ICN}$ will not see the packet). $CR_{0:ICN}$ will add $FIB(/P1) = CR_{6:ICN}$. The FIB is shown in yellow solid line in Fig. 6.2.

6.3.1.3 Data Dissemination According to FIB

On receiving a snippet with *CD* and Name `/P1/Data1`, S_1 queries data with `/P1/Data1` and include *CD* as a reference. When $CR_{2:ICN}$ receives the packet, it adds $PIT(/RP/query/P1/Data1) = S_1$ and forwards it according to entry $FIB(/RP/query)$, which is $CR_{1:ICN}$. Then, $CR_{1:ICN}$ forwards it to $CR_{0:RP}$ through $CR_{0:ICN}$. $CR_{0:RP}$ removes the `/RP/query` prefix and forwards the Interest (ContentName=`/P1/Data1`) to P_1 through $CR_{0:ICN}$. P_1 responds with the Data packet with name prefix `/P1/Data1`. It will be forwarded to $CR_{0:RP}$ and $CR_{0:RP}$ adds name prefix `/RP/query` to the packet. ICN modules on the other routers forward the Data packet and save it in the cache, just the case as defined in NDN. S_1 will receive data and the packet will be cached in ICN module CR_0, CR_1 and CR_2.

6.3.2 Strategy of Enabling ICN-aware Routers

Although deploying a larger number of caches in the network can increase the cache hit rate, incremental deployment of the ICN nodes may suggest the need to examine the cost of deploying these caches. A higher cache hit reduces server/publisher load, network traffic and load on the nodes that have to process the content further upstream towards the source. However, with a larger number of such ICN enabled nodes, there are more nodes that have to do comprehensive processing of the packet, which increases cost as well as add latency at each of those hops. In this section this work assumes the ISPs have a limited maximum amount of cache that can be deployed across various nodes in the network. This then raises the question of which routers should be replaced/enabled with the ICN (cache) functionality. As proposed earlier, in the case of the RP-based architecture, the query/response path follows the same tree as the one used for pub/sub multicast tree. Subscribers are at the



(a) Top down (b) Bottom up

Figure 6.3: Possible incremental deployment strategies.

leaves of the tree with the RP as the root node. In order to better understand the trade-off, 2 possible ways of deploying ICN-enabled routers are analyzed, considering the logical multicast topology: top down (from the RP down) and bottom up (starting from the end-hosts/subscribers). Fig. 6.3 shows a 5-level (binary) dissemination tree. The root node is the RP, the leaf nodes are the edge routers to the subscribers. According to the requirement of hybrid-COPSS, the RP and edge routers have to be ICN-enabled routers. The ICN enabled routers are marked as nodes with a double circle in the figure. The *top down* strategy deploys ICN-enabled routers starting from the routers directly connected to the RP in the logical tree. Fig. 6.3a shows the structure with 3 levels of ICN-enabled routers according to top down strategy. The *bottom up* strategy deploys ICN-enabled routers starting from the routers directly linked to edge (leaf) routers in the tree. Fig. 6.3b shows the structure with 3 levels of ICN-enabled routers based on the bottom up strategy. Note Fig. 6.3 is for illustration purposes (the number of ICN nodes in the two figures are different).

The advantage of a bottom up model is that since the cache nodes are deployed closer to the leaves (subscribers/querying nodes), a cache hit at the intermediate routers could result in lower latency as well as less network traffic. However, this strategy could suffer from the fact that the total cache is now divided across a larger number of ICN enabled routers. A smaller cache at each node may result in a lower cache hit rate. Alternately a smaller number of ICN enabled routers may be deployed with larger amount of cache on each router, but in the bottom-up case, this may result in only a subset of the end-nodes in the tree seeing the benefit of the cache. On the contrary, in the case of the top down model, the presence of a few cache nodes at the top levels of the tree allows for a larger cache and at the same time has the potential to serve a larger set of end hosts as well as take advantage of the aggregation of user requests, yielding a higher cache hit rate. However since they are farther away from the end hosts, it results in increased response latency and traffic. The performance of the two strategies with same amount of cache will be evaluated in §6.6.2.3.

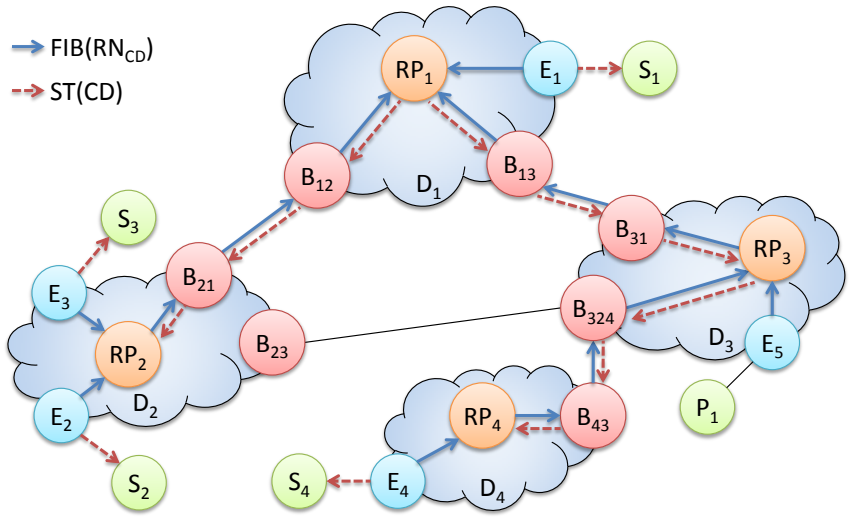


Figure 6.4: Inter-domain multicast.

6.4 Inter-Domain Communication

A problem with using IP multicast, to take advantage of forwarding efficiency, is the inability to go across domains (possibly due to business and deployment considerations). Hybrid-COPSS combines overlay multicast at the ICN layer and IP multicast in the underlay so that a global $Group_{CD}$ mapping is not required, i.e., all the IP multicast information is maintained within the individual domains. This allows ISPs to have different CD to IP multicast mapping in each domain based on considerations such as the load and subscriber count for each CD. Hybrid-COPSS takes advantage of pure IP multicast, while making sure that the content-centric COPSS overlay recognizes the administrative boundaries at the IP layer.

As shown in Fig. 6.4, similar to the requirements of the single domain solution, the edge routers and the RP in each domain are ICN-aware routers. Additionally, in the multi-domain case, boundary routers (marked B_x) are required to be ICN-aware. The overlay uses a COPSS multicast tree rooted at the first established RP (global RP) across all the domains. The individual domains have a local COPSS RP that subscribes to the global RP if there is at least one interested subscriber in its own domain, or a domain downstream.

6.4.1 RP Setup

The first subscription to a CD that is not yet served by any global RP initiates the process of setting up the RP within the originating domain. This RP serves as the root of the global

multicast tree (being the global RP). The RP disseminates the fact that it now serves the CD (for the mapping RN_{CD} = $RP\_Table(CD)$ identifying the RP) to all boundary routers (named *outgoing boundaries*) and the edge routers in its domain.

When an outgoing boundary receives information on the CDs that an RP (from its own domain) is serving, it will set up a forwarding table entry ($FIB(RN_{CD})$) for that particular RP and forwards the information to other boundary routers (named *incoming boundaries*) in adjacent domains. When an incoming boundary receives the "CD serving" information, it first checks if there is already an RP in its domain (to avoid loops). If not, it sets up a FIB entry ($FIB(RN_{CD})$) pointing to the boundary from which it received the serving information and sets up a new local RP for this CD. The newly created local RP then sets up a $FIB(RN_{CD})$ pointing to the incoming boundary in its domain and propagates the serving information to all the edge routers and all the boundaries except the incoming boundary. To minimize the forwarding latency when going across domains, the local RP can be co-located on one of the incoming boundary nodes of a domain. E.g., If RP_3 is located on B_{31} in Fig. 6.4, the latency through D_3 will be $B_{31} \rightarrow B_{324}$ instead of $B_{31} \rightarrow RP_3 \rightarrow B_{324}$. The edge routers will set up $FIB(local\_RN_{CD})$ pointing to the RP in its own domain on receiving the serving information. The FIB information in Fig. 6.4 shows the result of RP setup started at domain D_1, triggered by S_1. Note that B_{23} will not setup another RP in domain D_2 since there already exists an RP in D_2 when the new RP information was propagated to B_{23}. It will also not be considered an outgoing boundary or setup a FIB to RP_2. But B_{324} will serve as the boundary that serves D_4 (tree is routed through D_3), so it has a FIB entry pointing to RP_3.

6.4.2 Subscribe

The subscription procedure in the individual domains is similar to the subscriptions in a single domain case. The edge router forwards the subscription to the local RP, the local RP assigns an IP multicast group for the CD and then asks the edge router to join the local IP multicast group. However, the local RP will also forward the subscription upstream to the global RP (root) according to the forwarding table entry ($FIB(RN_{CD})$). The boundaries and RPs in between will setup their ST appropriately, but it is not necessary to assign an IP multicast group address within these domains. In the example shown, the ST information in Fig. 6.4 shows the result of a subscription by S_1 through S_4 (dashed lines showing the subscription tree). Since there is no subscriber in D_3, no IP multicast group is needed in D_3.

6.4.3 Publish

For the intra-domain multicast, the local RP multicasts the packet using local $Group_{CD}$. But on the overlay, a shortcut-enabled multicast tree is used to optimize forwarding performance

and reduce inter-domain traffic. That is, on receiving a multicast packet (encapsulated into an Interest with the ContentName of the RP), the local RP decapsulates the packet and sends it downstream using $ST(CD)$, except on the incoming face. At the same time, it re-encapsulates this packet using its own RN_{CD} and forwards it according to the FIB. With this shortcut, the Multicast packet does not need to go all the way to the root of the tree and come back down. Instead, it is disseminated to subscribers while being forwarded upstream to the global RP.

For example, P_1 in Fig. 6.4 sends a Multicast packet to E_5. E_5 encapsulates the packet using the ContentName RP_3 and sends it to RP_3. With no subscriber in D_3 ($Group(CD) = null$), RP_3 will only send a COPSS (overlay) Publication packet downstream (through B_{324} and B_{43} to RP_4) according to the ST. At the same time, RP_3 encapsulates the packet into an Interest using RN_{CD}, i.e., RP_1, and forwards it according to the FIB. The Interest will be forwarded through B_{31} and B_{13} to RP_1. RP_1 decapsulates the packet and forwards it according to the ST to B_{12} (and then to B_{21}, RP_2). RP_1 will not forward it to B_{13} since it is the incoming face. Also, RP_1, RP_2 and RP_4 will send IP multicast with $Group_{CD}$ in D_1, D_2 and D_4 respectively. $Group_{CD}$ may differ in the different domains according to the subscription status in each domain. Edge routers receive the packet and forward it to subscribers.

6.4.4 Automatic RP Balancing

Automatic RP balancing method was proposed to relieve traffic concentration ("hot spots") in §3.8. The same approach is adopted in hybrid-COPSS to minimize the effect on inter-domain traffic by using different physical RPs (i.e., $local\_RN_{CD}$) instead of just one global RN_{CD}. The introduction of a new RP and migration of CDs to it for load-balancing affects only the first domain downstream. E.g., if the RP for a CD tries to move from RP_3 to RP_3', RP_3' will set $RN_{CD} = RP_1$, $FIB(RP_1) = B_{31}$ and a subscriber from it (e.g., B_{31} will modify ST, but others like B_{13} will not be affected.) A new FIB entry $FIB(RP_3')$ will be created in RP_4, B_{43}, and B_{324} pointing upstream. RN_{CD} in RP_4 and edge routers in D_3 will be changed to RP_3'. But if there are other domains subscribing to D_4, they will not be affected.

6.5 Management of CD to Multicast Group Mapping

This section examines the efficiency and scalability of hybrid-COPSS in the management of CD to multicast group mapping (i.e., $Group_{CD}$ = $Group\_Table(CD)$). The mapping function controls the tradeoff between the IP multicast address space usage versus excess traffic carried over links when a group address is used for multiple CDs.

Since CDs are used to represent content, the sheer volume of CDs could be orders of magnitudes greater than the available space of IP multicast group addresses. Therefore there is a need to map multiple CDs to a particular IP multicast group address. The mapping of the unbounded, hierarchical ICN namespace onto a bounded, flat IP multicast group address space will naturally result in wasteful traffic being sent in the network. But different mapping functions can result in varying amount of wastage. Imagine that there are 2 CDs: `/FireAlarm` (subscribed to by almost everyone but does not have many updates) and `/ICNMailingList` (subscribed to by only a few people but with frequent updates). If a solution maps both the CDs onto a same IP Multicast group, this group will be subscribed by almost everyone. This results in the updates in the `/ICNMailingList` to be received and discarded by almost every edge router. A large amount of wasteful traffic will be carried by the network.

According to the example above, a mapping function that classifies CDs based on their subscribers and update frequency would be preferred. However, in the true sense of an ICN, neither publishers nor RPs should know who or where the subscribers are. Predicting the publication frequency of CDs is even more difficult since anyone in the network could be a publisher. This work suggests that instead of predicting, it is better to dynamically adapt, by having a re-map function based on various criteria to ensure fair load distribution and reduction of wasteful traffic.

In hybrid-COPSS, every edge router calculates the wasted amount of traffic delivered over an IP multicast group, using a sliding window. Waste is defined as a packet that is received at an edge router, but for which there is no outgoing face according to its own ST. When the amount of waste packets in a group exceeds a certain threshold, the edge router reports the overhead (including the group address and the waste packets for every CD) to the local RP. Based on the total amount of wasteful traffic on every IP multicast group address used, the local RP splits the heterogeneous CDs and assigns them to a new IP multicast group. In some other cases, the RP may also try to combine several CDs into one multicast group when they have similar behavior. When a new mapping is propagated to the whole network (within a domain), all the edge routers rejoin the new IP multicast group if there are subscriptions maintained by them that would be affected. Note that since IP multicast is used within a domain, such a re-mapping function will not affect the other domains. Each domain can have its own $Group_{CD}$ according to their subscription status.

6.6 Evaluation

This section first microbenchmarks a hybrid-COPSS implementation on a lab test-bed compared with pure COPSS for the forwarding efficiency and queuing. The work then uses an

Table 6.1: Average forwarding latency (95% CI) of COPSS and hybrid-COPSS.

(in μs)	**COPSS**	**Hybrid-COPSS**
1^{st} Hop	2778.14(579.13)	2860.21(592.49)
Internal Unicast	**2679.05(575.13)**	**34.71(3.04)**
RP	2749.33(572.32)	2804.65(574.47)
Internal Multicast	**82.76(5.60)**	**33.18(2.90)**
Last Hop	83.26(6.10)	140.65(5.79)

online gaming trace and a Twitter trace to evaluate the performance of these architectures under load with a simulation parameterized by the microbenchmark results.

6.6.1 Microbenchmarking

Hybrid-COPSS is implemented and deployed on the lab test-bed and compared to the implementation described in §3.10. Similar to the benchmark in §5.6.1, 62 players are used to load the test-bed implementation, but with a longer period (2*min* warm up and 10*min* evaluation) from the gaming trace to get statistically significant results. The benchmark also tracked every packet using Wireshark [98] and calculated the average processing time for different actions at every router by tracking the arrival and departure time of that packet. Six different kinds of operations are defined here. For the 1^{st} hop router, the last hop router and the RP, the benchmark does not breakdown the performance of individual encapsulation, decapsulation and forwarding functions. These routers are treated as black boxes. However, for internal routers, the functionality of unicast (Interest in COPSS; UDP unicast in hybrid-COPSS) and multicast (Publication in COPSS; UDP multicast in hybrid-COPSS) are measured separately.

6.6.1.1 Microbenchmarking Results

The average forwarding latency on different routers along with the 95% CI is shown in Table 6.1. The results confirmed the observation that ICN (especially NDN) forwarding is much more expensive than IP forwarding. The 1^{st} hop router and the RP in both COPSS and hybrid-COPSS require FIB lookup functionality, as does COPSS unicast forwarding even at the internal routers. With hybrid-COPSS, the internal unicast is UDP forwarding, which is far more efficient. The last hop router and the internal multicast take less time due to the simpler ST lookup in COPSS. With hybrid-COPSS, the internal multicast is IP multicast forwarding, which is relatively efficient. In hybrid-COPSS, although it incurs a

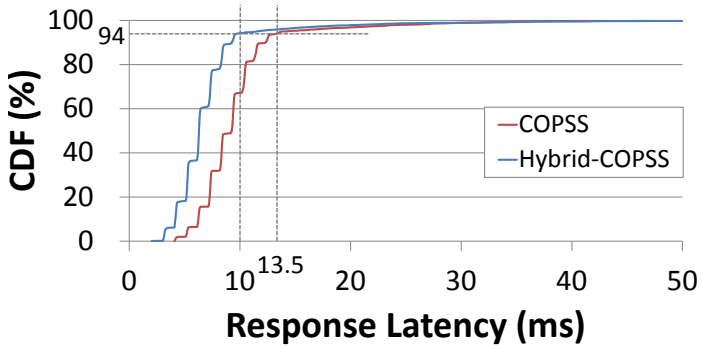


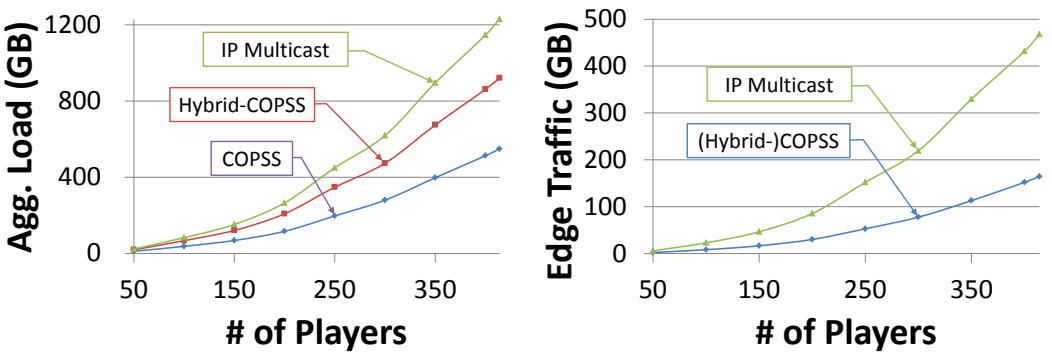
Figure 6.5: Response latency CDF of COPSS and hybrid-COPSS.

slight overhead on the edge routers and the RP (around $70\mu s$), the internal routers even outperform COPSS multicast since no name resolution is required there.

The average update latency in hybrid-COPSS is $6.95ms$, compared to $9.54ms$ in COPSS. Fig. 6.5 shows the CDF of the total update latency. Note that more than 94% (compared to only 67% for COPSS) of the new updates/publications in hybrid-COPSS incur a latency of less than $10ms$ while in COPSS the same 94% take $13.5ms$. Since the benchmark uses a simple topology with only $1-2$ internal routers between the RP and edge routers, higher performance gains should be observed in a typical network topology with more intermediate hops.

6.6.2 Large Scale Trace-Driven Experiments

To further evaluate the performance of hybrid-COPSS in a large scale environment with realistic network topologies, trace-driven simulation is used with two traces: one from a multiplayer online game (similar to §5.6.2.1), and the second from Twitter (similar to §4.3.2.1). The evaluations look at both single and multi-domain environments for the strategy for incremental deployment for pub/sub and query/response. Note that in the case of the Twitter trace evaluation, most subscribers are treated as pure receivers with only 50 of them being treated as publishers. This is intended to emulate scenarios where the ratio of publishers to subscribers vary. On the other hand, for the gaming evaluation, all players send as well as receive updates. As the number of subscribers grows, additional load is generated per player, making it more challenging to support in the network.



(a) Aggregate network load

(b) Edge traffic

Figure 6.6: Single domain performance: gaming trace.

6.6.2.1 Data Traces

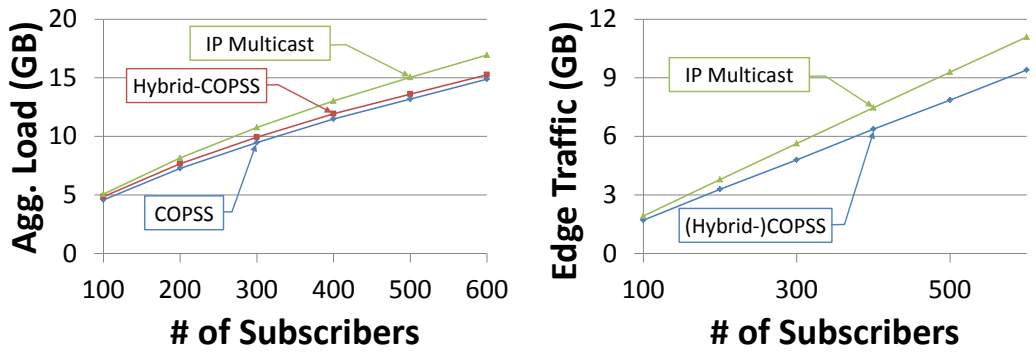
Gaming Trace: The gaming data trace from [103] is processed similarly to the one described in §5.6.2.1. In hybrid-COPSS, $Group_{CD}$ is manually assigned: CDs in one region share an IP multicast group (7 CDs in 1 group: $Group_{/i/*,/i} = 224.0.0.i$), and `/0` uses a single group since every player subscribes to it.

Twitter Trace: The Twitter trace is similar to the one described in §4.3.2.1. Hybrid-COPSS maps every 1^{st} level CD to a unique IP multicast address. To have a sufficiently large message for the query/response case, the Tweet size is scaled by a factor of 128. A publisher publishes the original message size as a snippet first and then the subscribers probabilistically query for the complete message on receiving the initial snippet (Tweet).

6.6.2.2 Single Domain: Pub/Sub

RocketFuel [100] (AS-3697, 79 routers, see Fig. 4.2) is used as the core topology of a single domain simulation. A total of 200 edge routers are randomly assigned to the core routers (1-3 edge router(s) per core router). Both gaming and Twitter traces are used to evaluate the performance of hybrid-COPSS here. The subscribers or players are evenly distributed on the edge routers. 3 routers with the minimum average shortest path distance to all the edge routers are chosen as the RPs (3 RPs are sufficient to efficiently handle the gaming trace as was described in §5.6.2.2). The IP multicast groups in hybrid-COPSS are distributed on the 3 chosen RPs (an RP can serve several multicast groups). Pure IP multicast is also compared using the same RP and $Group_{CD}$ settings.

In the Gaming trace, with cheaper IP forwarding, hybrid-COPSS achieves an average up-



(a) Aggregate network load

(b) Edge traffic

Figure 6.7: Single domain performance: Twitter trace.

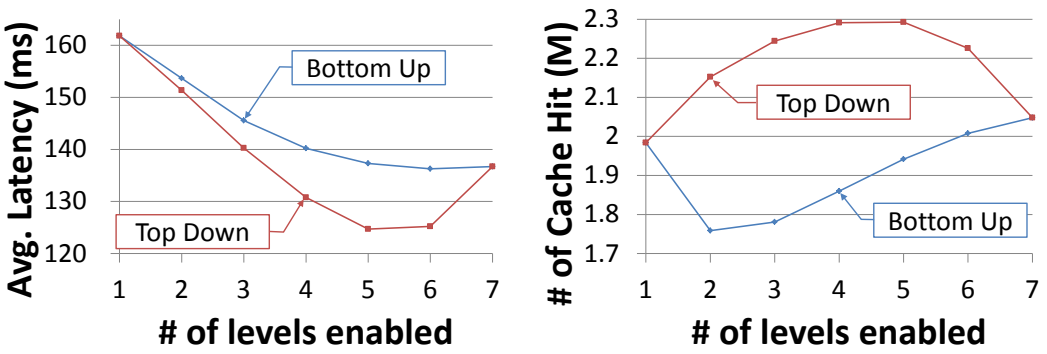
date latency of around 73.7*ms*, compared to 84.6*ms* with COPSS. However, because of an insufficient number of IP multicast groups (7 CDs per IP multicast group), hybrid-COPSS incurs a larger load in the network compared to COPSS, but less than IP multicast. Fig. 6.6a shows the performance of the solutions varying number of players. This demonstrates that hybrid-COPSS can be integrated into the current IP architecture without substantial performance degradation.

With the same number of multicast groups, IP multicast and hybrid-COPSS result in the same amount of traffic in the network core. However at the edge with IP multicast, since the last hop router does not do filtering, end hosts will have to receive all the unnecessary packets and discard them if they find them to be of no interest. The wastage on the edge is shown in Fig. 6.6b. It causes substantial computational and communication overhead on the end host as the number of players increases. For instance, this could be a substantial penalty on mobile devices with limited battery power. Since hybrid-COPSS does not change the user behavior, the edge traffic in hybrid-COPSS is the same as COPSS.

Fig. 6.7 illustrates the same trend for the Twitter trace, except that the network load(Fig. 6.7a) and edge traffic(Fig. 6.7b) grow sub-linearly in all solutions. This is due to the fact that although there is an increase in the number of subscribers, the network is able to take advantage of multicast. The aggregation of subscribers due to their common interests results in increased efficiency for multicast.

6.6.2.3 Single Domain: Query/Response

A separate simulation is used to compare the performance of the two cache deployment strategies. To better understand the results, a synthetic topology of an 8-level binary tree



(a) Avg. response latency

(b) Cache hit rate

Figure 6.8: Performance of different incremental deployment strategies.

(composed of 255 routers) is used in the simulation. The publishers are at the root of the tree and the 256 subscribers are the leaves of the tree. The one-way latency on the links is set to 10ms). The Twitter trace is used, where snippets are sent as an announcement and the subscribers then query for data they are interested in (e.g., video clips). The subscribers have a 50% probability of querying for the data on receiving an announcement. The delay before issuing the query ranges from 5sec to 1hr. The total cache size in the network is set to 2.55GB, divided equally among all the ICN-aware routers in the network. The total published data size is around 4.5GB.

The results are shown in Fig. 6.8, where the x-axis $level = n$ denotes the number of levels of routers that are cache enabled. Therefore $level = 2$ implies that 2 levels of routers are cache enabled, both in the top down approach and bottom up approach. Note that the results (in terms of response latency, cache hit rate and network load) are the same for both top down and bottom up at $level = 1$ and 7. For $level = 1$, only the RP and the edge routers are cache enabled, and for $level = 7$, all the routers in the network have caches.

There are multiple criteria interacting here that affect the tradeoff of where and how many ICN routers to deploy: *individual cache size*, *the cache locations* and the *strategy* (top-down or bottom-up). With the top-down strategy, when $level = 1$ (RP and 128 edge ICN routers), with a per node cache size of 20.24MB, the overall cache hit rate is lower than that with $level = 2$ (RP, 128 edge and 2 extra ICN routers) where per node cache size is smaller at 19.93MB. The higher cache hit rate is gained by the aggregation of requests across users at the second level in the tree that the 2 extra ICN routers can respond to from their caches. The latency also reduces as the levels of ICN routers increase. This improvement with the top-down strategy continues up to a point ($level = 5$), reaching a peak in the overall cache hits and reduced response latency. When there are 6 or 7 levels of ICN-enabled routers, the cache hit rate reduces because the individual cache sizes are smaller than the working

set size. Correspondingly, the response latency goes up, even though there are more ICN routers further down the tree.

For the bottom-up strategy, the cache hit rate behaves quite differently. Going from $level = 1$ to $level = 2$, the cache hit rate goes down, even though the number of ICN routers goes up. This is because the total cache is divided across more routers (RP, 128 edge and 64 extra ICN routers). Because the extra ICN routers do not see a benefit of any aggregation of requests and therefore do no yield an increase in overall cache hits. As the simulations go up the hierarchy in the tree with the bottom-up strategy (going from $level = 2$ to $level = 3$ and higher), the cache hit rate increases, and also results in reducing the response latency. Finally, when comparing the top-down vs. bottom-up strategy, the cache hits for the top-down is consistently higher (better aggregation of requests at the top levels of the tree). The average response latency for the top-down strategy is also consistently better, even though fewer ICN routers may be used (e.g., top-down $level = 2$ has only 3 extra ICN routers while bottom-up $level = 2$ has 64 extra ICN routers). This is because of the higher cache hit rate for the top-down strategy as a result of better aggregation of requests. The network load is also lower with the top-down strategy.

This result suggests that when an ISP has a fixed number of ICN nodes to deploy in the network (i.e., building an overlay network), it is better to take a top-down approach, enabling the ICN functionality starting from the nodes directly linked to the RP, but make sure every router having enough cache size compared to working set size. The top-down strategy can achieve lower response latency with fewer expensive ICN-enabled routers. This serves as an indication that the RP-based dissemination tree structure (see Fig. 3.7b) is better than a publisher focused, source-based tree (see Fig. 3.7a) solution. The source-based tree would have to inevitably employ a bottom-up approach from the edge to the core. Note that the strategy used here can be applied in the a multi-domain scenario as well.

6.6.2.4 Multi-Domain Simulation

The work then investigates hybrid-COPSS in a multi-domain scenario using RocketFuel (Telstra, AS-1221), which has clear domain structure, as the core topology. According to [100], Telstra has hubs in major cities (Sydney, Melbourne, Perth) with spokes elsewhere. Every city is considered as a domain and the routers at these major cities as the core routers. The topology (Fig. 6.9) also shows the weights of the inter-domain links between the 13 boundary routers (marked bold). 3 (global) RPs are selected on the boundary routers in Sydney, Canberra and Melbourne. 207 edge routers are added onto the core routers, based on the proportions of the number of core routers in that city in the original topology. Every edge router can have 1 or 2 link(s) to a core router in the same city (but not to the boundary routers). The latency from an edge to core router is a random value between $2ms$ and $8ms$.

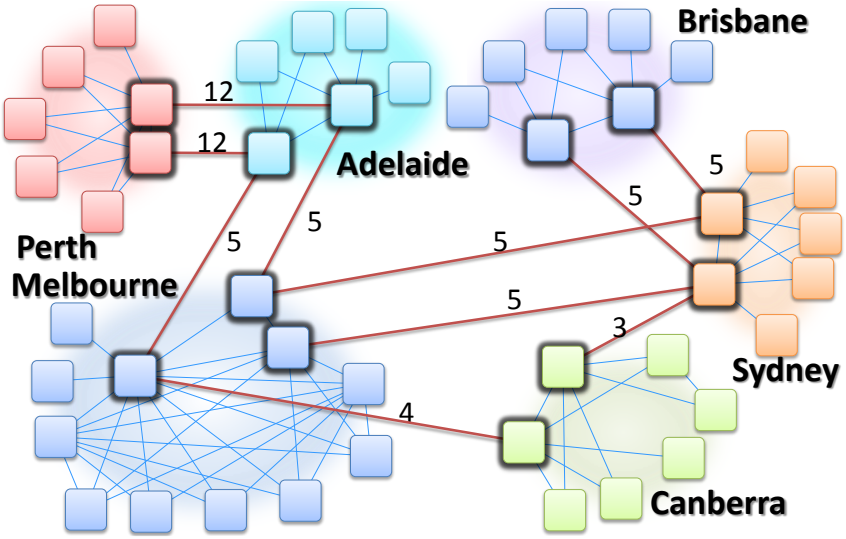
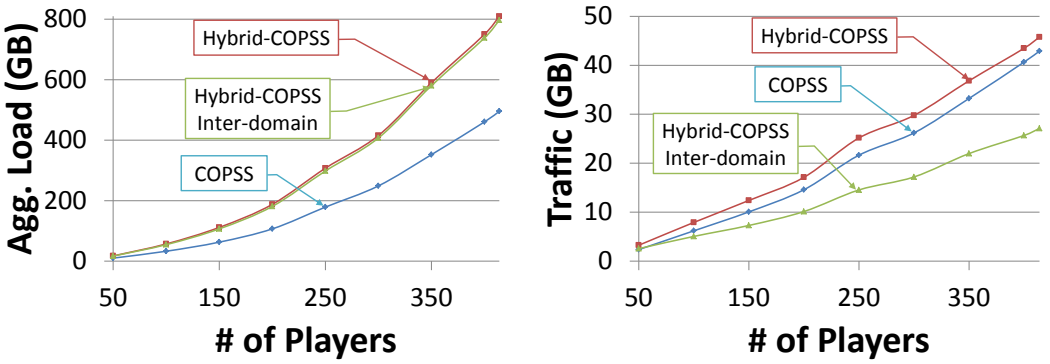


Figure 6.9: Multi-domain topology: RocketFuel (Telstra, AS-1221).



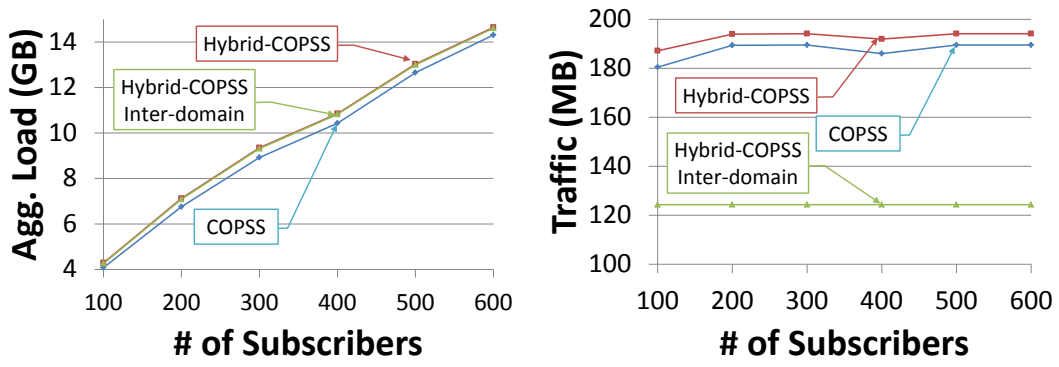
(a) Aggregate network load

(b) Inter-domain traffic

Figure 6.10: Multi-domain performance: gaming trace.

The subscribers or players are linked to the 207 edge routers evenly. Both the Gaming trace and the Twitter trace are used to compare the performance of pure COPSS solution, a simpler, basic hybrid-COPSS solution (that does not consider inter-domain properties) and the hybrid-COPSS inter-domain solution with varying number of players/subscribers.

Similar to the single domain, for the multi-domain gaming case, hybrid-COPSS achieves lower average update latency (around 52.09*ms* compared to 61.13*ms* in pure COPSS) but results in wasting network bandwidth because of the severe shortage of IP multicast group addresses. Inter-domain hybrid-COPSS, however, provides an alternative. Because mul-



(a) Aggregate network load

(b) Inter-domain traffic

Figure 6.11: Multi-domain performance: Twitter trace.

ticast routing can take advantage of shortcuts across domains, a multicast packet does not have to go all the way to the global RP to be forwarded to subscribers. The solution reduces the average update latency by about 2.46ms. Hybrid-COPSS also cuts the inter-domain traffic almost by half (Fig. 6.10b), and reduces the aggregate network load slightly compared to simple hybrid-COPSS (Fig. 6.10a). This means the inter-domain solution is much "cheaper" for ISPs even compared to a pure COPSS solution. Moreover, as described above, this solution does not need routers in different domains to know each other's IP multicast group mappings. Thus, it becomes more practical than other solutions that depend on inter-domain multicast.

With the multi-domain Twitter trace, hybrid-COPSS has shorter response latency than pure COPSS (51.35*ms* vs. 59.05*ms*) and inter-domain solution reduces it by an additional 1*ms*. As for the inter-domain traffic, since subscribers do not publish (in the Twitter model) and there is at least one subscriber per domain in all the scenarios, the inter-domain traffic does not vary with an increasing number of subscribers (Fig. 6.11a). But, it can be observed that the inter-domain hybrid-COPSS solution reduces the traffic by almost $1/3$ compared to a COPSS solution that is not aware of domain boundaries (Fig. 6.11b).

6.7 Chapter Summary

This presents hybrid-COPSS, an architecture to integrate ICN functionality with the current IP architecture. A detailed solution is proposed to integrate both the pub/sub and the query/response based information-centric architecture in an IP network. Hybrid-COPSS is designed to be as generic as possible and is therefore applicable to the query/response based NDN solution as well.

This work addressed the 3-way tradeoff that arises when considering the incremental deployment of ICN, in terms of *traffic*, *latency* and *cost*. There is a higher amount of packet traffic (both within a domain as well as inter-domain) as the solution goes more towards a pure IP environment; there is increased latency in a pure ICN environment because of the additional per-hop processing; finally there is an increase in processing cost for each ICN hop because of the additional complexity. The evaluations suggest that hybrid-COPSS strives to achieve a proper balance in this trade-off by putting ICN functionality at key points and hash-based forwarding at the other routers. Moreover, the query/response dimension of hybrid-COPSS is optimized in the work and the evaluation shows that an RP based top down approach provides the best means for service providers to incrementally deploy caching-enabled ICN nodes.

The hybrid-COPSS inter-domain solution recognizes the current challenges in having inter-domain IP multicast and overcomes it with the use of ICN overlay nodes at the domain edges. The inter-domain hybrid-COPSS cuts inter-domain traffic almost by half even compared to the pure COPSS solution.

Chapter 7

Enhancement: Reliability and Congestion Control

Large scale data dissemination seek efficient ways for content distribution, no matter if it is pub/sub-based (e.g., video and audio streaming) or query/response-based (e.g., software updates and social media applications). However, most of such applications have limited loss tolerance. The new ICN paradigm offers an alternative of reliable data delivery by naming content and delivering from any intermediate point (e.g., caches). But managing congestion is a challenge in ICN because of the lack of an end-end session context over which a "flow" may be controlled. Flow and congestion control as well as reliability are often considered even more of a challenge in content-centric publish/subscribe systems, where the nature of information dissemination is similar to multicast.

This work first shows that a particularly thorny problem of receivers going *out-of-sync* by using existing in-sequence congestion control mechanisms in environments with heterogeneous receivers renders ICN solutions ineffective, even with routers caching content. The author argues that separating reliability from congestion control would lead to more scalable and efficient data dissemination, and proposes *Control Protocol for Scalable and Adaptive Information Dissemination* (SAID), that can be used by both pub/sub and query/response data dissemination. To maximize the amount of data transmitted at the *first attempt*, receivers request *any next* packet in a flow instead of *next-in-sequence* packet, independent from the transmitting rate of the provider. This also allows the provider to send data at an application-efficient rate without being affected by the slower receivers. SAID ensures all receivers successfully obtain all the packets eventually by cooperative repair. By exploiting ICN capabilities, SAID assures reliability and preserves privacy without unduly trusting other receivers. The contributions of SAID include:

- A model-based proof and emulation-based experimental result to demonstrate the fundamental problem of out-of-sync and how it affects the network,

- A new data request model ("any next packet") that leverages the benefits of the NDN framework with minor modifications to avoid the out-of-sync problem in order to maximize the utilization of every packet sent from the provider,
- A modified AIMD window control for congestion avoidance that can adapt to any sending rate to satisfy different application requirements,
- An application-level control mechanism for the sending rate from the data provider,
- An efficient repair mechanism assuring receiver privacy and maintaining content integrity, and
- A detailed evaluation on the behavior of the individual components as well as the performance of the overall solution in both file delivery and streaming applications.

Contents

7.1 Motivation

Large scale data dissemination such as video streaming (YouTube, NetfFlix, etc.), online social networks (Facebook, Twitter, etc.) and news/entertainment distributed over the network (CNN, BBC, RSSs feeds, etc.) have become common. Many of these applications are loss intolerant and seek efficient and reliable content distribution support from the network, while at the same time having to deal with heterogeneous receivers. Researchers have been working on achieving efficient and reliable large scale information delivery for a long time. NDN is a new paradigm that has the potential to achieve efficient large scale data delivery. With the use of in-network caches, NDN can exploit the temporal locality among the consumers to outperform existing IP network-based approaches for a variety of information delivery situations. Moreover, since entities in the network exchange information primarily on content names, the identity of the consumers need not be revealed. This enables user privacy. Data integrity and provenance can be established based on the per-packet data generator-signature required in NDN.

Although NDN does not mandate a congestion control mechanism, most of the existing solutions [58–62] choose to use a TCP-like mechanism to limit the number (window) of requests outstanding from a receiver. Since the network maintains flow balance where one Interest retrieves at most one Data packet, these mechanisms adapt the window using the *Additive Increase Multiplicative Decrease* (AIMD) principle, much like TCP [105, 106]. When considering efficient large scale data dissemination like pub/sub, where every data might be consumed by a large number of receivers, TCP-like mechanisms can have significant shortcomings, especially with receivers having different receive rates.

With heterogeneous receive rates, a problem is that the receivers might end up being *out-of-sync* even when they start getting the data from the same time and with optimal cache management in the network. The out-of-sync problem can briefly be described as follows: NDN routers cache content as they are forwarded. When there is temporal locality of requests from different receivers, that router can respond with content that is cached. However, with receiver heterogeneity, even the requests from receivers for the same data flow can diverge over time. Requests from the faster receivers can be well ahead of those from slower receivers. Eventually, when the gap between the faster and slower receivers becomes too large, their requests can no longer be aggregated at the intermediate routers or satisfied by the cache. The slower receivers' requests will have to be satisfied by the provider (via retransmission) as a separate flow. These retransmissions will compete for the bottleneck bandwidth and the overall throughput can dramatically reduce. This is a fundamental issue as long as there are heterogeneous receivers and with limited cache sizes at the routers. It can be exacerbated with scale – and this occurs even more often in the core of the network.

This chapter proposes *Control Protocol for Scalable and Adaptive Information Dissemi-*

nation (SAID) in ICN, a novel mechanism enabling large scale efficient data dissemination. SAID achieves efficiency by separating reliability from congestion control. By requesting for "any" packet that comes next (similar to a subscription) instead of the "next packet in sequence", SAID is able to maximize the amount of packets delivered *at the first attempt*. Reliability is eventually achieved by consumers performing information-centric repair which also has the attractive feature of preserving privacy and trust. SAID can be used by both query/response applications and pub/sub applications, even those who do not mandate reliability (but requires congestion control).

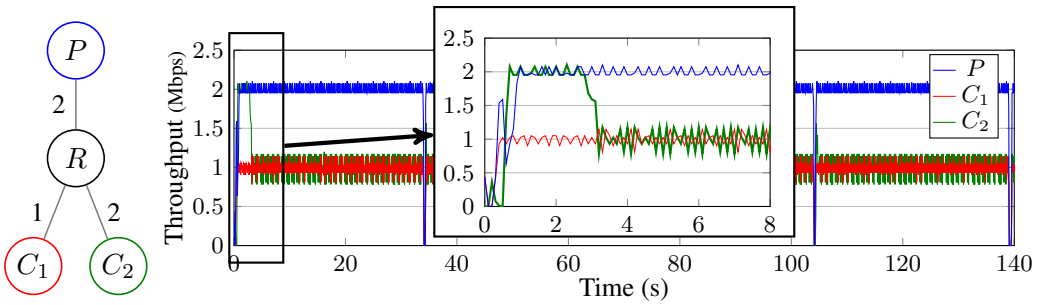
7.2 Study on Out-of-Sync

This section describes the out-of-sync problem that persists despite the different congestion control mechanisms proposed for ICN. This work first shows how the out-of-sync problem diminishes the benefit of in-network caches and use of pending interests. This is demonstrated with an emulation using CCNx 0.8.0 along with a congestion control mechanism similar to ICP [58]. After that an analytical model is used to prove that the out-of-sync problem is generic with heterogeneous receivers, and should be expected with other receiver-driven in-sequence congestion control mechanisms as well.

7.2.1 Demonstration of Out-of-Sync via Emulation

A simple emulation performed in Mini-CCNx is used to demonstrate the out-of-sync problem. The network topology and the link rates (in $Mbps$) are shown in Fig. 7.1a. The latency on all the links is $2ms$. Router R has a 50 packets' cache[2]. Consumers C_1 and C_2 start to request the same content (comprising $8,965pkts$, $\sim 35MB$) from provider P at the same time. The throughput of the end hosts are shown in Fig. 7.1b. For the first 2.5 seconds, the PIT and the network cache benefit both receivers. Requests from C_1 either get aggregated or get a cache hit on R. During this period, the overall network throughput is $3Mbps$ (sum of downstream link capacities of R). Subsequently, with heterogeneous receiver rates, the receivers' requests deviate farther apart. The requests from C_1 can no longer be satisfied by the cache and they are forwarded to P as a different flow. The response to these requests start to compete for the bandwidth on the link from P to R and the receive rate of C_2 is thus affected. Since the congestion control protocol tries to achieve fairness between the receivers (flows), the receive rate of the two consumers becomes $1Mbps$ each, and the overall

[2] A relatively small cache is used to quickly demonstrate the out-of-sync problem. However, the problem is fundamental and occurs even with much bigger caches. Please see §7.2.2 for the relationship between the cache size and the heterogeneity allowed among the receivers.



(a) Topology (b) Throughput on the end hosts

Figure 7.1: Out-of-sync problem emulated in a simple topology.

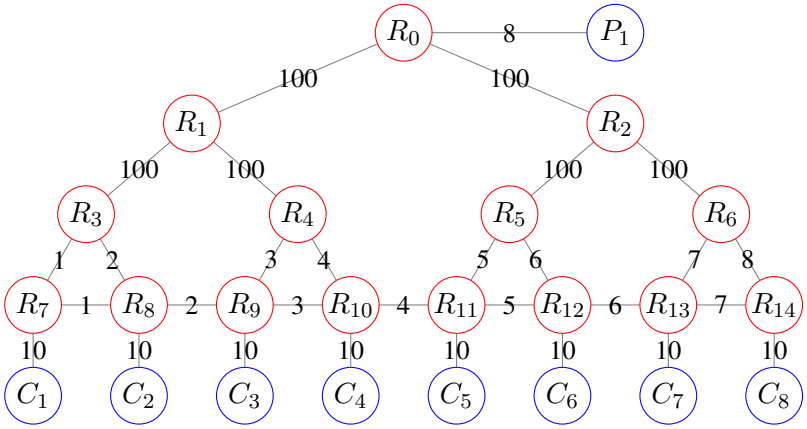
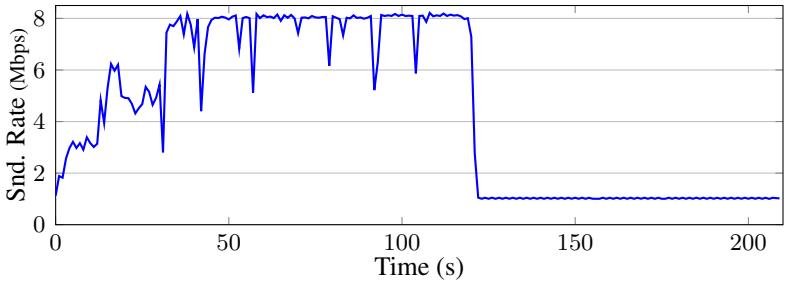


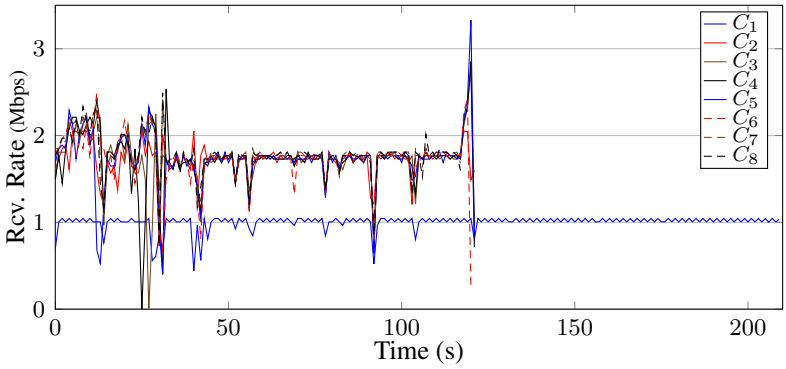
Figure 7.2: Dissemination tree topology (and bandwidth in *Mbps*).

network throughput is reduced to $2Mbps$, thus under-utilizing the bandwidth by 33%. For the entire session, less than 2% of the requests from C_1 see a cache hit.

Similar results occur in more complex topologies (e.g., Fig. 7.2). P_1 is the content provider and all the consumers C_1-C_8 start to request the same content from P_1 at the same time. The cache size is $50pkts$ and the content comprises $5,230$ packets. The sending rate of P_1 and the receive rates for all the consumers are shown in Fig. 7.3. The throughput of P_1 reaches a maximum prior to the receivers getting out of sync from each other. However, after 40 seconds (when the network becomes stable), the throughput observed on each receiver is only around $2Mbps$. The total network throughput is around $13Mbps$ compared to the ideal $36Mbps$. The average number of transmissions of each packet by the provider P_1 is 4.36 instead of 1 in the ideal case. P_1 receives 5 requests for $\sim$70% of all the packets; $2-4$ requests for $\sim$28.5% of all the packets, 1 request (desired) for only 74 packets (less than 1.5%).



(a) Throughput from P_1

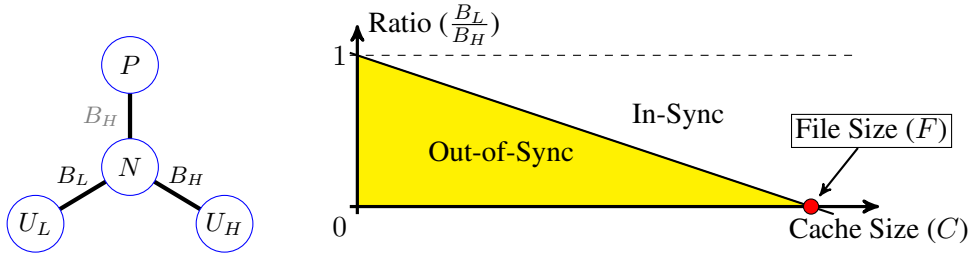


(b) Throughput to consumers

Figure 7.3: Out-of-sync: On larger dissemination tree.

Through the emulation experiment, one can see that due to the heterogeneity of the receivers, the cache in the intermediate routers might not be enough to absorb the difference in the request rates of the fastest and slowest receivers. The out-of-sync situation happens. When the slower receivers re-issue requests, these requests are seen as being for a different "new" flow, since they can no longer be aggregated or be satisfied from the cache at the routers. These "new" flows will then compete on the network links with packets of the original flow. In some cases, this would even be with faster receivers on the common links, and affect their download rate as well.

The out-of-sync problem can happen even when all the receivers start requesting at the same time and even with the optimal cache replacement policy. Note that the provider has to re-transmit packets as long as the intermediate router drops the packets within the gap (the difference in the sequence number of the packet requested by the fastest and slowest receiver). When the gap is larger than the available cache size for the flow, no matter which packet the replacement policy chooses, an additional transmission from the provider is required.



(a) Single-branching model

(b) Relationship between out-of-sync and request rate, cache size and flow size

Figure 7.4: Out-of-sync in a single-branching model.

7.2.2 Analytical Model for the Occurrence of Out-of-Sync

Receiver-driven feedback-based in-sequence congestion control protocols (e.g., TCP) share the following features: *1*) each data consumer has a local view of the request as if (s)he is the only consumer in the network, *2*) all the data consumers tend to get a (statistically) fair-share of bandwidth, and *3*) the packets in a data object are requested in-sequence and out-of-order is seen as an indication of congestion. Almost all the existing congestion control protocol proposed for ICN fall into this category.

To show the universality of the problem, the model for congestion control is generalized by assuming a best-case scenario where each receiver is receiving a flow of data with a constant bit rate which is exactly the fair-share that receiver can get. This part studies the maximum heterogeneity that can be supported given a certain cache and flow size while the receivers still remain in-sync till the end of the flow. Since the study mainly focuss on a single flow, it is reasonable to assume a simpler case that the network status (i.e., available bandwidth, cache size and latency) does not change during the lifetime of the flow.

The study starts with a more precise definition of out-of-sync.

Definition 7.2.1 (Out-of-sync). *Consider a network with multiple routers interconnected in a tree topology and a flow f that has a provider on the top and receivers at the leaves of the tree. At a branching router N, where available bandwidth to the downstream receivers is in the range $[B_L, B_H]$, the out-of-sync occurs when the difference in the received file size (the gap, G) between the fastest and slowest receiver in the sub-tree below N is larger than the available cache size C for flow f.*

A single-branch model is used to demonstrate the relationship among the cache size, flow size and receiver heterogeneity in a simple scenario.

Lemma 7.2.1. *For a branching router N with cache size C in a dissemination tree, with the request rates of the immediate downstream links in the range $[B_L, B_H]$, to avoid out-of-sync, the following condition should hold, i.e.:*

$$\frac{B_L}{B_H} \geq (1 - \frac{C}{F}), \tag{7.2.1}$$

where F is the size of the flow.

Proof. According to Def. 7.2.1, to avoid out-of-sync at N, it is sufficient to consider 2 immediate downstream links with the largest and smallest available bandwidth. That is, consider the single-branching topology in Fig. 7.4a such that the two data consumers (U_L and U_H) are requesting for a same flow with size F and their available bandwidth are B_L and B_H ($B_L \leq B_H$).

When the receivers are in-sync, the request sent upstream by N targets a downstream rate of B_H, matching the receive rate of the faster receiver (this is true for all protocols where the network node does not perform an explicit congestion control function and depends on the receivers to generate an appropriate request rate). The download period for U_H is:

$$t = \frac{F}{B_H}.$$

The maximum gap G between the U_L and U_H is therefore:

$$G = (B_H - B_L) \times t = (1 - \frac{B_L}{B_H}) \times F$$

According to Def. 7.2.1, to keep the consumers in-sync, we need:

$$G = (1 - \frac{B_L}{B_H}) \times F \leq C. \tag{7.2.2}$$

□

We can get (7.2.1) by reforming (7.2.2).

The requirement for clients in-sync (Eq. 7.2.1) is presented in Fig. 7.4b. The result shows that the requirement for being in-sync cannot be satisfied when the heterogeneity ($\frac{B_H}{B_L}$) is larger, the flow size F is larger, and/or available cache size C is smaller.

Since the request and data paths in large scale data dissemination usually form a tree structure rooted at the data provider, in-sync requirements is then extended into a k-level tree.

Theorem 7.2.2. *For a dissemination tree with k levels and every intermediate router having cache size C, all the receivers will be in-sync only when the available bandwidth between the fastest receiver and the slowest receiver follow:*

$$\frac{B_L}{B_H} \geq (1-\frac{C}{F})^k \tag{7.2.3}$$

Proof. This theorem is proved by contradiction. Suppose that the request rates of the highest and the lowest receivers satisfy

$$\frac{B_L}{B_H} < (1-\frac{C}{F})^k, \tag{7.2.4}$$

and all the receivers are in-sync.

Without loss of generality, we assume that the receiver with the lowest rate B_L is a downstream consumer of a router N_t at level $t \in [1,k]$. Let $B_{H,t}$ be the highest request rate among the downstream consumers of the router N_t. Note that the consumer with request rate B_H does not have to be the immediate next hop of N_t, the intermediate routers will always forward requests according to the fastest receiver. According to Lemma 7.2.1, we have

$$\frac{B_L}{B_{H,t}} \geq 1-\frac{C}{F}. \tag{7.2.5}$$

According to (7.2.4) and (7.2.5), it follows that

$$B_{H,t} < B_H \times (1-\frac{C}{F})^{k-1}. \tag{7.2.6}$$

The router N_t is a downstream consumer of a router N_{t-1} at level $t-1$. Similarly, let $B_{L,t-1}$, $B_{H,t-1}$ be the lowest and highest rates among the downstream consumers of the router N_{t-1} respectively. Since $B_{H,t} \geq B_{H,t-1} \geq B_{L,t-1}$, according to Lemma 7.2.1, we have

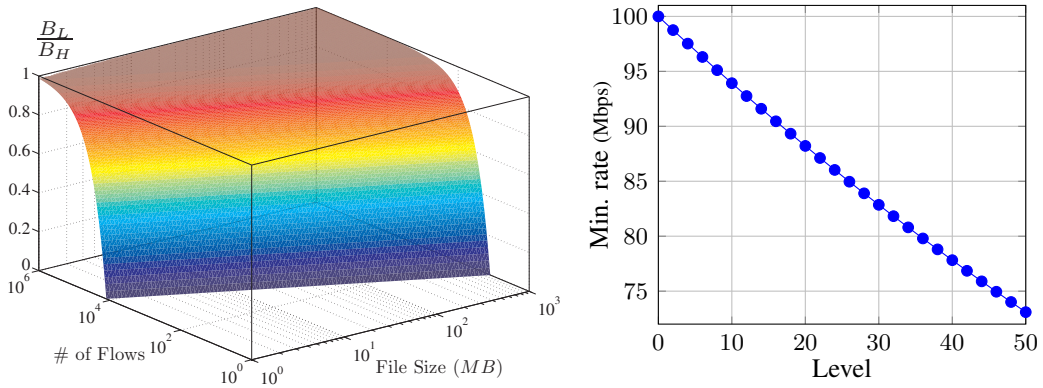
$$\frac{B_{H,t}}{B_{H,t-1}} \geq \frac{B_{L,t-1}}{B_{H,t-1}} \geq 1-\frac{C}{F}. \tag{7.2.7}$$

According to (7.2.6) and (7.2.7), it follows that

$$B_{H,t-1} < B_H \times (1-\frac{C}{F})^{k-2}.$$

By the similar argument, we can show that

$$\begin{aligned} B_{H,t-2} &< B_H \times (1-\frac{C}{F})^{k-3}, \\ B_{H,t-3} &< B_H \times (1-\frac{C}{F})^{k-4}, \\ &\vdots \\ B_{H,1} &< B_H \times (1-\frac{C}{F})^{k-t} \leq B_H. \end{aligned}$$



(a) Relationship among out-of-sync criterias in real router

(b) Min. allowed download rate vs. hierarchical level

Figure 7.5: Difficulty of keeping heterogeneous receivers in-sync.

Since the highest rate of downstream consumers of the router at level 1 should be B_H, i.e., $B_{H,1}{=}B_H$, we now reach a contradiction and the proof is completed. □

Theorem 7.2.2 implies that with the number of levels (k) increases, the gap between the fastest and the slowest receiver can become larger. The design of the in-network cache helps in absorbing the heterogeneity of the receivers.

Now the study shows that out-of-sync is difficult to avoid in an ICN router deployment.

Remark. *The problem of receivers going out-of-sync persists with receiver-driven feedback-based in-sequence congestion control protocols as long as there are heterogeneous receivers.*

The cache size at a router will inevitably be much smaller than the total amount of content available in the network. According to [107], ICN requires a 25TB cache for a 50% hit rate on Youtube data and 175TB cache for a 50% hit rate on BitTorrent data. But [23] suggests that a deployable ICN router (with ∼$1,500 overall hardware cost) would likely have around 100Gb of cache at current costs. Thus, a router cache will be much smaller than the required cache size for the kind of content accessed in current day networks.

For tractability, the study assumes that concurrent flows in a router share the 100Gb cache size equally. The relationship between the bandwidth ratio ($\frac{B_L}{B_H}$), file size (F) and number of flows is shown in Fig. 7.5a. The intersection of the curve with XY plane is where $C{=}F$ (# of flows$=\frac{100Gb}{F}$). The region behind the curve represents the region receivers are out-of-sync.

Note that both the X- and Y-axis are log-scale, which means the area when the receivers are in-sync is just a very small portion of the overall region.

Although a core router might have relatively larger sized cache, the number of concurrent flows on that router is also correspondingly large. The available proportion of cache for each flow is therefore still quite small. If a set of receivers request 20M bytes of data through a core router with 100k concurrent flows, the ratio of rates should satisfy the following:

$$\frac{B_L}{B_H} \geq 1 - \frac{C}{F} = 1 - \frac{100Gb/100k}{160Mb} = 0.99375$$

If B_H can reach 100Mbps, B_L should be $\geq$99.3Mbps. When the hierarchical tree model in Theorem 7.2.2 is applied, the minimum download rate vs.level is plotted in Fig. 7.5b. Even if the network has 50 levels in such a hierarchy, the minimum required download rate is still $>$73Mbps. This is difficult to achieve due to the number of flows that may be multiplexed on a given link.

7.2.3 Existing Solutions for Out-of-Sync Prevention

Although existing NDN transport layer solutions do not support explicit in-sync delivery of data items, to meet user requirements applications like audio/video conferencing [101] tend to achieve it at the application layer. In the initiation phase, the parties of the communication will negotiate an agreement on the data delivery rate (e.g., $n\ pkt/s$). To maintain real-time communication, the end hosts will skip packets if they cannot arrive before they are used. Therefore, instead of requesting for the "next-in-sequence" packets, the end hosts will request only the packets that can possibly arrive in time. The prediction is performed by the applications.

But such solution has two main drawbacks: *1*) It unnecessarily complicates the application, when the underlying communication layer can provide this capability. The communicating parties have to have an agreement before the data starts being delivered. The application also has to predict packet retrieval based on a continuous monitoring of the packet arrival rate and the consumption rate which might vary over time. For applications like file delivery, system updates and etc., the data provider and consumers might find it difficult to determine a proper sending rate given the changing available bandwidth in the network. *2*) The user experience is sacrificed. In audio/video conferencing, when there are missing packets, glitches occur. It is acceptable in conferencing since the quality is sacrificed to maintain real-time communication. But most existing VoD systems tend to wait for (buffer) the missing packets before the video can proceed, which will still result in out-of-sync behavior.

Therefore, to maximize the utility of the network in both file delivery and VoD, and to provide better support for in-sync communication applications, there is a need for a transport-layer protocol that can keep the data receivers in-sync while retrieving data.

7.3 Design Rational of SAID

This section first sums up the requirements for an efficient control protocol for large-scale data delivery and then describes the design rationale of SAID in meeting these requirements.

7.3.1 Protocol requirements

- **Network Efficiency:** Receivers being out-of-sync causes higher overall transmissions, resulting in higher aggregate network load, higher provider load and lower network throughput. An efficient control protocol should allow packets to be delivered *at the first attempt* to more receivers – it is desirable to keep the data consumers in-sync (where possible) when the data provider responds to their requests.
- **Network Friendliness:** Although SAID is targeted for a one-to-many data dissemination scenario, the goal of the design is that all the flows follow the protocol, and are "network friendly". The network friendliness is defined as: the flow should occupy a fair share at the bottlenecks on *each* path from the data provider to data consumers.
- **Reliability:** Due to the difference in the requirements of the applications, the protocol should be general enough to support most data dissemination applications. E.g., file delivery might only need a reliability without being concerned about packets being out of order; VoD would tolerate out-of-order delivery only in a small range (within the play out buffer size) so as to ensure uninterrupted play out; and conferencing applications do not need reliable delivery up to a point.
- **Application Efficiency:** One possible solution to achieve these three requirements of network efficiency, network friendliness and reliability is the data provider send data at the slowest receiver's consumption rate (assume that the network has a way to inform the provider of the slowest receiver and his/her requests). Such solution is not application-efficient – the transmission rate can be dramatically reduced due to one or a few slow receivers. One possible approach to not bring down the overall transmission rate to the slowest receiver is to enable slower receivers to seek alternate means (e.g., get missing packets/repair from other faster receivers). Repairs certainly add to the network load, but the intent is to achieve a tradeoff between network load and session completion time.

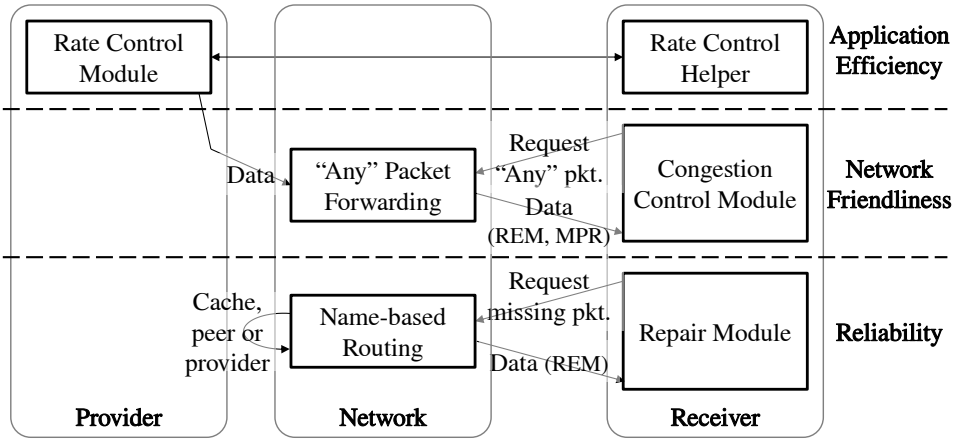


Figure 7.6: SAID overview.

7.3.2 Rationale 1: Decouple reliability from congestion control

Network friendliness is a necessity to ensure the fairness and efficiency of using network across all users. However, the need for reliability and in-order delivery varies across applications. Solutions that couple the congestion control, reliability and in-sequence delivery will either cause overheads (resulting from receivers being out-of-sync) or result in inefficiency (sender waiting for the slowest receiver).

Taking the need to have a general protocol that meets diverse application needs and the heterogeneity of receivers into consideration, SAID decouples the in-sequence reliable delivery from congestion control. The content provider could transmit data at a certain rate (driven by goals of *application efficiency*). To maximize the amount of first-attempt delivery, the receivers should request the packets that can arrive without retransmission (for *network efficiency*). But it is difficult to achieve due to the varying network condition and the transmission rate from the provider. Instead, SAID allows the receivers to request for "any-next" packets and let the network decide which packets can pass through. To ensure *network friendliness*, receivers limit the amount of requests based on the feedback from the network. With this mechanism, receivers may see holes in the sequence of packets. To achieve *reliability*, the receivers can now send new "repair" requests to the content provider, caches or even other receivers, based on the application's requirements.

Fig. 7.6 demonstrates the overview of SAID. By allowing "any packet" going through the network, and the provider try and control the sending rate, SAID tries the delivery of packets the first time to more receivers. Meantime, the receivers also assist the congestion control to ensure network efficiency. The receiver can seek for a repair on receiving data with holes.

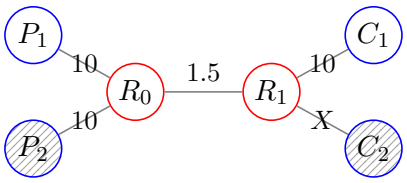


Figure 7.7: Dumbbell topology (bandwidth in *Mbps*).

Even if the in-network caches have flushed out that data, the receiver can still get the data from other receivers who tend to keep it around. With the help of information-centricity, the receivers can seek for (and provide) help with privacy and trust.

7.3.3 Rationale 2: Receiver-driven congestion control

There are two main trends for achieving network friendly congestion-control: end-system assisted solutions and pure network solutions. The class of pure-network based solutions divide the outgoing link bandwidth fairly for each flow going to the *next hop*. Such solutions are conceptually simpler (without addressing scalability and implementation concerns), but there is no way for a flow which has a limit downstream to indicate a reduced demand. This results in inefficiency, just as was suggested by [108]. Consider a Dumbbell topology (see Fig. 7.7) with 2 flows ($P_1 \rightarrow C_1$, $P_2 \rightarrow C_2$) and the routers (R_0, R_1) are using per-flow fair queueing. There is no control exercised through feedback by end hosts. R_0 will allow $0.75Mbps$ for each flow. Such a solution will be efficient when X is also $10Mbps$. But the problem occurs when X is smaller than $0.75Mbps$. E.g., when X is $0.1Mbps$, R_1 will have to drop $0.65Mbps$ on the interface towards C_2 and the overall throughput of the network will drop to $0.85Mbps$, compared to the $1.5Mbps$ in the ideal case. However, an end-system assisted solution on C_2 will demand less bandwidth (through a lower request rate) and R_0 can allocate the remaining throughput to the $P_1 \rightarrow C_1$ flow.

SAID chooses to use an end-system assisted solution (receiver-driven) to avoid such inefficiency. All the receivers maintain a window of unsatisfied "any-next" requests for packets. This window reflects the maximum number of packets that can be in flight towards this receiver. SAID could seek to adapt this, just like TCP using an AIMD approach so as to find the capacity from sender to receiver, but there are still two major issues: *1*) reception of out-of-sequence packets should not be treated as an indication of congestion, since the receivers are asking "any" packets, and *2*) faster receivers might have a very large window due to the absence of congestion experienced in their path. Because of these considerations, new mechanisms are required to achieve congestion control and fairness. SAID uses *Random Early Marking* (REM) [109] as an indication of network congestion and uses a counter – *Minimum Pending Request Count* (MPR) – as an indication of whether the receiver is using

a window (of outstanding packets) that is larger than the number that can be accommodated in the path. It uses a slightly modified AIMD window adaptation according to REM and MPR to achieve congestion control. The policy followed by receivers is described in detail in §7.4.

7.3.4 Rationale 3: Achieve reliability via efficient repair

SAID fills the required holes in the sequence via a repair mechanism. ICN provides an efficient way to achieve sequence-specific data retrieval via the PIT and the Content Store. ICN also provides a fundamentally important and desirable capability of ensuring *privacy* of the receivers, without the need to *trust* other receivers (especially when issuing a request for repair from other receivers) compared to the existing IP network. In addition, the network has more information about the receivers downstream. SAID optimizes the FIB propagation to redirect the repair requests towards nearby peer receivers who have already received those packets that are missing at the requesting receiver. This achieves lower network load and lower content provider load as well. These issues are covered in §7.5.

7.3.5 Rationale 4: Application-specific data releasing rate

Different applications have different transmission rate requirements. Live streaming and conferencing applications need a certain transmission rate based on desired video/audio quality. Other applications do not have a fixed demand on the rate, but might have preference on either to satisfy most of the receivers or seek to satisfy the fastest receivers (e.g., BitTorrent). For those applications, the transmission rate of the provider is a tradeoff between the (network, provider) load and the content delivery completion time. A high sending rate could shorten the receive period on the faster receivers, but also result in higher retransmission rate in the network to reliably deliver to the other receivers. The application, the provider or the receivers might face higher charges from the ISP as a result of such retransmissions. A low sending rate may seek to operate at the opposite end of the spectrum – sacrificing completion time for lower network load.

As a selective component at the application layer, SAID also accommodates a mechanism to enable data providers to determine this balance. Due to the varying network environment and the receiver situation, it is difficult (and undesired) for a data provider sending at a constant rate. The provider might want to satisfy a certain part (e.g., 60%) of the receivers first. Therefore, instead of specifying a constant sending rate, SAID allows the provider to pick an *ACKer* from the receivers to pace the flow similar to pgmcc [48]. But, unlike pgmcc, the ACKer does not have to be the slowest receiver. The ACKer selection component is covered in §7.6.

7.4 In-Sync Receive Mechanism

This part first describes the mechanism for the receivers to control the request rate for data packets in a fair and efficient manner and the corresponding changes to the forwarding engines. The resulting critical challenge of identifying and responding to network congestion is then discussed and solved.

7.4.1 Receive "Any" Packet from the Provider

To solve the out-of-sync problem, SAID needs a slight modification to the existing communication model from the receiver asking for a specific packet (as is currently done by NDN and even TCP, etc.) to instead ask for "any" incoming packet. This subtle change forms the core of this solution, but does present fresh challenges. This modification to respond to a request with "any" packet of that content instead of the "exact" packet enables the network to deliver as many packets as possible in the first attempt.

In order to retain the existing architectural framework of NDN and COPSS as much as possible, SAID can either modify the Interest packet in NDN or the Subscription packet in COPSS. For Interest packets (i.e., prefix, selector and nonce in TLV format), SAID set the prefix as the name of the flow and the `Exclude` field indicating that this is a request that seek any of the next n packets with the prefix. For Subscription packets (i.e., CD, selector and nonce in TLV format), SAID should set the CD as the name of the flow and only allows n Publication to return. Such a request is in the middle of an Interest and a Subscription: It is an Interest for any next packet (without specific name), and a Subscription with lifetime of only 1 Publication. The remaining part of this section will see the request as an Interest for the simplicity although the Subscription works the same way.

These requests could accumulate at the routers, i.e., when a receiver sends multiple such requests for a same prefix to the network, a `Data` packet should only consume one of them. On receiving such a request, an upstream router places the prefix, the incoming face (the same logic as what NDN has now) and a *Pending Request Count* (PR) into its PIT. PR is incremented on receiving a request packet with the same prefix, and is decreased when a Data packet is forwarded towards that face. The router does not utilize its Content Store (cache), so as to avoid sending duplicate packets downstream. The routers propagate the maximum of the PR upstream until this propagated value reaches the 1^{st} hop router next to the data provider. The implementation is more memory efficient compared to the "exact" packet requests currently used. For multiple requests for the same prefix, SAID only consumes 1 entry and 1 counter unlike the need to maintain multiple entries in the existing approach.

SAID retains the "flow balance" suggested by NDN, in which every Interest packet will

have at most one Data packet returned. Therefore, receivers can maintain a window (explained in §7.4.3) to control the portion of the flow they are going to receive. Note that although each receiver has a congestion window, the sending rate on the provider side is not controlled by this window (similar to a publisher) as the sender seeks to transmit to heterogeneous receivers. The difference between the sending and receive rate naturally results in the receiver missing some of the packets in the sequence ('holes in the sequence"). Although the receivers are now skipping packets in the sequence due to the limit of the bottleneck bandwidth in their respective paths, instead of causing the provider to transmit slowly, they essentially continue to receive the packets (i.e., keeping in pace in as much as their bottleneck will permit, but not necessarily receiving every packet) from the provider. Therefore, the network is able to maximize the utility of every packet sent by the provider. This results in reducing the provider load as well as the network load.

7.4.2 Identifying Network Congestion

The change in the communication logic of requesting "any packet" instead of the "next in sequence" packet causes a challenge for congestion control – gaps in the sequence number of received packets is no longer an indication of queue/buffer overflow (or congestion) on the routers. The network needs to provide a separate indication of congestion.

REM [109] is used at the intermediate routers to provide end systems an indication of network congestion. A simple AIMD mechanism works acceptably *on the receivers* that have available bandwidth below the sending rate of the provider. The introduction of REM into NDN can also simplify the existing congestion control solutions and help avoid any unnecessary bufferbloat (an issue frequently arising in the current Internet [110]).

7.4.3 Window Control on the Receivers

The multiplicative decrease on seeing REM can work well in the competition scenario but a simple additive increase mechanism is not enough when the path to the receiver has high bandwidth. Unlike TCP or ICP, the transmission rate on the provider is not controlled by the receivers' congestion windows. The receivers with available bandwidth (or a fair share on a multiplexed link) higher than the transmission rate will not see any marks and their window will continue to grow to a highly inflated value (especially if there is no limit). This growth in the window essentially causes a large number of pending requests to be queued at the upstream router. These pending requests will allow a large number of packets to be delivered in a burst towards the receiver when the provider decides to increase transmission rate or new contention occurs on the path. The large build up of a queue at the bottleneck router also increases the feedback delay. Thus, for the period when a burst of packets

is transmitted from a buffer that has high occupancy, those packets will be marked, thus causing the receiver to react and reduce the window. This form of "bang-bang" effect has the potential to under-utilize the network or cause unfairness between flows over a subsequent interval.

While an excessively large window might cause inefficiency and unfairness, a very small window might also lead to inefficiency because a short burst caused by transient congestion might consume all the pending requests and result in unnecessary packet loss. Therefore, while deciding if the receiver should increase window, SAID needs to avoid: *1*) too many pending requests in the path, *2*) all the pending requests consumed by a short burst, and *3*) unnecessary increase due to transient changes.

Although all the receivers have the same purpose – control the pending requests, the exact maximum window size can be different due to the differing *Bandwidth-Delay Product*s (BDPs) of the receivers. SAID decides on the window increase based on an observation over a period of time on the instantaneous minimum pending request count on the path. The network piggybacks the instantaneous MPR while data packets are forwarded towards the receivers. Note that different receivers can see different MPR even for the same data packet since MPR appears at the branching point (the router having other outgoing faces that are receiving faster than the current receiver). A receiver should observe all MPRs in a window and ensure: *1*) these values are as small as possible (to avoid too many pending requests in the path), *2*) the *Minimum MPR* (MMPR) is larger than a small value (to avoid all the pending requests consumed by future short burst), and *3*) compare to the MPRs in the previous window (to avoid fluctuation).

To simplify the calculation, SAID decides the window increase on MMPR alone. It will stop increasing the window when MMPR is larger than X and only increase window when it can rule out the possibility of reacting to a transient.

7.4.4 Implementation of Receiver Window Control

7.4.4.1 Network support for MPR delivery

To control the congestion window size on the receiver side as was described above, SAID needs the cooperation from the network. The intermediate routers need to piggyback the instantaneous PR with the data packets similar to REM. The PR equals the current value of the counter in PIT to accumulate such "any-next" requests.

In order to allow the receivers to get MPR in the whole path, an MPR field is added in the Data packet. Data provider should initiate this field as $+\infty$. The intermediate routers

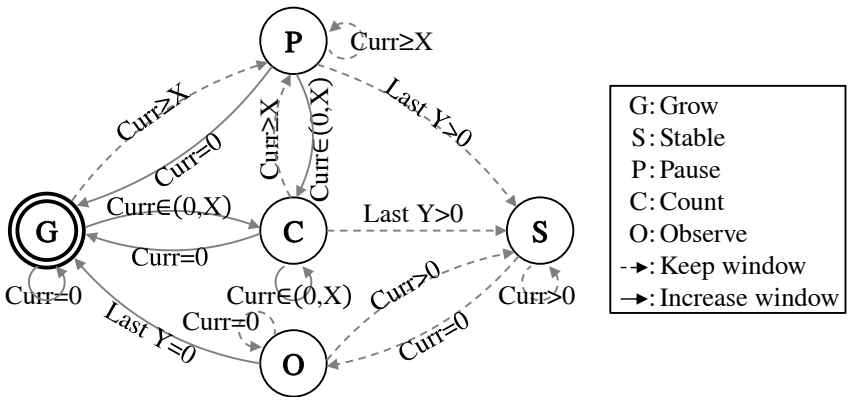


Figure 7.8: State machine for receiver window increase.

forward the packet based on PIT and set $MPR \leftarrow min(MPR, PR)$. Note that the PR here is PIT count for the flow *towards the interface* of the the data packet. If the router needs to replicate the data packet, the MPR on each packet might be different.

To ensure the integrity of MPR (if necessary), the intermediate routers (and the data provider) can sign the MPR separately without touching the content of the packet. The intermediate routers can use the private key of the ISP and the receivers can check the signature via the ISP's public key.

For a receiver (C), the minimum instantaneous PR always appears on the branching point of C ($Br(C)$). According to the definition, on the router $Br(C)$, there exists other outgoing face(s) that can receive faster than C, or it is the 1^{st} hop router of the data provider if the C can receive at a rate faster than the sending rate. In Fig.7.2, $Br(C_1) = R_3$ since R_3 has subtree R_8 that can receive at $2Mbps$. Likewise, $Br(C_2) = R_1$. The routers below $Br(C)$ might see fresh requests sent from C and their PR should always be larger than or equal to the PR at $Br(C)$; The routers above $Br(C)$ would see more pending requests from another interface and therefore the value should be larger than or equal to the PR at $Br(C)$ also.

7.4.4.2 Receiver decision based on the MPR

On receiving the data packets with the instantaneous MPR value, the receiver picks the smallest one in a window (MMPR). SAID uses a state machine (Fig. 7.8) to keep track of how long the MMPR on a receiver is in a particular state ($=0$, $<X$, or $>X$) and decide the size of the next window. X is a parameter that decides the maximum number of pending requests that are used to absorb a burst. $X = 5$ works well across different evaluations performed. The receivers adjust their congestion window based on the following rules:

1. Every receiver starts in the "Grow" state (G) and stays in G as long as $MMPR = 0$. $MMPR = 0$ means the data from the provider can always consume all the pending requests, and therefore the receiver should increase the window size.

2. Whenever MMPR grows above X, the receiver should stop increasing the window immediately.

3. On seeing an REM, no matter which state it is in, a receiver should clear all the MMPR counters, reduce the window size by half (multiplicative decrease) and go to G (not shown in Fig. 7.8).

4. MMPR can fluctuate even when the window size is large enough due to a short burst of packets from the provider. A Stable state (S) is therefore defined as the receiver sees $MMPR > 0$ for more than Y windows. Window should stop increasing in state S.

5. An intermediate Counting state (C) is introduced to count Y windows. While in state C, the receiver keeps increasing the window until the count Y is reached (to S) or the MMPR grows larger than X (to P).

6. When the MMPR drops to 0 in S, it might indicate that one of the two conflicting scenarios is happening: either the publisher increases the sending rate and thereby consumes the pending requests, or there is a competitor on the bottleneck that causes the receiver to send requests slower. The receiver should increase the window in the first case while decrease it in the second. SAID chooses to wait for Y windows before increasing the window. If a competitor shows up, the receiver should see marks and operate based on rule 3.

The parameter Y reflects the stability of the mechanism. A larger Y means the longer a receiver waits on seeing $MMPR = 0$, it can be beneficial in the presence of a competitor, but will under-utilize the network when the provider increases the sending rate. Both X and Y are set to 5 in implementation.

7.4.4.3 State Transition Example

An example is used to trace the MMPR and the window increase decisions to aid in understanding how the scheme works. The MPR, window size and the state transitions in Fig. 7.9 are shown for a receiver with $2Mbps$ available bandwidth. Every cross represents a decision made at the end of a window. The MMPR grows with the window size after $1sec$. 4 windows later, the MMPR reaches 5 and the receiver stops increasing the window. The MMPR stays at around 5. Although the MMPR becomes lower than 5 for several windows at around

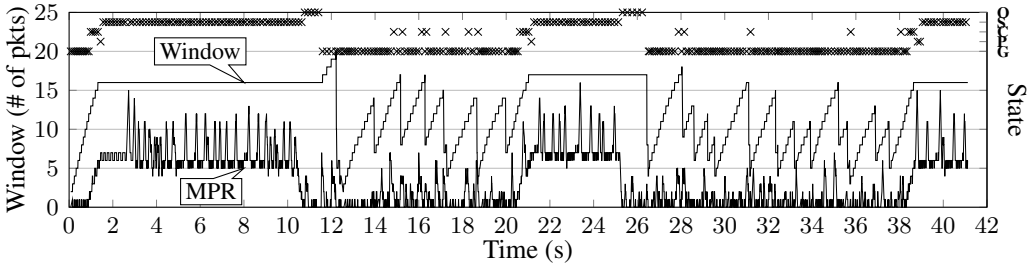


Figure 7.9: Result of window control at the receiver using state machine.

4*sec* (due to the transient changes) the window size is not increased. When there is new contention at the bottleneck link (additional competing flow) occurring between 10*sec* and 25*sec*, MMPR drops to 0 and the receiver then starts to increase the window. The receiver receives packets marked as a result of REM at 12*sec* and then to perform normal AIMD to respond to the contention introduced by the competing flow. The fairness of SAID is shown in §7.7.1.

7.5 Efficient Repair

For applications that need reliable delivery (as is often the case), SAID depends on the application interface to retrieve missing packets. Applications should be able to determine which packets are of higher priority. E.g., applications like VoD may prioritize the packets within the play-out buffer. Applications like file transfer do not need such a receiver play-out buffer and they can perform repair at any time for any packet (but usually the earliest in sequence first). The request for a missing packet is done via standard NDN query/response which ensures privacy and trust.

7.5.1 Repair via Data Provider

Every data provider has to propagate (a prefix of) the flow to the network before transmitting packets of the flow. When a data consumer requests a repair with the ContentName `/flowID/segID`, the routers will then forward the packet according to the FIB and the request will eventually reach the data provider. E.g., in Fig. 7.10, the data provider P is disseminating a flow named `/dissertation/SAID.pdf/_v1`. Before starting the flow, (s)he propagates the information that (s)he serves the prefix `/dissertation/` (the dotted arrows in the figure). Data consumer C_2 is behind a congested link (R_5 to R_7) and therefore sees losses. When C_2 requests for a missing packet `/dissertation/SAID.pdf/_v1/_s20`,

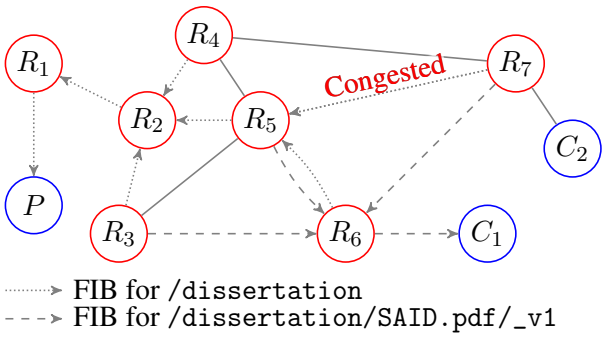


Figure 7.10: Example of different kinds of repair.

(s)he sends an Interest with that name. Note that the repair requests can also be aggregated, when more than one receiver in the subtree experienced a loss of the same packet.

Although this solution is relatively simple, it has a drawback. The repair from the provider might share the same path with the ongoing transmissions from the content publisher. In Fig. 7.10, when the repair comes back from P to C_2, it sill needs to traverse through the congested path between R_5 and R_7. This data packet might either get dropped or compete with the fresh data. Therefore, this solution is kept as a backup and it will only be used when the bottleneck is very close to the receiver and there is no other way to get data. Moreover, this solution is only useful if the repair is performed after the whole flow is finished.

7.5.2 Repair among Data Consumers

Similar to traditional ICN, in addition to the data provider, all the end hosts in SAID that have (part of) the data can support repair requests. But the difficulty lies in identifying and routing the requests to the receivers who have received those missing data packets earlier. Moreover, since the repair packet has to share the bandwidth on the dissemination tree, independent of whether the repair request is delivered from a cache in the network or the provider, the repair has the potential to aggravate the congestion in the bottleneck link from the provider to the receiver. Therefore, this work proposes a local FIB propagation optimization that can potentially mitigate the impact on an already congested link.

With SAID, after receiving (part of) the data from the provider, the receivers should flood the prefix of the data over a limited number of hops. E.g., in Fig. 7.10, after receiving the packets, C_1 propagates prefix `/dissertation/SAID.pdf/_v1` in a 2 hops range. When C_2 is requesting for `/dissertation/SAID.pdf/_v1/_s20`, R_7 will forward the request to R_6

and eventually with get a response from C_1. The repair can then bypass the congested link and increase the overall throughput to C_2.

7.5.3 Prefix Granularity

The simple mechanism described above might still face the issue of inefficiency. Whenever C_1 propagates the prefix, routers will forward repair requests from nearby nodes to him/her (instead of going upstream). Therefore, for efficiency, C_1 can only propagate the prefix when (s)he receives *all* the packets in the flow, so as to be able to respond to the repair requests. Such a propagation is useless for applications like VoD that needs repair during the flow.

One variation of FIB propagation can be: C_1 propagates the exact ContentName of each packet (s)he receives. The requests from slower receivers can then be redirected to C_1 immediately. This solution benefits the applications that need in-flow repair, but it puts a huge overhead on the FIB since every packet will result in a FIB entry in the nearby routers.

Therefore, to achieve a balance between the goals of repair efficiency and reduced FIB size, SAID suggests that the data provider should group n packets into a "chunk" and replace the `segmentID` in the ContentName with `chunkID/segmentID`. E.g., if the publisher uses n=100, the name of the packet with segmentID=205 should be `/dissertation/SAID.pdf/_v1/_c2/_s205`. This will not affect the basic solution since every packet still has a globally unique ContentName. But C_1 can propagate `/dissertation/ SAID.pdf/_v1/_c2` after receiving packets 200-299 either via the initial transmission or via repair. The FIB entries created will be much smaller and at the same time the other receivers will be able to get repairs from receivers much earlier. The chunk size can be specified by the data provider based on the timeliness requirement of the flow. If the flow has a very tight deadline, the chunk should be smaller so that the receivers can achieve their repair earlier while the flows with larger size and those that don't have a tight deadline can set the chunk size larger to reduce the FIB size.

7.5.4 Privacy and Trust

Similar to peer repair solutions [49–52], two major concerns with "peer receiver-based repair" are privacy for data consumers and the integrity of the data received from a peer [55]. For some sensible flows, especially some business flows, the data consumers don't want to reveal the identity among each other since some consumers don't want the others to know that they are interested in the specific topic. Also, when the data comes from a peer, not the data provider himself, the consumer is difficult to check the integrity of the data.

In IP (location-based) networks, the data consumer in most cases will reveal his/her own location (identity) while communicating with a peer (except those that explicitly choose to remain anonymous, e.g., by using Onion Routing). Some other solutions (e.g., multicast) [47,48] require every receiver to multicast a NAK to a whole group on seeing a packet loss. This results in the data provider eventually having to align to the slowest data receiver and a substantial bandwidth is wasted on retransmission of packets to users that have already received that packet.

ICNs give these applications a natural way to achieve privacy and trust. Since every data receiver requests repair via a Content Name, the receiver doesn't have to reveal his/her identity. It is difficult for a peer to know who is actually making the request. Even for peers that perform the local FIB propagation, they propagate the prefix of the flow (or the chunk) rather than their own identity. Thus privacy can be maintained. Every Data packet in ICN has a key digest of the data provider and a signature for the data content from the provider. The data consumer can easily verify the integrity of the packet no matter if it is received through repair from other receivers or from the provider.

7.6 Provider Transmission Rate Control

The data-provider's sending rate is typically determined by application requirements. Applications like audio/video streaming need a certain minimum rate for proper audio/video progression and might also have a limit on the maximum rate. However, certain other applications such as file delivery are more elastic and therefore the sender can find a balance between the average completion time and the network load (as illustrated in Fig. 7.13b).

For general-purpose data transmission, SAID suggests that the data provider should send at the rate that the *majority* of the receivers can receive. Therefore, those consumers can receive all of the data transmitted without much repair, while the slower receivers (the *minority*) will depend on the other receivers for the missing packets. However, the estimation of the receive rate of the majority is not easy either, because of the highly dynamic nature of network. Therefore, instead of *estimating* the receive rate for the majority directly, SAID divides the task into two parts: 1) find a proper receiver (normally the slowest in the majority) as an *ACKer* and 2) *align* the content provider's sending rate to the ACKer's available bandwidth. Unlike pgmcc [48], the ACKer the provider picks is the slowest in the majority, not the slowest among *all* the receivers. This allows SAID be more efficient than pgmcc because SAID exploits the efficient repair capability in ICN. Note that SAID can also align to the slowest receiver if the application demands it, but by having the ability to also align to a constant rate or to that of the majority, SAID is more flexible and can support different types of application needs.

7.6.1 ACKer Selection

A critical component of the mechanism is the selection of the ACKer. Due to the dynamic nature of the network, the slowest receiver in the majority might change from time to time. Therefore, the provider should have the ability to change the ACKer based on network conditions. But compared to the per-RTT estimation of sending rate, the ACKer selection can be made over a relatively longer time scale. The provider initiates the ACKer selection periodically.

Unlike some of the IP multicast protocols, SAID does not adopt a session-initiation phase. Instead, it relies on the feedback sent by the receivers to the first packet in order to elect the initial ACKer. Periodically, the data provider triggers the collection of receiver statistics, by piggy-backing a trigger packet containing the statistics collection range (start segment ID and end segment ID) for a period of time (e.g., a window).

The number of packets received can be used to estimate the current receive rate. But receivers with an available bandwidth higher than the ACKer will all have the same receive count. Therefore, a second value is taken into consideration: the number of received packets that were marked. Faster receivers will receive fewer marked packets compared to the ACKer. Different receivers might have received a slightly different number of packets, due to queueing at routers. To better reflect the receive rate, the last received packet is also considered. On receiving the statistics from the receivers, the provider calculates the score of every receiver as

$$S = \frac{ReceivePktCount - 0.5 \times MarkPktCount}{LastReceiveID - RangeStart}. \tag{7.6.1}$$

A marked packet is valued half as good as a normal packet, so as to space out the faster receivers. The receiver at rank R will be elected as the new ACKer. However, changing the ACKer frequently might affect the stability of the flow and therefore SAID only elect a new ACKer if the current ACKer is not in the range $R \pm R_{th}$. But when the condition in the network or the receivers changes dramatically, a shift on the ACKer is inevitable.

Selecting a slower ACKer than the current one is relatively easy. Using Eq. (7.6.1), the provider can select a slower receiver that has a lower receive count and higher mark count in proportion to their available bandwidth. But selecting a faster ACKer is not straightforward since the new ACKer should be among the faster receivers who all receive at the same rate as the current ACKer. The mark count in Eq. (7.6.1) will take effect in this case. Although the faster receivers have the same receive count as the ACKer, they have a lower mark count. If the new ACKer happens to be too fast, the provider can easily find a slower one based on the slower ACKer selection procedure. The provider can gradually find a proper sending rate using this mechanism.

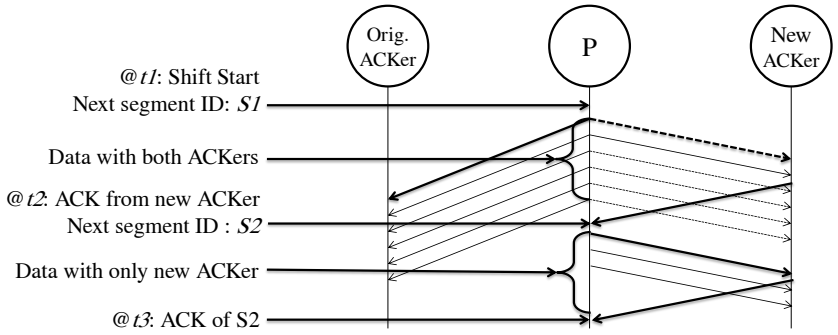


Figure 7.11: Procedure of ACKer shift.

7.6.2 ACKer Logic

Since every packet sent from the data provider will be received by multiple receivers, the data provider places the ACKer ID in every data packet to assign a proper ACKer for that packet. On receiving a packet with his own ID, that receiver sends an ACK back to the data provider. Using a congestion control model like TCP, a closed feedback loop is thus formed between sender and the ACKer to align the sending rate to the ACKer's receive rate.

Unlike the communication model of the other receivers, the rate is controlled on the sender (content provider) side for the ACKer (and thereby the 'faster' receivers forming the majority). The ACKer is likely to receive *all* the packets for that flow. This is achieved by either modifying Interest to a standing request (which will not be consumed by Data packets) or using a Subscription directly.

7.6.3 ACKer Switching Policy

To ensure a reliable switch between the new and the old ACKer, SAID uses an in-band notification procedure. The procedure is shown in Fig. 7.11. When the data provider decides to shift to a new ACKer at time $t1$, (s)he starts to send packets indicating that both the original ACKer and the new ACKer should provide feedback (ACK). The new ACKer may miss some of these packets especially if (s)he has a lower available bandwidth (the dashed lines on the right). But the new ACKer will eventually receive a data packet with his ID in the ACKer list. This is the first time the new ACKer sees his ID in the ACKer list. He/She will first send out a standing request as described in §7.6.2, and then send an ACK to the packet from the data provider. Both the standing request and the ACK will follow the same path towards the data provider. So at time $t2$, when the data provider receives the ACK from the new ACKer (segment number might be larger than $S1$), the provider can be sure

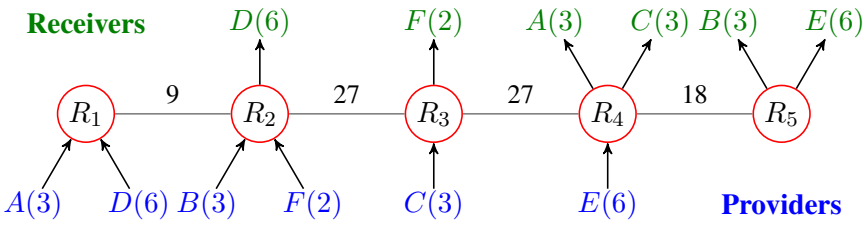


Figure 7.12: Generic Fairness Configuration (GFC).

that the new ACKer has a standing request ready on the path. From the next packet onwards (segment number $S2$), the provider sends data with only the new ACKer's ID.

For packets with segment number between $S1$ and $S2$, the new ACKer's ACKs are not reliable since during that period, the new ACKer can miss packets because the standing query is not ready yet. Receiving an ACK from the new ACKer for $S2$ ($t3$ in the figure) might also not indicate the end of the shift since the original ACKer might have a longer latency than the new one. To better control the provider window during this phase, the provider assigns every packet an ACKer. For packets in range $[S1, S2)$, although both ACKers might receive and send ACKs, the provider only respond to the original ACKer. The new ACKer's ACKs in that range will be discarded by the provider. For packets with segment number $\geq S2$, the provider will respond to the new ACKer.

7.7 Evaluation

This section studies the behavior of different components in SAID with simulations. SAID is evaluated in both file delivery and streaming applications. Two related mechanisms are compared with SAID: *1*) A modified ICP solution that utilizes REM. While most of the proposed approaches use packet loss as an indication of congestion, this work resorted to enhancing ICP with REM so as to have a reasonable comparison with SAID in a network with heterogeneous receivers. *2*) pgmcc [48] as an example of the solution that aligns the data provider transmission rate to the slowest receiver. It also uses a window-based congestion control to ensure the whole dissemination tree is fair to the bottleneck on the slowest path.

7.7.1 Evaluation of Fairness among the Receivers

The fairness of SAID's receiver-driven congestion control mechanism is illustrated via *Generic Fairness Configuration* (GFC) [111] which comprises multiple bottlenecks result-

Table 7.1: Fairness result in GFC.

Group	Flows	Hops	Fair	SAID		ICP	
				Avg.	$\frac{\text{Avg.}}{\text{Fair}}$	Avg.	$\frac{\text{Avg.}}{\text{Fair}}$
A	3	5	1	0.7347	0.7347	0.7245	0.7245
B	3	5	2	1.2352	0.6176	1.2875	0.6437
C	3	3	6	6.5399	1.0900	6.4909	1.0818
D	6	3	1	1.1284	1.1284	1.1363	1.1363
E	6	3	2	2.3653	1.1826	2.3522	1.1761
F	2	3	9	9.0665	1.0047	9.0427	1.0047

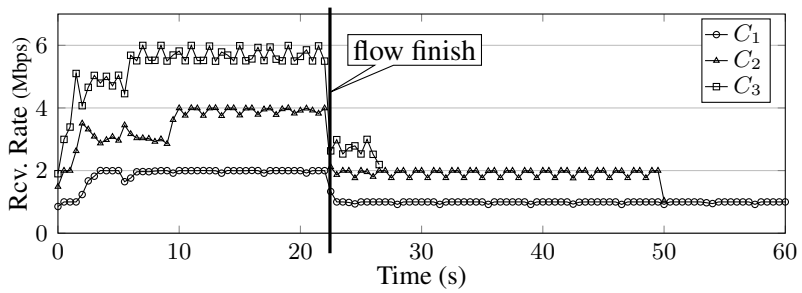
ing in different fair rates for the different flows in the network. The topology and the link capacity (in $Mbps$) is shown in Fig. 7.12. There are 6 flow groups ($A-F$) each with different # of flows (in the bracket after the group name). The ideal fair throughput of the flows ($fair$) are listed in Table 7.1 (in the "Fair" column).

Since SAID uses a receiver-side congestion window to control the receive rate, a feedback loop is formed between the receiver and the branching point. The change in the congestion window does not affect the other receivers or the data provider. Therefore, every "flow" in GFC can be seen as part of a dissemination tree (a path from provider to receiver). The data providers transmit data faster than any of the receivers' rate (at $27Mbps$). After ignoring start up effects, this work determines the average throughput of each of the flows in the subsequent $15sec$ period and compare it to the ideal value. ICP is similarly evaluated, as a comparison.

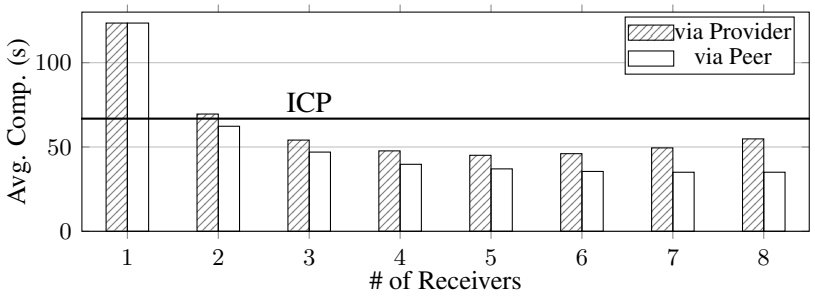
The average throughput and the fairness ratio ($\frac{avg.}{fair}$) for each flow with the two protocols are listed in Table 7.1. Similar to all the feedback window-based congestion control mechanisms, SAID slightly benefits the flows with shorter RTT ($C-F$). Similar results are observed for ICP. In general, SAID performs similar to ICP (and all the other window-based congestion control mechanisms) from a fairness standpoint. SAID would also be fair to protocols using AIMD on *each* path of the dissemination tree separately.

7.7.2 Evaluation of SAID Repair Protocol

A dissemination tree rooted at the provider P_1 (shown in Fig. 7.2) is used to illustrate the efficiency of the repair mechanism. In addition to connectivity to P_1 over the binary dissemination tree, every data consumer also has a connection to a faster consumer with a link between the corresponding first hop routers. To limit the capability of such inter-router links, the bandwidth on these links are set to the bottleneck bandwidth that the receiver



(a) Receive rate on slow consumers (P_1 send at $6Mbps$)



(b) Average completion time with different sending rates

Figure 7.13: Repair efficiency.

has. E.g., C_1's bottleneck bandwidth is $1Mbps$, therefore the link between R_7 and R_8 is also $1Mbps$.

7.7.2.1 Individual throughput with repair

In this simulation, P_1 sends a total of $120Mb$ data to all the receivers at $6Mbps$ (a balance between the receive rates of different receivers). The instantaneous throughput on C_1 to C_3 is shown in Fig. 7.13a. C_1 has a $1Mbps$ bottleneck from P_1. For the first 1.5 seconds, C_1's throughput is $1Mbps$. After that, C_1 can get repairs from faster receivers, and its throughput reaches $2Mbps$. The throughput on C_2 shows almost the same trend. But since C_1 and C_2 share the link between R_8 and R_9, the repair from C_3 to C_1 affects the performance of C_2. The overall throughput on C_2 is only $3Mbps$ when it cannot satisfy all the requests from C_1 within the first 10 seconds. The throughput of C_4 and C_5 are close to $6Mbps$ since they can get $4Mbps$ and $5Mbps$ from the provider directly and the remaining from C_6.

7.7.2.2 Overall efficiency under different sending rates

The overall completion time for three repair mechanisms (repair via provider, peer repair and ICP) is evaluated with different provider sending rate. The average completion time is

Table 7.2: Stall time (s) in streaming demo (video length=40s).

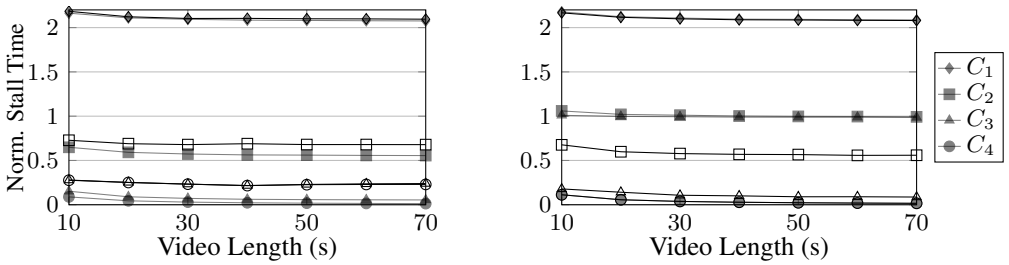
	C_1	C_2	C_3	C_4	Rep.
Baseline	83.384	22.507	2.461	0.886	–%
ICP	84.166	27.570	8.656	8.645	–%
pgmcc	84.804	84.770	84.768	84.767	0.00%
SAID-F	83.821	40.062	39.569	1.131	12.44%
SAID-S	83.541	22.754	4.010	1.123	21.91%
SAID	44.304	1.271	1.151	1.131	12.44%

shown in Fig. 7.13b. When the data provider aligns the sending rate to the slowest receiver ($1Mbps$, like pgmcc), no repair is needed. But the average completion time is even higher than ICP (as shown in Fig. 7.3b, most of the receivers can receive at around $2Mbps$). When the sending rate becomes higher, the average completion time reduces. The peer repair solution helps receivers get repair packets even before the flow is finished. Therefore it always has a lower average completion time than the "repair via the provider" solution. The completion time reduces below ICP when the sending rate is above $2Mbps$, with reduced provider load (SAID retransmit rate is less than 2 compared to 3.36 for ICP). As the sending rate increases, the peer repair reduces the load on the provider, compared to both of the other solutions, albeit not necessarily reducing the average completion time significantly.

7.7.2.3 Repair for streaming

A streaming video application with a playout rate of $3Mbps$ is used in this simulation to evaluate the effectiveness of various approaches by considering the metric of video *Stall time*, to reflect the impact on user experience. No stalling (i.e., no interruptions) occur when there are no holes in the sequence in the play out buffer for the next $1s$ of playback. Simulations with only one consumer requesting the video is taken as the baseline. SAID-F represents the repair by the provider at the end of the flow (similar to "download-and-play"). SAID-S is repair by the provider while streaming. SAID denotes the peer-assisted repair solution. These solutions are compared on the tree topology (Fig. 7.2) to ICP and pgmcc.

Table 7.2 shows the stall time and the repair ratio ($\frac{\text{\# of pkts via repair}}{\text{\# of total packets}}$, Rep in Table) for a $40s$ video. C_5-C_8 are similar to C_4, and are not shown. When all receivers request the video simultaneously in ICP, they go *out-of-sync* soon thereafter. The stall time for C_3-C_8 (some not shown) deviates above the baseline. With pgmcc, since the provider has to align with the slowest receiver (C_1), the stall time for the rest of the receivers is up to $80s$, higher compared to the baseline. The user experience for the faster receivers therefore deteriorates considerably.



(a) Baseline(gray) vs. NDN (black)

(b) SAID-F(gray) vs.SAID-S(black)

Figure 7.14: Repair efficiency for streaming with different video size.

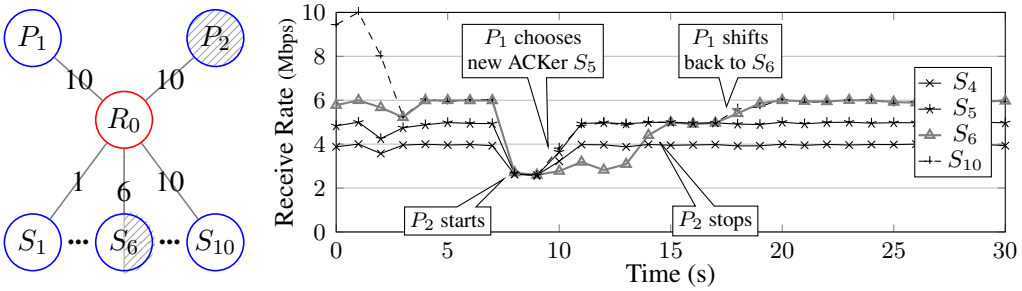
Although SAID-F achieves a relatively low repair rate, the stall time for C_1-C_3 remains high, because the repair is performed after the flow finishes. C_2 and C_3 benefit from the repair during streaming in SAID-S. But this benefit comes at the cost of higher network load. The repair ratio of SAID-S is higher than SAID-F ($\sim 22\%$ vs.$\sim 12.5\%$) since the retransmission has to go through the bottleneck link and affects the primary "any packet" stream. But the retransmission rate from the provider of SAID-S is still lower than ICP (~ 1.9 vs.~ 3.4, not shown) which means SAID-S consumes less network and provider resources. SAID (peer-repair) however is superior as it is able to utilize the extra bandwidth between end-hosts. The repair does not affect the multicast session and the slower consumers can get double the bandwidth compared to SAID-F and SAID-S. Despite the repair, the stall time for the slower receivers ($C_1 - C_3$) are much smaller than with the other solutions, even though they are playing the video at the same rate of $3Mbps$.

The application is further evaluated with different video lengths. *Normalized stall time* ($\frac{\text{stall time}}{\text{video length}}$) is used as the comparison metric. The result is shown in Fig. 7.14. The normalized stall time does not vary much with the video size. The results in Table 7.2 still hold no matter how large the video is. Similar results were observed for the repair percentage.

7.7.3 Evaluation of ACKer Selection

7.7.3.1 Competition on the ACKer

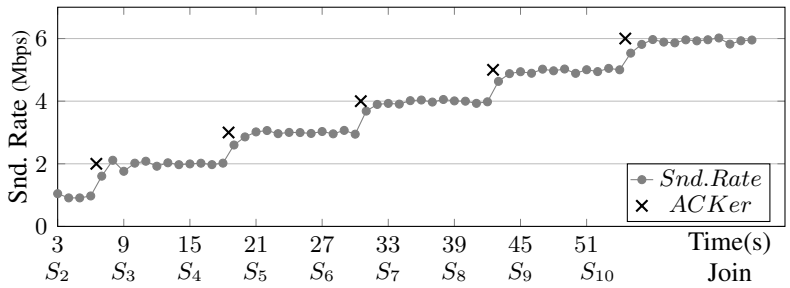
A synthetic topology (Fig. 7.15a) is used to demonstrate the correctness of the ACKer selection logic. 10 consumers S_1-S_{10} are hanging on the 1^{st} hop router with two data providers P_1 and P_2 on the other side. Consumer S_i has an available bandwidth of $iMbps$. Both the publishers have enough bandwidth to support the fastest receiver. Data providers in the simulation set the transmit rate requirement to be in the bandwidth range of $55\% \pm 5\%$. This



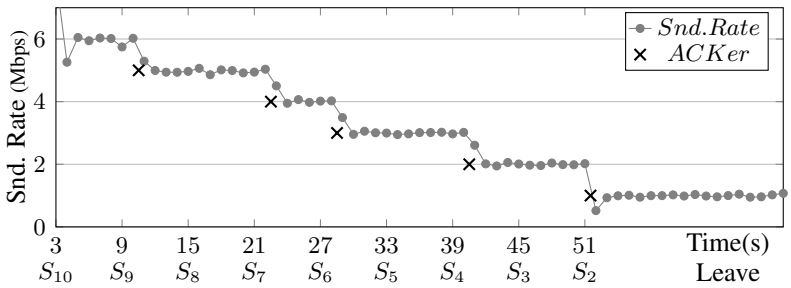
(a) Topology

(b) Receive rate of P_1's receivers

Figure 7.15: Evaluation on ACKer selection: competition.



(a) Rate & ACKer on receivers join



(b) Rate & ACKer on receivers leave

Figure 7.16: Evaluation on ACKer selection: receivers join and leave.

places a corresponding requirement on the selection of the ACKer (without competition S_6 should be the ACKer). The receive statistics are collected once every 2 seconds.

Receivers S_1 to S_{10} receive data from provider P_1, and S_6 receives data from both providers P_1 and P_2. A receiver with $6Mbps$ bandwidth (not shown in the Fig.) is manually assigned as P_2's ACKer. The receive rates observed on S_4, S_5, S_6 and S_{10} are shown in

Fig. 7.15b. P_1 starts first. It can be observed that in the first round, the provider picks S_{10} as the ACKer since his/her reply reaches the provider first. From the second round onwards, the provider picks the correct ACKer based on the settings – the maximum receive rate across the consumers is $6Mbps$. Provider P_2 starts sending at $8s$. P_1 rate drops to $3Mbps$ (fair share) at that point. the receivers' rate. At the next round of ACKer selection (starting at $10s$), P_1 realizing S_6 is no longer eligible. Based on the statistics, (s)he picks S_5 as the new ACKer. When P_2 finishes at $14s$, the receive rate of S_6 grows back to the sending rate ($5Mbps$). Since S_6 sees fewer marks, the ACKer selection done at $18s$ causes P_1 to shift back to S_6.

7.7.3.2 Tolerating Receiver Joins and Leaves

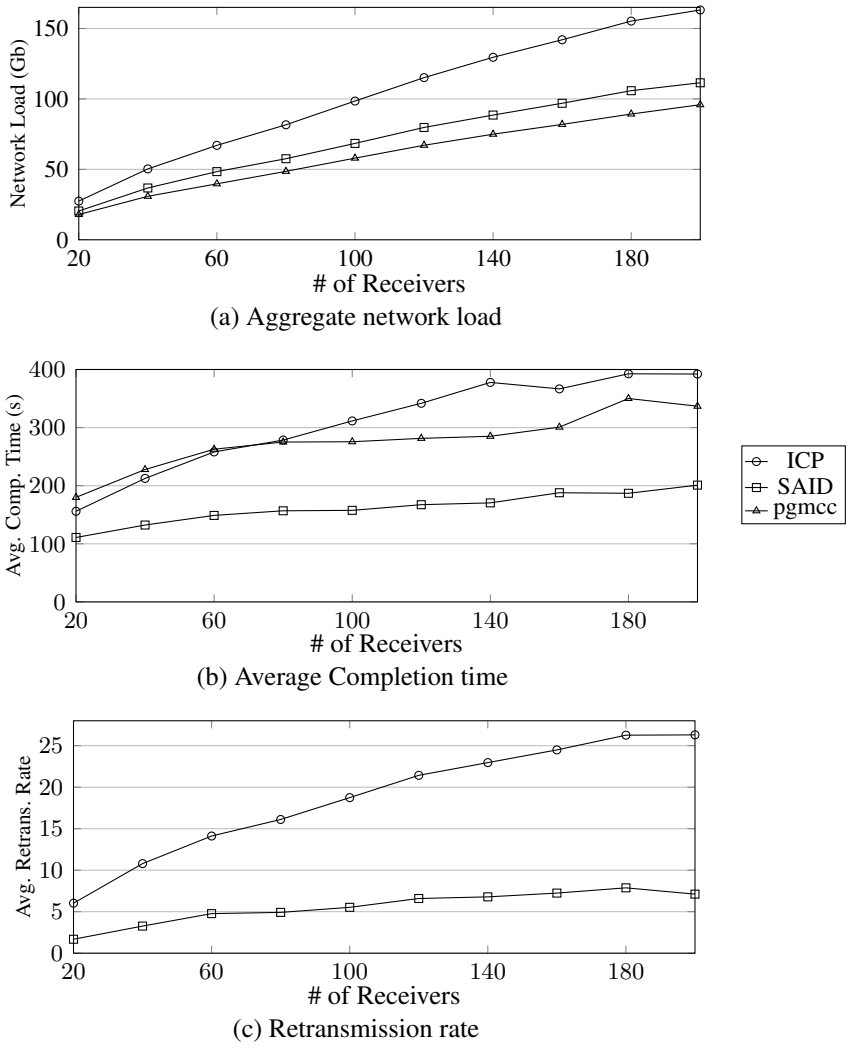
Receivers joining and leaving (i.e., churn) might also affect the selection of a new ACKer. S_1 is the only receiver at the beginning. After $3s$, a faster receiver joins every 6 seconds (S_i joins at $(i-2)*6+3s$). For leave events, S_1 to S_{10} are receiving the flow. From $3s$ onwards, a fast receiver leaves every 6 seconds (S_i leaves at $(10-i)*6+3s$). The sending rate and the selected ACKer are shown in Fig. 7.16. While the receivers change, the data provider can pick a proper ACKer as needed. As long as the original ACKer is within the required bandwidth range, the provider does not pick a new ACKer.

7.7.4 Overall Evaluation of SAID

The totality of SAID is evaluated with the RocketFuel topology (Fig. 4.2). 20-200 receivers are randomly placed on the 79 core routers in the topology. Without the bandwidth information for the topology, available bandwidths are randomly assigned in the range of 1-10$Mbps$ for each link. The simulation result of a trace with 100 flows using ICP, SAID and pgmcc is shown in Fig. 7.17.

By decoupling reliability from congestion control, SAID incurs lower network load compared to ICP (Fig. 7.17a). As the number of receivers increases, SAID has a lower network load compared to ICP, and is less by up to 46% with 200 receivers. Since $\sim 60\%$ of the data is delivered at the first attempt, SAID has a much lower average number of transmissions of each packet from the provider and average flow completion time compared to ICP (Fig. 7.17c).

But SAID consumes more network bandwidth ($\sim 10\%$) compared to pgmcc by aligning to a faster receiver and doing repairs subsequently. For the 10% additional network load, SAID can achieve lower average completion time (by $\sim 65\%$, Fig. 7.17b).



(a) Aggregate network load

(b) Average Completion time

(c) Retransmission rate

Figure 7.17: Overall evaluation result.

Further simulations with different combination of number of flows, number of receivers per flow and size of the flow are carried out. The results (not shown) show the same trend.

7.8 Chapter Summary

Through emulation and an analytical model, this chapter showed that heterogeneous receivers will get out-of-sync with existing receiver-driven in-sequence congestion control

mechanisms. The retransmissions reduce the network efficiency. To overcome this issue, the chapter proposed SAID, a control protocol to enable applications to effectively use a name-based one-to-many information dissemination architecture. SAID allows receivers to request "any-next" packet, instead of the "next in-sequence" and therefore allows more packets to be delivered on the first attempt. For the missing packets, the receivers can get "repair" from other receivers even if the in-network caches have been replaced. Privacy and trust is maintained during the repair phase.

The evaluations show that SAID achieves fairness on each path between the provider and receivers. SAID can reduce aggregate network load (up to $\sim$46%) and transmission time (more than 50%) compared to an existing congestion control mechanism. SAID also reduces $\sim$ 40% transmission time while increasing $\sim$ 10% network load compared to a multicast solution that aligns the sending rate to the slowest receiver. With efficient repair, the streaming application using SAID can have shorter stall time compared to the other mechanisms.

Chapter 8

Extension: *Function-Centric Service Chaining* (FCSC)

Networks are becoming increasingly complex and service providers incorporate additional functionality in the network to protect, manage and improve service performance. *Software-Defined Networking* (SDN) seeks to manage the network with the help of a (logically) centralized control plane. The author observes that current SDN solutions pre-translate policy (what) into forwarding rules at specific switches (where). This choice limits the dynamicity, flexibility and reliability that a software based network could provide. ICN shifts the focus of networks away from being predominantly location oriented communication environments and therefore has the potential to significantly improve the flexibility for network management.

This chapter focuss on one of the problems of network management – service chaining – the steering of flows through the different network functions needed, before it is delivered to the destination. An extension of COPSS – *Function-Centric Service Chaining* (FCSC) – is proposed to exploit ICN in providing flexible network management that utilizes virtualization to dynamically place functions in the network as required.

A trace driven simulation on a real-world topology is used to compare the performance of FCSC and a more "traditional" SDN solution. The results show that FCSC can react to failures with fewer packet drops, adapt to new middleboxes more quickly, and maintain less state in the network.

The key contributions of this work are:

- This work exploits the combination of ICN with SDN to meet the dynamic requirements of service chaining and proposes FCSC, a scalable and flexible architecture, that clearly separates the policy (required functions) from the routing by introducing a light-weight (function) naming layer.

- With the help of varying number of flows and dynamic creation/deletion of virtual service instances on a synthetic and a real-world Rocketfuel topology, this work shows how FCSC compliments the current SDN solution in terms of network state amount, packet drop rate on node failure and overall latency.

Contents

8.1 Introduction

Service provider networks (and networks in general) are becoming increasingly complex. Both network operators and users require various additional functionalities in the network for management and processing of data flows. *Software-Defined Networking* (SDN) aims to manage the network and the functions provided by separating the control plane from the data plane. The SDN controller(s) possess a global view of the network and can therefore simplify the network management as compared to the traditional distributed architectures typical of the Internet. However, even in an SDN environment, management logic ("what") is intricately coupled with the node location ("where"). With the use of virtualization and the prevalence of mobility the location of a particular function in the network may no longer be fixed. The author envisions that the performance of SDN would be further improved by incorporating the ideas of information-centricity that decouple the location of a particular network function instance from the identity of the function it provides. This work makes a first attempt by incorporating Information Centric capabilities into a common and important problem of network management – Service Chaining.

The need to perform additional processing of packets of a data flow in the network before it is delivered to the destination has become an integral element of providing Internet services. These functions include the modification of the packet header (e.g., NAT, proxy), discard packets (e.g., firewall), collection of statistical information (e.g., *Deep Packet Inspection* (DPI)) or even the modification of the payload (e.g., optimization and compression). They are provided in the form of *Middleboxes* [75–77] for policy control, security and performance optimization. The middleboxes have to be resident on the path of a flow, which implies that the traffic has to be deviated from its "natural" IP shortest path and forced through the middleboxes. This work uses the term *Service Chaining* to describe the action of steering packets through these middleboxes. For example, a network operator might require flows that access dynamic web pages such as Facebook, Twitter, FourSquare, Google Instant, or MyYahoo to go through middleboxes like CDN, *Dynamic Site Accelerator* (DSA) [112], TCP optimization over tunnel, etc., in order to improve the perceived user experience [113].

The limited presence of middleboxes at specific locations in the network often results in sub-optimal routing and lower performance (e.g., increased latency, lower throughput, etc.). This is especially true in environments like cellular networks [114, 115] where middlebox functions are restricted to be in the "Network Data Center" and thus have a significant impact on latency. The recent introduction of *Network Function Virtualization* (NFV) [116, 117] promises to make it easier to dynamically and flexibly deploy middleboxes. NFV allows for middlebox functions to be virtualized and therefore be present in greater number and positioned on-demand. The author envisages network service providers

will increasingly adopt NFV to provide network resident functionality, not only for reducing CAPEX but also for offering more flexibility to customers who would like customized processing of their packets. However, managing such a network of dynamically placed functions can be much more complex. Current routing protocols deployed in IP networks constrain how packets can be deviated from well-defined path (e.g., shortest path) and thus cannot take full advantage of the great flexibility offered by NFV.

Recently proposed solutions for Service Chaining in SDN [86–88] attempt to perform Network Management by making use of a (logically) centralized controller that has the capability to setup flow-based forwarding rules on the switches [118, 119] of the desired path. Such solutions provide greater control over the network in order to steer packets of a flow more flexibly, without being constrained by traditional routing such as OSPF [120] and BGP [121]. But the controller has to keep track of the status of the middleboxes and the network.

The author argues that the existing approaches have a common issue of unnecessarily coupling the routing with the policy. I.e., when an SDN controller decides the *functions* a flow needs, it also decides the *path* the flow has to go through and setup *state* on the intermediate switches. These solutions have limitations in *scalability*, *dynamicity* and *flexibility* and therefore have difficulty in adapting to the requirements of a large scale, dynamically changing middlebox set supported by NFV (see §8.2.3 for detailed descriptions).

ICN [4, 5, 12] is a new networking paradigm that introduces ContentNames to decouple the user interests from data location. Following this line of thinking, this work presents *Function-Centric Service Chaining* (FCSC), a novel approach that decouples the *functions* a flow needs from the *location* of network function instances (and thus routing) via a naming layer (see Fig. 8.1). Such a decoupling facilitates the dynamic modification of the functions needed by a flow on the controller or the middleboxes (e.g., DPI, load balancer). This also enables switches to dynamically detect the load (popularity) of a certain function and accordingly instantiate/dispose of network function instances (co-resident with the switch or on some other node). The enroute function-based routing allows more dynamic use of the newly created instances and faster recovery from node/link failures. FCSC intrinsically supports the presence of multiple instances for the same functionality and can perform network-layer load-balancing among these nodes at any time. By placing the flow state in the packet header, FCSC helps to reduce the amount of state stored in the network and results in much better scalability compared to the per-flow state solutions like SDN. FCSC is therefore able to provide a highly dynamic and adaptive Service Chaining capability and effectively exploit the promise of NFV in the software-based network of the future.

8.2 Scenario Description and Problem Statement

This section describes the scenarios we envision of how network resident functionality of middleboxes could be utilized and point out the shortcomings of the state-of-art SDN solutions. These will be used as the basis to demonstrate the benefits of FCSC.

8.2.1 Service Chaining Scenario

An *Autonomous System* (AS), for example an IP network, data center or an information centric network, is typically composed of many edge routers and a set of core routers/switches. Packets from users enter this AS from one of the edge routers (Ingress). These packets categorized into flows (either by 5-tuple in IP or "Interest" prefix in ICN) need to go through a specified set of functions in the core in *a particular order*, as required by policy. The functions may include DPI, policy, *Quality of Service* (QoS), NAT, DSA, proxying, transparent caching, accounting, logging, etc. It is also possible that a subset of these functions may in fact be provided by third parties, and possibly in a cloud-resident platform [122].

8.2.2 Detailed Requirements

With the growth of the middleboxes and the network traffic, the author envisions that an efficient service chaining network should meet the following requirements:

- **Flexibility:** The outcome of packet processing by a middlebox may change the set of function(s) to be applied on subsequent packets of the flow. E.g., after a packet goes through DPI, the policy or algorithm may determine the need for additional network resident functions like intrusion detection, logging, etc., to be applied on the flow. It is also possible that functions can reduce/replace the functions a flow needs to go through. E.g., after observing a set of packets in a flow, the DPI can decide to remove the virus scan and even DPI itself from the function list. Therefore, even if a set of apriori service functions were specified, they might be changed during the lifetime of the flow. An efficient service chaining network should support such changes in a flexible way – the middleboxes should be able to determine the functions of a flow *themselves* and the changes should take effect *immediately*.
- **Dynamicity:** The advent of NFV allows for network resident middleboxes to dynamically incorporate (additional) functionality by spinning up additional virtual machines on demand. E.g., if there are many more flows that require firewall functionality but fewer flows require DPI functionality, the network manager should be able to instantiate more firewall nodes and reduce the number of DPI nodes. Since

more functions are running on virtualized platforms, these functions can potentially be placed anywhere in the network instead of on only a selected set of predefined nodes. This requires the network to be able to apply these changes as soon as possible while keeping the communication cost low. For the functions having multiple instances, the network should also be able to balance the load on these instances to optimize performance.

- **Scalability:** The scalability requirement comes in three dimensions: the number of functions, the number of flows and the size of the network. With more customized services provided to network users, it is envisioned that there would be an increase in the number of network functions available. For the networks that adopt NFV, the number of instances of network functions can also grow to be large. A scalable service chaining solution should not limit the number of users/flows, the number of functions a flow should traverse, or the size of the network due to the response latency or the number of states stored in the network.
- **Reliability:** A productive service chaining solution should also take reliability into consideration. The solution should be able to dynamically react to the node (middleboxes, switches or controllers) failures and the link failures within a threshold. As suggested by [123], the recovery time of a failure should be within 10s' of milliseconds.

8.2.3 Limitations of Existing SDN Solutions

Current SDN solutions [86–88] perform better than PBR and indirection based solutions. However, they are still not able to meet the requirements mentioned above, because:

Flexibility: When a middlebox like DPI needs to change the functions a flow requires, it has to rely on the controller to build a new path that goes through a certain instance of each of these functions. This results in extra control overhead in both communication and latency for every flow whenever the set of functions are changed. This is not desirable since the controller in SDN design is supposed to *generate* the rules but not be involved in the real-time handling of packets [124].

Dynamicity: It is difficult for SDN controllers to perform real-time decisions on the path of a flow to balance the load in the network and on the function instances. The problem will become more severe when the number of flows grows and NFV enables more dynamic instantiation/disposition of function instances.

Scalability: SDN solutions place rules for every flow on the switches. The number of rules stored in the network is proportional to the number of flows, the functions the flows require and the size of the network. It is very difficult to scale when the network has larger

number of flows or the network itself grows larger.

Reliability: When a middlebox or a link fails, the switches in the existing SDN solutions have to rely on the central controller to build a new path for the flow. This increases the convergence time while dealing with such failures and might violate the typical 30-50ms convergence time target requirement typical in a large provider networks. Alternatively, the controller has to setup backup paths proportional to the number of hops for every flow to ensure quick convergence time. But this exacerbates the scalability problem.

8.3 FCSC Overview

This section starts by reasoning the design choices of FCSC and then describes the whole architecture based on the design choices. Design details will be provided in §8.4.

8.3.1 Design Rationale

To achieve the requirements of flexibility, dynamicity and reliability as described above, this work proposes to add a naming layer (similar to ICN) to the current SDN architecture. Moreover, to improve the scalability of the system, FCSC chooses to put flow-state in the packet header rather than in the switches. But FCSC is still backwards compatible with existing SDN-based service chaining solutions.

8.3.1.1 Naming Layer

ICN provides flexibility to users – they only need to request the network with *what* they want rather than *where* that data might reside. Such a shift in the focus of the network also provides better dynamicity and reliability – a request can go to *any* of a set of possible data providers/caches in the network.

There is a strong similarity between the fundamental needs that drive service chaining and the capabilities offered by ICN. Middleboxes that need to change the function list of flow (e.g., DPI) require flexibility – they only need to care about the *functions* the flow requires rather than asking the SDN controller to build a path to *where* the flow should go through. While forwarding a packet, the network can forward the packet to *any* of the instances that provides the same function. This allows the dynamic adoption of new function instances and can also help fast recovery when a function instance/link fails.

To achieve high performance (line-speed forwarding in the network), this work primarily incorporates the hierarchical naming capabilities of an ICN environment like NDN, to represent the function list and the longest-prefix matching in the FIB to forward packets. The reverse-path forwarding (PIT) and caching (Content Store) capabilities that are used in an ICN for information distribution is not key to this solution. According to [91, 125], line-speed (hashed) name forwarding is achievable with existing hardware.

8.3.1.2 Flow State in the Packet Header

The solutions of existing SDN-based service chaining incorporate flow state in the switches – the switches maintain state on how to forward a specific packet based on the 5-tuple (or other header features) of that packet. Such a design results in the number of rules in the network to be proportional to the number of flows and the number of functions these flows require. It does not scale well with the growth of clients (flows) and the adoption of new functions in the network.

Therefore, FCSC chooses to put the flow state – the functions the packet still need to traverse through – in the packet header. The function list of a packet (in the form of an ICN name) is tagged to the packet header when it enters the network. After traversing through a middlebox, the name of the applied function is removed from the header and the network will forward the packet to the next function it requires.

In FCSC, the network only needs to maintain forwarding information on a per-function basis rather than a per-flow basis. The amount of state stored is therefore proportional to the functions in the network but not the flows, and thus our solution can scale much better than existing solutions.

8.3.1.3 Compatibility with Existing SDN Solutions

Network management, as exemplified by service chaining, needs SDN for flexible placement of functions and more powerful routing, and achieves this because it has a (logically) centralized view of the whole network. The purpose of our solution is not to replace these ideas in SDN solutions or to remove the existence of the (logically) central controllers, but to make them more flexible by the modifications proposed in our paper: namely the naming layer and the use of flow state in the packet header. FCSC can also be backwards compatible with the existing SDN solutions by naming all the intermediate switches (in the form of IP or MAC addresses) and setup a separate forwarding table on every switch in a hop-by hop manner. But that will result in the loss of the benefits mentioned above.

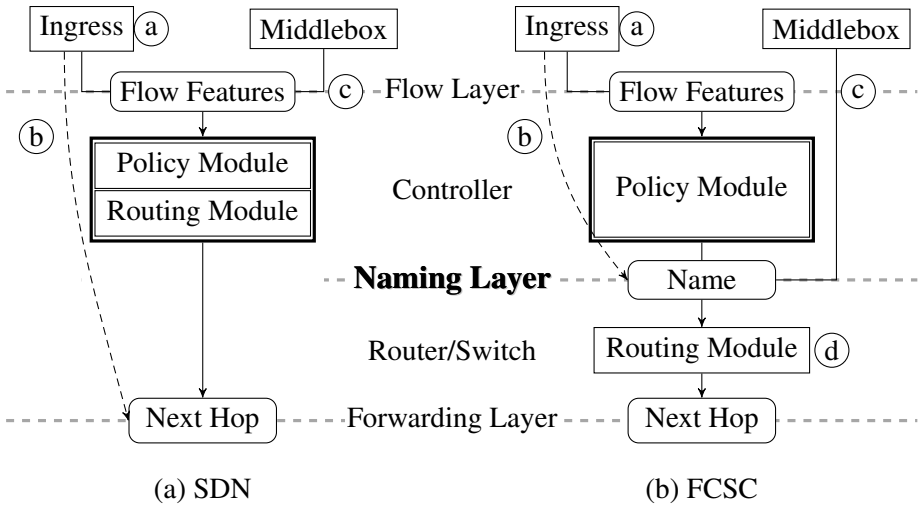


Figure 8.1: Architectural design of FCSC vs. SDN.

8.3.2 Architecture Description

Fig. 8.1 illustrates the logical separation of the architecture of FCSC compared to existing SDN solutions. As described above, a *Naming Layer* is added in the architecture that separates the *policy* module (the module that manages *what* functions should be applied to a flow) from the *routing* module (the module that manages *where* the function instances reside). The representation of the naming layer (the function list a packet should go through) resides in the packet header to scale the network better. To help understand the figure, this section describes the differences between the two solutions in the following 4 scenarios (following the marking on the figure):

(a). Flow initiation:

In SDN, on seeing a new flow, the ingress sends the flow feature (e.g., 5-tuple) to the controller. In this case, the controller has the *function* module and *routing* module coupled. The controller determines the result, a set of forwarding rules (e.g., 5-tuple$\mapsto$NextHop) that are then incorporated on different switches on the path.

In FCSC, the controller determines which functions the flow needs and return the result only to the ingress. The ingress then tags the packets of the flow with the function list. On seeing the functions carried in the packet header, switches in the network will look into their FIB (in the form of Function$\mapsto$NextHop) and decide the outgoing "face" of the packet. The FIB of the switches are controlled by the *Routing Module*. The *Routing Module* can either

be distributed (e.g., OSPFN [126], IS-IS [127]) or logically centralized (e.g., SDN).

ⓑ. Proactive rules:

The controller can also setup wildcard proactive rules on every ingress. An SDN controller essentially has to build a path for every flow from each ingress. This increases complexity since almost every edge router can be an ingress and there might exist $O(N^2)$ $src-dst$ paths where N is the number of edge routers even without considering different paths for a same $src-dst$ pair.

In FCSC, the controller only needs to flood the wild card rules (FlowFeature↦FunctionList) to all the ingress. The core of the network does not have to keep any state on a per flow basis any more.

ⓒ. Policy change by middleboxes:

When certain middleboxes need to change the policy (function list) of a flow, in SDN the middleboxes have to request the controller to build a new path for the flow. This might result in even more state in the network and also higher latency, just like the experience at the beginning of a flow.

FCSC allows middleboxes to determine the new policy, without having to request the controller to change forwarding rules at specific switches. These middleboxes change the function list in the packet header and the network will forward it towards the next middlebox automatically. Additionally, they may notify the ingress to change the function list for the future packets of the flow. This solution therefore only requires a change in the state of the packet header and the ingress as opposed to every switch on the old and new path. Therefore, unlike existing solutions, it does not require additional set up time while a middlebox like DPI tries to modify the policy. It is thus able to quickly enforce the policy on the newly arriving packets.

ⓓ. Dynamic routing:

With existing SDN solutions, the functions a flow requires is represented by the path it follows. Whenever a middlebox fails, the failure notification has to be reported to the controller before the new path that includes another instance of the function is built for the flow. Approach in [123] pre-computes backup paths to shorten the recovery time on such failures, but this results in exponential complexity due to the permutations and function combinations. When a new instance of a function is adopted, it is also difficult to use it for existing flows for purposes of load balancing.

FCSC separates the routing of flows from the policy. The switches can decide (an alternate) outgoing face based on its own FIB. This shortens the response time for node/link

failures and can make use of new instances of functions on the fly (as long as the FIB is updated based on the *Routing module*).

8.4 FCSC Design Details

This section describes in detail how we design the architecture to ensure a highly efficient and scalable service delivery network.

8.4.1 Naming Strategy

FCSC extends the ICN principle of *naming content* to *naming function*. Every instance that provides the same function is referred to by the same name, e.g., `/DPI`, `/Firewall`, etc.

When the network policy requires a flow to go through a sequence of functions, the policy executor (the controller or the ingress) will encapsulate each packet of the flow with a header containing a name that represents the sequence of functions to be executed, in a FCFS manner. E.g., a packet header with name *chain:*`/DPI/Cache/R5` implies that DPI and cache function must be applied to that packet before it exits the network from the egress R_5. Here, FCSC uses the scheme identifier (as per URI Generic Syntax [9]) "*chain*" to represent the packets for service chaining. Other identifies like "*monitor*", "*ctrl*", etc., can be used to represent packets meant for other purposes (e.g.monitoring and controlling, etc.)

The switch fronting a middlebox (SxFM) will pop the first part of name (prefix) in the packet header before it forwards the packet to a middlebox function associated with it. Some policy nodes can also change the name to redirect the packet towards other functions.

Prefix popping is a simple and stateless task. It can be separated from the switching and the middlebox functions. If necessary, FCSC can include a designated hardware component for acceleration, or instantiate a virtual prefix popping function, on the SxFM (although we believe it is a simple task). Since it is a stateless task, the SxFM can also have multiple of these components (either hardware-based or software-based) that run in parallel to ensure line-speed forwarding is achieved on the SxFM.

Fig. 8.2 illustrates the lifetime of a packet a FCSC network. The ingress encapsulates the incoming packet with a header *chain:*`/DPI/Cache/R5` as desired by policy. The network forwards the packet to the SxFM of the "best" DPI function (in terms of relative location, latency or other criteria). The prefix `/DPI` is removed when passing through the prefix popping function (represented by a green box) before entering the DPI box. The DPI function decides the packet should also go through a firewall and since there is a load balancer for

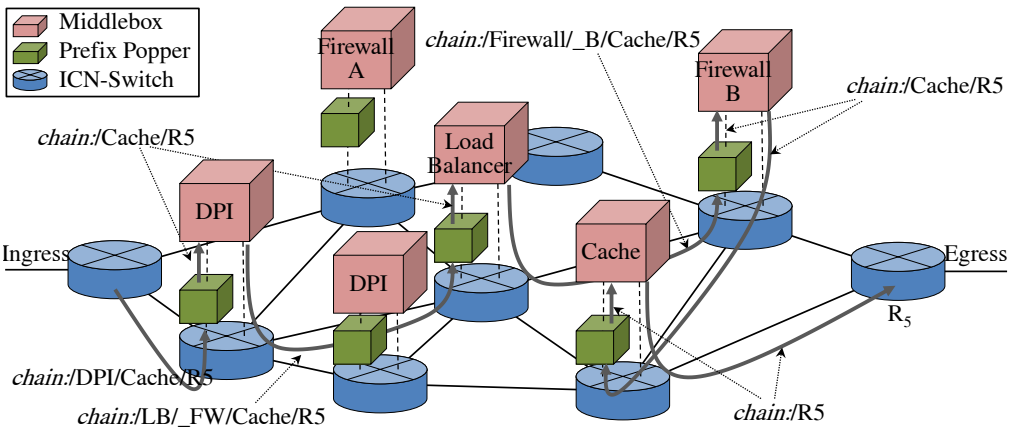


Figure 8.2: Example of the name changing of a packet in FCSC.

the two firewalls in the network, the DPI adds a prefix `/LB/_FW` to the header. The prefix is replaced with `/Firewall/_B` by the load balancer since it decides that the flow should go through Firewall B. The remaining prefixes are popped one by one when going through the Firewall B and Cache. On reaching R_5, the egress sees its own name and therefore it decapsulates the packet and forwards the original packet out of the network.

8.4.2 Routing Strategy

Middleboxes need to advertise their existence before they can be used by the flows. A middlebox offering a certain service (e.g., Firewall) advertises the name of the service as prefix (e.g., `/Firewall`). Packets whose names have the prefix `/Firewall` can be routed towards this middlebox as a consequence of the normal name based routing. A packet in FCSC is only forwarded to one middlebox even when multiple middleboxes exist for the same function (prefix). The intermediate switches can monitor the popularity of a function based on these prefixes, and they can create additional instances where needed with NFV support.

The decision of *which* exact instance of the function a packet should traverse is determined by the routing module. FCSC does not limit the routing module that can be adopted. To better support different routing strategies, FCSC provides a simple standard interface for the routing module to control the forwarding decision. A "cost" field is added in the FIB and thus the data structure looks like `Function`$\mapsto\{$`NextHop, Cost`$\}$. If multiple "NextHop"s exist for the same function, the switch will always forward the packet to the next hop with the lowest cost. The routing module can have different interpretations of the "cost", e.g., link latency, policy, energy/work-load considerations, etc.

The choice of the routing scheme can affect the dynamicity and reliability of the whole solution. The possible routing solutions can be generally categorized into two categories: centralized and distributed schemes. But, it should be noted that regardless of which routing scheme is used, FCSC is able to achieve better scalability since it only maintains function state.

Centralized routings solutions (e.g., SDN [87]) have better control over the node state including what is maintained at switches and middleboxes. They provide more flexible control in determining where middleboxes should be placed and monitoring node state. Routing based on names of function instances may offer better real-time load balancing capability, faster failure recovery and utilization of new function instances.

Distributed routing solutions (e.g., [126]) allow every switch to have the intelligence to make routing decisions on its own. This would enable dynamicity in routing to a newly created instance or avoiding a failed link/middlebox. But these solutions might incur higher control overhead for synchronizing the network state on every switch, especially when automatic load balancing is required for different instances of the same function.

Both the centralized and the distributed routing methods face the difficulty of achieving real-time load balancing similar to [128]. A compromise is for the network to incorporate a load balancer to dynamically distribute the load on the servers/middleboxes that have the same functionality. FCSC can fully support such load balancers (see the example in §8.4.1).

8.4.3 Stateful Middleboxes

There might be some functions in the network that need to maintain states. In such a case, all the packets of a flow should go through the same instance, even though they may not care which actual instance they might use. This implies that the different instances for the same function cannot be treated equivalently. The two firewalls in Fig. 8.2 could be an example of this kind. FCSC adopts the hierarchical name in ICN to meet this requirement. Instead of using the same name, the multiple instances share a common prefix (function name), but they have function-level unique ID. E.g., the firewalls in Fig. 8.2 are called `/Firewall/_A` and `/Firewall/_B` respectively.

While advertising the prefix, the middleboxes advertise the whole name instead of the function name itself. If a packet can go through any of the instances for a function, it just puts the function name in the header (e.g., *chain:*`/Firewall/Cache`). Otherwise, it will use the full name (e.g., *chain:*`/Firewall/_A/Cache`). This can be determined by the policy executor or on the fly. ICN switches perform the longest-prefix match, and therefore the packet can be forwarded to the required function instance, if specified. While popping the

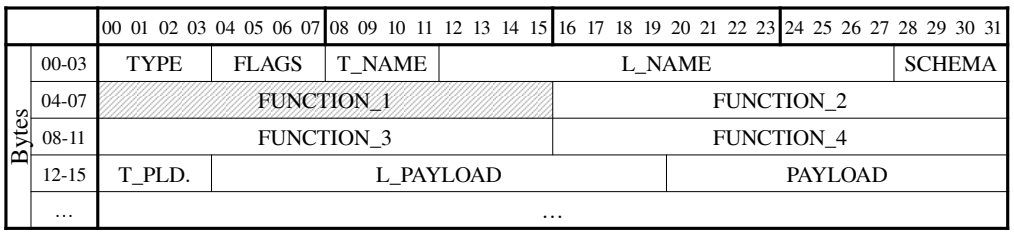


Figure 8.3: Example of a packet header in FCSC.

name from the packet header, the prefix poppers can also perform "longest-prefix popping" of the full instance name from the name list. To avoid ambiguity, this solution requires that the instance ID space should not overlap the function name space. E.g., Firewall B pops both the `/Firewall` and `/_B` prefixes from the packet header in Fig. 8.2.

For the functions that require visibility of the bidirectional packets of a flow, the policy module can also specify the function instance via its full name and create a function (instance) list in the reverse order. E.g., if say the firewall function (only) requires packets from both directions, the policy layer can create name *chain:*`/DPI/Firewall/_A/Cache` for one direction and *chain:*`/Cache/Firewall/_A/DPI` for the other.

8.4.4 Packet Header Optimization

The format of a packet header is critical for achieving line-speed forwarding and processing. Since FCSC adopts a variable-length packet header, the solution should try to avoid complex lookup method in the intermediate switches. With a carefully designed header format, FCSC switches only need to do lookup based on a certain set of bytes in the header. The variable length will not affect the lookup.

To achieve efficient lookup, FCSC (or the network administrator) first hashes each function name into a integer with certain-length. A 16-bit integer is likely to be sufficient to represent all the functions needed. FCSC keeps the TLV format adopted by NDN for extensibility, but it requires that the first TLV set be the ContentName. The placement and the length of the first function ID is then determined. Therefore switches in FCSC has a constant lookup time and complexity, thus achieving forwarding efficiency similar to MAC address based forwarding.

Fig. 8.3 gives an example of a possible packet header in FCSC. Here, `T_NAME` and `T_PLD` are the 4-bit type constants for name and payload. `L_NAME` and `L_PAYLOAD` are the length for name field and payload field. They are also represented by a 16-bit integer here. There are 4 functions listed in the packet header. The intermediate switches only need to check

the function ID in bytes `04-05` while forwarding (highlighted in the figure).

8.4.5 Security

Security is another concern in service chaining. The users of the network should not have any chance to infiltrate the network and steer the packets through paths that are not allowed by the policy.

FCSC encapsulates each packet at the point they enter the network and decapsulates them on egress. Therefore, the service provider network is essentially transparent to users. FCSC does not provide any user interfaces to the clients and therefore there should be no way a user can interact with the encapsulation/decapsulation function or the other function modules in the network. No client of FCSC can violate this policy by altering the packet in some way.

8.5 Evaluation

This section evaluates the dynamicity, reliability and scalability of FCSC with a distributed routing module via simulation and compares it to a relatively simple (physically) centralized SDN solution that is conceptually similar to [86, 87].

These approaches use the basic OpenFlow protocol constructs that is a controller interacting with network forwarding elements [119].

This work recognizes that there are approaches for decentralized SDN solutions like [124, 129], but the results from [130] show that the inconsistent SDN control state can *significantly* degrade performance of logically centralized control applications that are agnostic of the underlying state distribution. In addition, the communication overhead for keeping all the controllers synchronized has to be addressed. Moreover, even if there exists multiple controllers, it is still fair to assume that each of these controllers is in charge of a set of routers (a portion of the network). Therefore, the centralized solution used here can also be viewed as such a portion.

The benefits of FCSC is first demonstrated on a small synthetic topology (shown in Fig. 8.4). Subsequently, a large-scale simulation on a real world topology is performed to evaluate the scalability and the efficiency of our solution in a more realistic environment.

8.5.1 Study of FCSC Behavior

Fig. 8.4 shows a simple topology with multiple middleboxes. R_1-R_6 are FCSC switches. N_1-N_4 and *DPI* provide functions *A*, *B* and DPI as noted in the figure. *Src* and *Dst* are the source

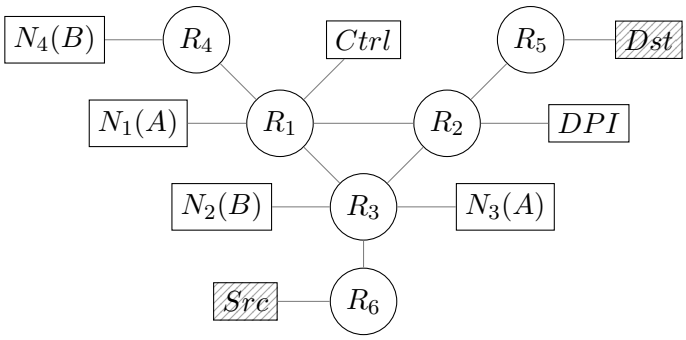


Figure 8.4: Demo topology to evaluate FCSC.

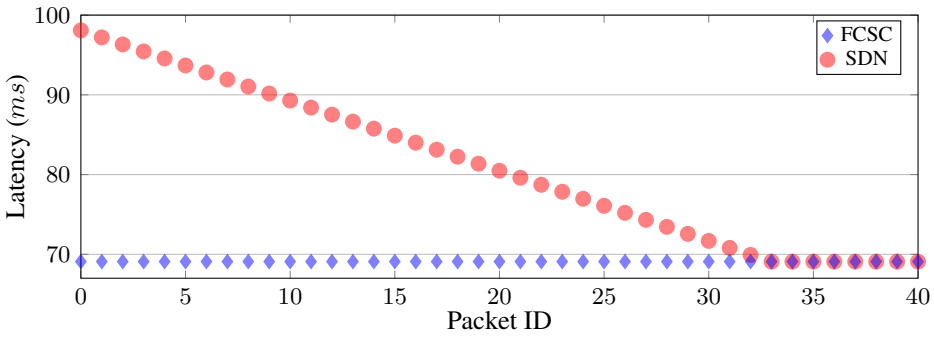


Figure 8.5: Flow initiation using proactive rules.

and destination of a flow of interest. *Ctrl* is the central controller in an SDN solution. The link latency between switches is $2ms$ and the latency between switches and the end-systems (middlebox, src, dst, control) is $10ms$. The bandwidth on the link is $100Mbps$ (large enough to support the flow). The processing latency on all the middleboxes (including *Ctrl*) is $1ms$, or $1000pps$ (packets per second). The sending rate at *src* is also $1000pps$. Several scenarios are used to compare the behavior of FCSC with the simple SDN solution:

8.5.1.1 Proactive Rules for Flow Initiation

The evaluation first compares the initiation phase of FCSC with an SDN solution without proactive rules set up in the switches. Fig. 8.5 shows the overall latency (the amount of time spent from *Src* to *Dst* in the network) of every packet in the initiation phase of a flow that requires *DPI* and *B* function. Since FCSC does not make a request to the central controller, it can achieve significantly lower latency for the first 30 packets of a flow compared to the SDN solution. This reduction may be critical for small flows that require timely processing

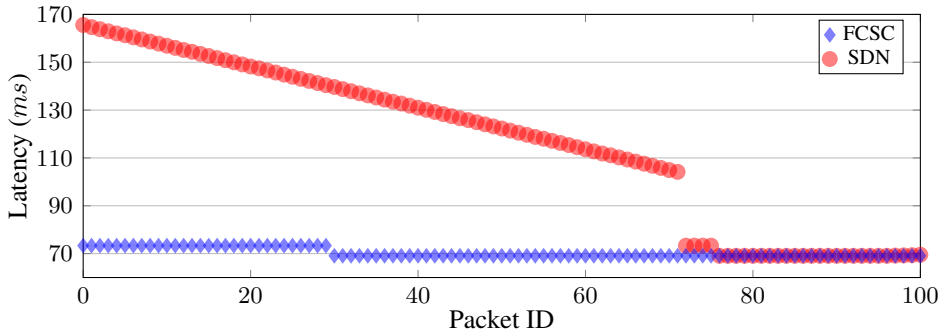


Figure 8.6: Dynamic function modification by DPI.

of middlebox functions (e.g., games).

With proactive rules set up in the switches, SDN solutions can achieve lower latencies, as good as FCSC. However, it still requires a lot more effort to pre-calculate the paths for different permutations as compared to FCSC, which makes general proactive rule setup quite infeasible in SDN.

8.5.1.2 Dynamic Policy Change on Middleboxes

In the second experiment, the default policy requires a flow to go through *DPI* and *B* functions (represented as a *chain:*`/DPI/B`). But the DPI then decides to add function *A* and also removes itself from the function list (the name then becomes *chain:*`/A/B`) after it examines the first packet of the flow (this is a typical dynamic function processing required in service provider networks for actions such as mobile video processing etc.). In the SDN solution, *DPI* requests *Ctrl* to create a new path for the flow and block the packets from being forwarded before the new path is built. In contrast, with FCSC, *DPI* directly renames the packets that continue to arrive and notifies the ingress (R_6) to change the policy. There is no need to block the packets at the DPI.

Fig. 8.6 shows the latency of the first 90 packets in the flow. In FCSC, only the first 29 packets go through *DPI* with less than 75*ms* overall latency . However, in SDN, 73 packets flow into *DPI* before *Ctrl* can setup a new rule for the later packets. The overall latency of the first packet grows up to 165*ms*. Another 4 packets experienced a loop since the rules are not setup atomically.

The result shows that FCSC responds faster to the dynamic policy changes, this can results in lower packet latency and also lower DPI load (process & modify 29 headers vs.process & buffer 73 packets in this example).

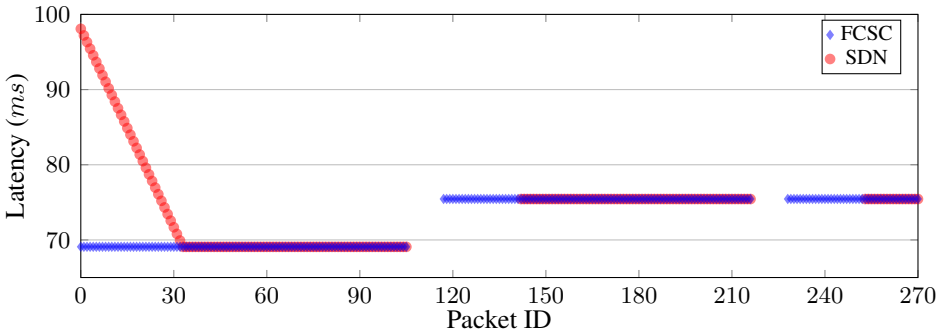


Figure 8.7: Dynamic failure recovery.

8.5.1.3 Dynamic Failure Recovery

In the third trace, the flow is required to go through functions A and B (represented as *chain:*`/A/B`). The initial shortest path routing in both SDN and FCSC choose to go through N_3 for A and N_2 for B. N_3 is disposed at 150s and N_2 is disposed at 240s. The packet loss and recovery time are compared in FCSC and SDN.

Fig. 8.7 shows the overall latency of the packets that reach *Dst*. The packets in the outgoing buffer of SxFM and on the link to a failed middlebox ($\sim$ 10 pkts) have to be dropped in both solutions. Since the intermediate switches can redirect the packets without going to the central controller, FCSC can have around 25 more successful deliveries every time a node fails. This value can increase when the network is becoming more complex or the controller is farther away from the failure.

8.5.1.4 Dynamic adaption to new instances

The last trace has a flow that goes through functions A and B. At the beginning of the trace, only N_1 and N_4 are instantiated. N_3 is created at 150s and N_2 is created at 240s.

Fig. 8.8 shows the overall latency of the packets in the flow. The SDN solution does not modify the path of the ongoing flows due to the complexity (the problem is similar to a warehouse location problem and is NP-complete). Therefore, the latency does not change even when the new instances are created for the functions. FCSC enables the middleboxes to advertise their function prefix to the network and the switches can redirect flows based on that information and therefore this solution can adapt to the new instances and the latency is lowered. Note that when N_3 is instantiated at 150s, the flow is redirected to N_3 for the shorter distance from the ingress, but the overall latency is not changed because the same

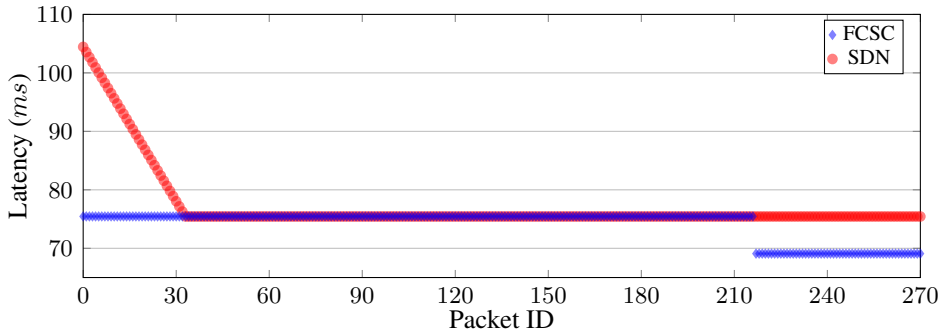


Figure 8.8: Dynamic adaptation to new instances.

number of hops are traversed. However, adding N_2 reduces the latency in FCSC as the packets do not have to flow to N_4.

8.5.2 Large Scale Evaluation

A slightly modified Rocketfuel topology [100] (Exodus, AS-3967, see Fig. 4.2) is used to evaluate FCSC in a real world environment. The 18 cities present in this topology is used as the core network. The latency between every pair of these core switches is determined by the average of the latency on the links between the two cities. The processing results in 30 links with latency ranging from $2ms$ to $21ms$ and a mean value of $6.6ms$. The latency between the core switches and the end-hosts, middleboxes and the controller is set to $6ms$. Since the original topology only contains latency information, $100Mbps$ bandwidth is assigned to all the links.

The simulation assumes that there exist 11 different functions in the network. One of them is unique in that the DPI-like function rewrites the function list as needed. The flows belong to one of the 100 different applications. Every application requires a range of different network functions varying in number from 1 to 4. DPI can dynamically change the functions a flow that needs.

8.5.2.1 Varying Number of Function Instances

The evaluation first studies the benefits of adopting FCSC in a network that dynamically creates virtual instances at random switches. 100 long-lasting ($5min$) flows starting at $0s$ with different sending rate (ranging from $120kbps$ to $1.05Mbps$) are used in this simulation.

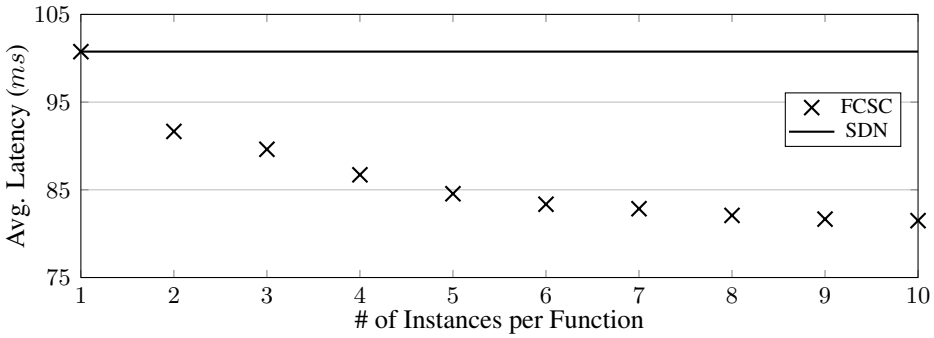


Figure 8.9: Benefit of increasing # of instances per function.

At the beginning of the simulation, there are only 1 instance for each function initialized. Then, a new randomly located instance for each function is instantiated every half a minute until the maximum number of instances is reached (the maximum number of instances for each run is shown in the x-axis in Fig. 8.9). In the first run, only the initial middleboxes are used for the entire 5 minutes; in the second run, in addition to the initial functions, another one instance per available function is randomly placed on a switch in the network at time 30*s* and lasts until the end of the simulation. In the third run, a third instance is put into the network at time 60*s* and so on.

Fig. 8.9 shows the average latency vs.the number of instances eventually initiated for that simulation run. FCSC can automatically adapt by making use of new instances that are closer, even for the ongoing flows and the average latency drops from 100.75*ms* to 91.66*ms* when adding a second instance. The latency is further reduced to around 85*ms* when we have 5 instances created. This is beneficial for long flows compared to the alternative SDN solution where the ongoing flows are unaffected by dynamic addition of functions in the network, unless the controller resets the rules.

The results illustrate that FCSC is able to seamlessly take advantage of new instances of virtual middleboxes that have the same functionality, even when the network is not overloaded. Another observation is that the higher the number of instances, the lower the incremental benefit. In the simulation environment which consists 18 switches, more than 8 instances do not yield additional benefit. Ignoring the absolute numbers, since it is topology dependent, one can nevertheless envision that there is a tradeoff between user-experience and the cost of the deployment. Another way to reduce the latency is to pick an optimal location to instantiate the middlebox. This is an additional optimization that can complement the architectural proposal.

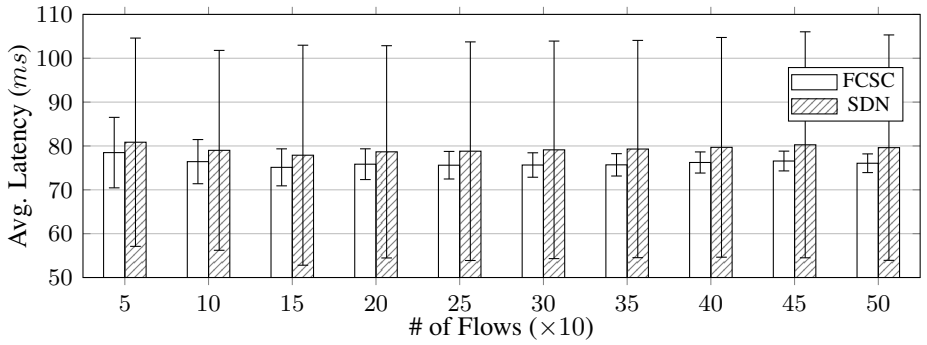


Figure 8.10: Latency (and 95% CI) vs. # of flows.

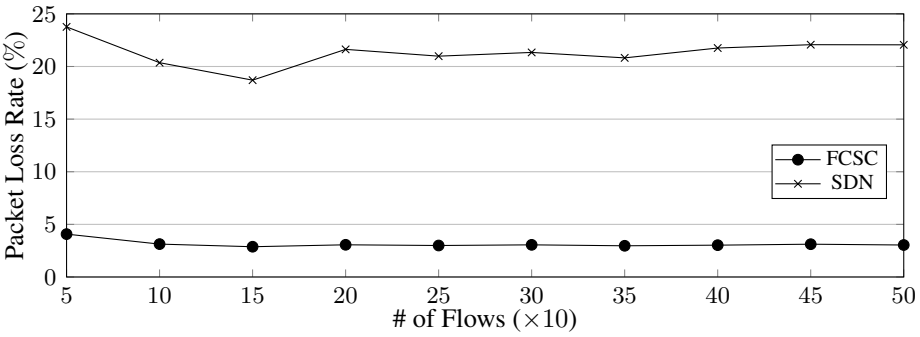


Figure 8.11: % of packets lost vs. # of flows.

8.5.2.2 Varying Number of Flows and Function Instances

The simulation is now loaded with a varying number of flows (50, 100, ..., 500). Each flow has its own arrival time (within the first 5*min*), a sending rate in the range (1.2*Mbps* - 11.9*Mbps*) and duration (0.05*s* to 91.24*s*). The simulation also randomly generated 1,151 middlebox creation/failure events during the simulation period. FCSC and SDN solutions are compared with metrics average latency for the flows, packet loss caused by middlebox failure, and the number of rules stored on the switches.

Fig. 8.10 shows the average latency along with 95% confidence interval (CI) for the different flows. FCSC provides lower average latency and less variability compared to the SDN solution, since the flows are able to take advantage of new instances that are closer.

The overall percent of packet lost in Fig. 8.11 shows that with the dynamic failure recovery, FCSC helps to deliver more packets to the destination. Lower loss rate usually means lower re-transmission rate and also lower overall network cost.

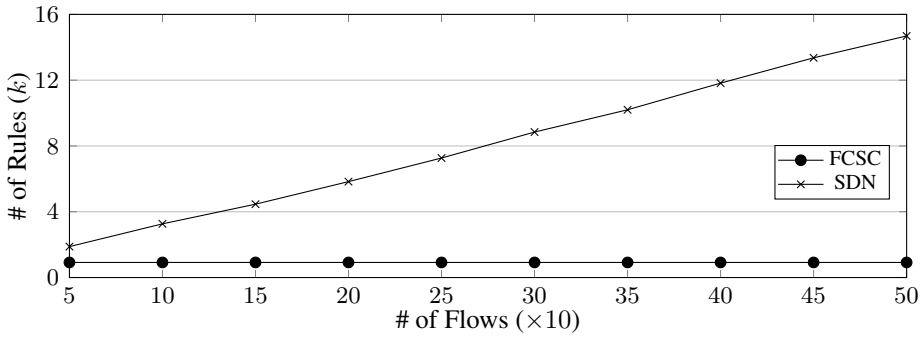


Figure 8.12: # of rules vs. # of flows.

FCSC switches only maintain rules on a per available instance of a function, unlike the SDN solution that keeps rules for each flow type (defined by an n-tuple, potentially with wild-cards). Therefore, the number of rules do not change with the number of flows (as shown in Fig. 8.12). However the number of rules in SDN grows with the number of flows. It shows that FCSC be more scalable especially in a large network with millions of concurrent flows.

8.6 Chapter Summary

Existing SDN-based network management solutions pre-translate the policy (what) into the forwarding rules at specific switches (where). Such a design choice limits the benefits that a truly software-based network could provide. This chapter proposes FCSC which explores the potential of using information-centric concepts within a network management environment, especially focusing on service chaining.

Using both synthetic and realistic topologies, the work shows that FCSC is able to provide policy makers simpler interfaces to control a flow (flexibility), is able to react to middlebox failures with fewer packet drops (reliability), is able to more quickly adapt to new instances of middlebox functionality (dynamicity), and requires less state to be maintained in the network (scalability).

Chapter 9

Extension: Object Resolution

ICN enables accessing data oblivious of its location, by allowing end-systems to retrieve content based on names. But, existing architectures do not yet provide a mechanism for end-system applications or users to obtain these names. There is a need for an object resolution system that addresses a most important and as yet unimplemented component of obtaining a name in ICN. This chapter proposes ORICE, an architectural design for supporting Object Resolution services in Information-Centric Environment that satisfies this need. To prove the feasibility of the design, a prototype is built on top of NDN and COPSS to help the users get names/CDs from keywords.

The key contributions of this work include:

- An object resolution architecture that provides intelligent search functions at the application layer and leverages an ICN for efficient data dissemination.
- A separation between the name certification system and search engines in ICN that can satisfy more diversified search requirements and content reliability.
- A prototype of the architecture to showcase the efficiency and necessity for an object resolution system in ICN.

Contents

9.1 Introduction

ICNs typically assume that the data consumers know the content identity before they send requests or subscriptions. To make the content identifiers meaningful, NDN and COPSS adopt human-readable, hierarchically structured Names and CDs. However, these solutions do not mandate how or from where a user finds the "ICN-name" for the information they seek. The lack of such a functionality can make communication ineffective, especially in a pub/sub environment. E.g., if a subscriber subscribes to CD `/sports/football/Germany` but the publisher publishes to `/Germany/football`, the subscriber will miss these messages.

To find content of interest in the current-day Internet, users express the feature of the desired content via keywords and other characteristics like images, videos, etc. Object resolution is used to translate these features to the URL for the object, which can be used to find the location of a node that provides the data. Similar object resolution systems are needed in ICN to bridge the gap between the users' inputs (e.g., keywords) and the network's requirement for ICN-names/CDs.

Search engines (e.g., Google, Yahoo, etc.) in conjunction with DNS provide the object resolution function in IP networks. The search engines translate the keywords into URIs which are then mapped to IP addresses by DNS. These systems are provided at the application layer since the translation might involve sophisticated logic and require access and processing of a large amount of data. Separating these functions from the network layer can keep the network simple and efficient. Such a separation is also desired in the design for the object resolution service in ICN– the application-layer object resolution systems provide intelligent translation from keywords to names while the network then propagates the query (Interest in NDN) identified by the name and retrieve the data in a scalable and efficient manner.

In addition, retrieving of information that is typically done on the "web" using the Internet currently, can also be optimized by exploiting the capabilities of ICN. The benefits include:

- **Diversity:** With multiple copies of a same data "aggregated" to be the same name, the results of the resolution system can potentially be more scalable compared to the IP-based solutions.
- **Service scalability:** Service distribution and load balancing can be easily achieved in ICN. The in-network cache can further reduce the content server load by leveraging temporal locality of the search queries.
- **Retrieval/Dissemination efficiency:** The data retrieval and publication can be benefited by the name-based routing and in-network caches.
- **Security:** The integrity of the content and provenance can be assured by the signature

from the original content provider associated with the content.

This chapter presents ORICE, an architectural design for Object Resolution services in Information-Centric Environment. A resolution service is also presented using the architecture to help COPSS subscribers to translate keywords to the CDs (names) they might be interested in, and also help find related CDs of the publication when a provider wants to publish the data. This resolution system targets to assist both pub/sub applications and query/response applications.

To allow multiple resolution systems to work on the same name/CD space, the design separates the logic of name/CD certification (that manages the name/CD space) and the logic of object resolution (that finds proper names/CDs in the name/CD space).

9.2 Requirements

With the growth of the existing object resolution systems and the higher demands for better resolution services, an efficient object resolution architecture should meet the following requirements:

- **Intelligence:** Since the object resolution systems (e.g., search engines, recommendation systems) usually involve sophisticated AI logic and might need to access a large database while processing user requests, the architecture should allow such intelligence implemented in a flexible manner while not affecting data transmission efficiency.
- **Diversity:** Data consumers usually have different preferences on the resolution services, e.g., researchers would prefer papers to films while looking for technical material; artists might favor images than texts while looking for inspiration. To satisfy these preferences, many different resolution systems are deployed in the Internet. Similarly, the target architecture should also provide support for the diversity of resolution systems.
- **Scalability:** Modern object resolution systems usually have to deal with large amount of user requests. Many of them take advantage of distributed systems (e.g., MapReduce, NoSQL database) to provide scalable services. A desired object resolution architecture should allow the distributed systems easily designed and deployed in the network to satisfy varying number of user requests.
- **Name Space Maintenance:** A topic-based name space which is in the form of an ontology is more intuitive to the users compared to other kind of name spaces like location-based ones. Such name spaces can also maximize the data dissemination efficiency and minimize the forwarding overhead. Consider a name space having

`/sports` as a top-layer interest and `/sports/football`, `/sports/basketball`, etc. as its children. When a data consumer tries to express the interest of "everything related to sports", (s)he can choose to use the top-layer interest instead of all the lower-layer names. This aggregation reduces the entries (states) stored in the network and also simplifies procedure with which users express interests. However, an unorganized name space can do the contrary – a user might have to use many interests just to satisfy one requirement. Therefore, a name space maintenance (certification) system should be enabled in the resolution architecture to optimize the organization of the topic ontology. At the same time, this system can also help to authorize the data propagation and retrieval.

- **Efficiency:** Object resolution systems need to be backed by efficient data delivery. The underlying network should be responsible for propagating interests and data in a scalable and efficient manner, which in turn renders the object resolution system efficient.

9.3 Object-Resolution Architecture Design

This section describes the design of ORICE to meet the aforementioned requirements.

9.3.1 Separate Network and Application Layer Functionality

There are two main trends in the object resolution architecture designs – service network and service applications. The service network designs (e.g., ONYX [15]) process user requests on every node (broker) in the network and disseminate interests/data in a hop-by-hop manner (by calculating XML predicates). This kind of design has the benefit of lower network traffic since the intermediate nodes can decide which interface to send the packet to. But they face the scalability issue as the states stored and the computation needed in the network can be overwhelming for deployable routers when the number of users increase.

The service application designs are dominant in current Internet. The object resolution services (search engines) are deployed as separate functions in the application layer. Data consumers express the features of interested data via explicit calls to these services. The services respond with identities of the candidate data objects in the form of URIs. These URIs are further translated into IP addresses by DNS before a real data request is sent to retrieve the data. Although these explicit calls consume more network traffic, the design keeps the network function simple and services easy to deploy – a change in the resolution logic does not have to be deployed on every network node. Such design also allows the

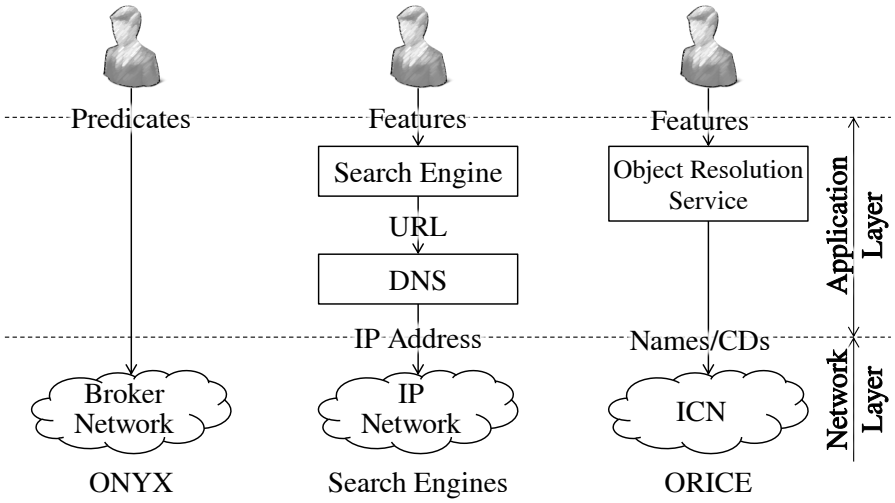


Figure 9.1: Layer division comparison among ONYX, search engines and ORICE.

services to have intra-system communication to access large amount of data and perform sophisticated calculation without affecting the performance of outer network.

To allow scalability and intelligent resolution functionality, ORICE chooses to adopt the service application design. The layer division of ONYX, search engines in IP and ORICE are compared in Fig. 9.1. In ORICE, the object resolution services are provided in the application layer. The users need to send explicit requests to the resolution servers to fetch a list of candidate data object identities. Different from IP solutions, ORICE users do not need to get the location of the data. Instead, they give the identity of the data (names/CDs) to ICN and the network retrieves the data in a smart way. The adoption of ICN can provide benefits like data diversity (every data identified by only one name), dissemination efficiency (name-based routing and in-network caches), security (each data is signed by the provider), etc.

9.3.2 Separate Name Space Management from Object Resolution

In the application layer, ORICE further separates the name space management functionality from the object resolution functionality. A logically centralized name certification service is created to support all the resolution services. The certification service controls the authorization and modification on the name space similar to DNS. Meanwhile, it keeps the resolution servers synchronized on the name space through a management channel. But different from DNS, the certification service does not perform the name to location translation which is left to the ICN at the underlay. This design enables more flexible service layouts

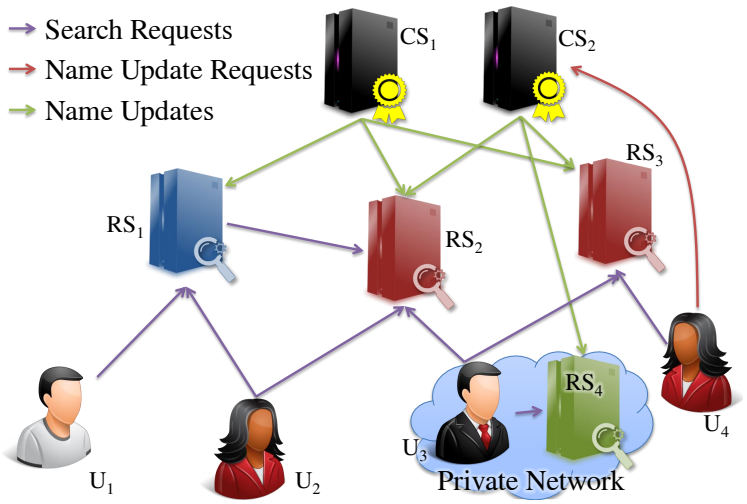


Figure 9.2: ORICE application-layer example.

and efficient name space management.

To better explain the application-layer design of ORICE, Fig. 9.2 is used to represent a resolution architecture having 2 certification servers (CS_1, CS_2), 4 resolution servers (RS_1-RS_4) providing 3 different object resolution services marked in different colors, and 4 users of the services (U_1-U_4). Note that since there is an ICN in the underlay, entities send requests/updates with names/CDs and the network takes the responsibility of forwarding these packets to proper destinations.

Consider a basic service layout which has only 1 Certification Server (CS_2) and 1 Resolution Server (RS_3). When a user (U_4) wants to get a name/CD for an interested data object, she sends an Interest with name `/ServicePrefix/Search/KEYWORDs`. The network will forward the request towards RS_3 and RS_3 responds with a set of candidate names/CDs. U_4 can then request for the data or subscribe to the CD accordingly.

When U_4 wants to generate a new CD or create a new data object in the name space (since the result is not satisfying), she can send an Interest with name `/CertPrefix/NEW_NAME` and content indicating the "add" action and the user identity if required. The network forwards the request towards CS_2 according to FIB. CS_2 responds a Data indicating if it approves or declines the request. On approving the request, CS_2 needs to multicast an update with CD `/ManageChannel/NEW_NAME` and the add action as content. All the resolution servers that subscribe to this CD can get the update and modify their local cache of the name space accordingly.

Other than the basic service layout described above, ORICE can also support the follow-

ing scenarios:

9.3.2.1 Multiple Resolution Services

Users might want different object resolution services due to the different preferences while these services operate on the same name space. In ORICE, every service provider can register a different prefix (`/ServicePrefix`) and listen to the requests under that prefix. E.g., the provider of RS_1 registers `/BLUE` while the provider of RS_2 and RS_3 registers `/RED`. Since all the resolution servers subscribe to the management channel `/ManageChannel`, the certification service can keep them synchronized on the name space.

ORICE also facilitates smaller service providers that only focus on a minority of users. Consider `/BLUE` (RS_1) as a service that serves resolutions related to ICN. It only needs to maintain the subset of the name space it might operate on, by subscribing to `/ManageChannel/ICN`. The reduced name space size can lower the storage and computation requirements of the service provider. RS_1 can also request other resolution services (e.g., `/RED`) on receiving some requests it cannot handle.

9.3.2.2 Private Resolution Service

An object resolution server can also reside in a private network or even on the same machine with the client. But since the resolution server still subscribes to the updates from the certification server(s), maintains an up-to-date name space. This service layout can provide extended privacy and more personalized resolution service.

Consider a film provider (U_2) who wants to find proper CDs for a movie which will be released soon. Even if there is a public available resolution service that can find related CDs for video clips, the film provider still faces the risk of content leakage since he has to upload the video to that service. In such situations, the film provider can deploy a resolution server in his private network (RS_4) instead of using a public one. To accept requests, RS_4 propagates its own prefix (`/GREEN`) just like public services but only within the private network. The communication between U_3 and RS_4 remains the same as public services.

9.3.2.3 Distributed Servers

In ORICE, both the certification and the resolution service can be distributed based on the name structure. E.g., CS_1 and CS_2 can listen to `/CertPrefix/Sports` and `/CertPrefix/` to handle different name space modification requests. A request of name modification in

sports will go to CS_1 based on the longest prefix match. Similarly, the requests under other prefixes can go to CS_2. When CS_1 receives too many requests under `/Sports/Football`, a new certification server can listen to `/CertPrefix/Sports/Foot-ball` to reduce the load on CS_1. Similar mechanism can be adopted on resolution servers. The automatic load-balancing provided by NDN can also be leveraged even when two resolution servers are listening to a same prefix.

9.4 ORICE Implementation

A prototype of ORICE is built on CCNx 0.8.0 and COPSS to show the feasibility and flexibility of the design. This prototype demonstrates how clients can search for CDs to publish and subscribe and how clients can get data objects from keywords. To support asynchronous data dissemination the broker design (described in §4.2 is also implemented. The broker, certification and resolution servers register their prefixes (i.e., `/BrokerPrefix`, `/CertPrefix` and `/ServicePrefix`) to handle the incoming interests so that CCNx can route each Interest to the respective destination based on its Name. With the pub/sub enhancement provided by COPSS, the resolution servers subscribe to the management channel (`/ManageChannel`) for the updates in the name space, and brokers subscribe to the responsible CDs to receive publications. The system can be described with the following three uses cases.

9.4.1 Subscribe

Subscription begins with a client searching for a list of CDs (s)he might be interested in. The implemented system accepts keywords as the search feature. The application generates an interest with name `/ServicePrefix/Search/KEYWORDs` and waits for the response from resolution server. The response contains a list of CDs and data objects related to the keywords. In order to verify the relevance of the CDs, the client can click on a CD to view the recent publications. The application will generate a request with name `/BrokerPrefix/CD` to retrieve most recent publications from the broker. The user interface with search result for "sigcomm" and the recent publications of CD `/ACM/SIGCOMM/2015` (selected) is shown in Fig. 9.3.

The client can decide to subscribe to either a CD from the suggestion or a new CD entered in the search field. The application will subscribe to the specified CD in COPSS directly if it is from the list of suggestions. Otherwise the application will request (with Interest name=`/CertPrefix/CD`, content="add") for an approval from certification server before

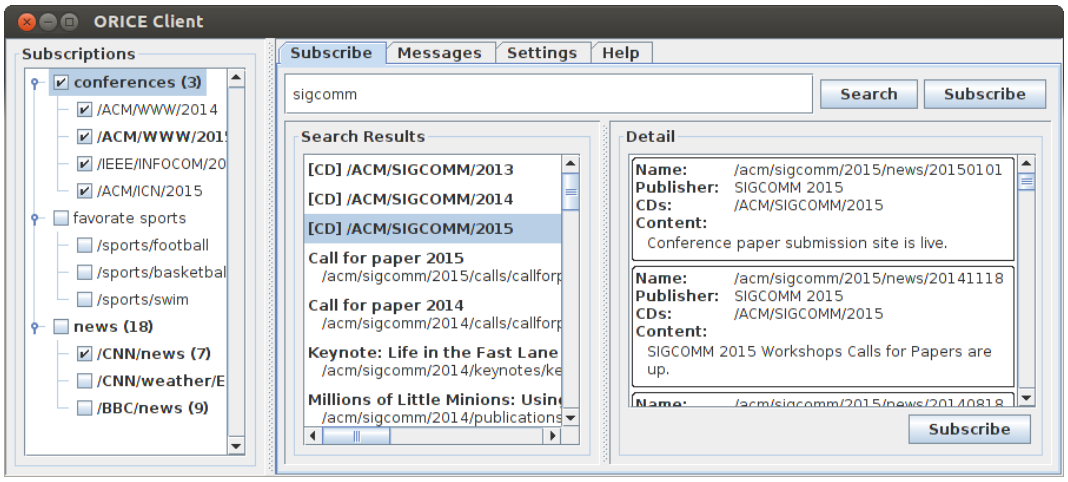


Figure 9.3: Subscription View.

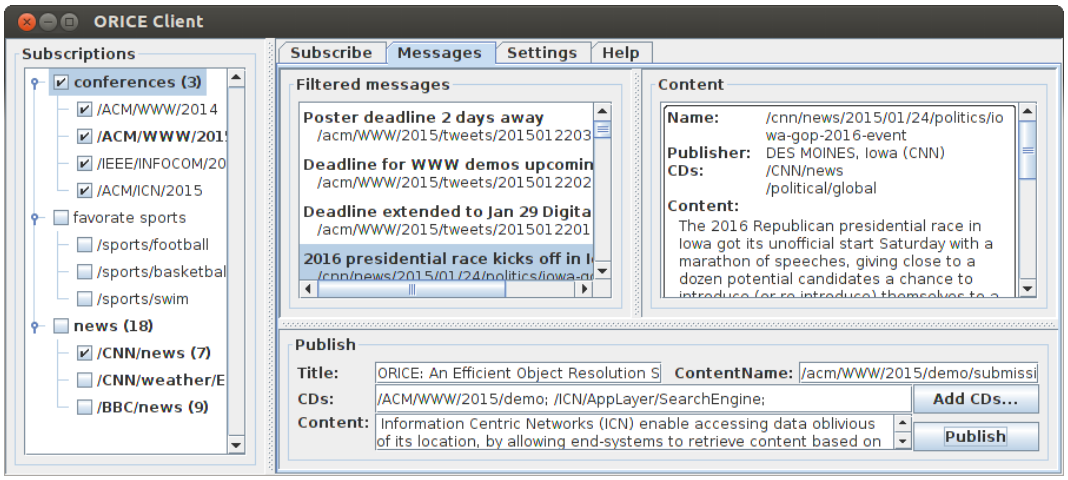


Figure 9.4: Publication and Messages View.

subscription. The certification server will notify the resolution servers with changes in the name space using a multicast (CD=`/ManageChannel/CD`).

9.4.2 Publish

During publication a client might need a list of related CDs. The application can retrieve the list of candidates from resolution server as in the previous use case. The client can decide to publish to the CD (s) from the suggestion or even new CDs. The application will send

the publication to the network directly if all the CDs are already existing. Otherwise, it will get approval from the certification server first.

The application allows the client to browse through the received publications categorized by CDs. The application will notify the client on receiving new publications. Fig. 9.4 shows a publication in progress along with the recent messages under the selected CDs.

9.4.3 Reconnect

To prevent clients from missing messages when they are offline, brokers are set up to receive all the publications. When a client comes back online, the application requests for any missed publications using name `/BrokerPrefix/CD//NAME_OF_LAST_MSG`. The broker will respond with the list of messages published after the last message.

9.5 Chapter Summary

This chapter presented ORICE, an object resolution architecture in ICN for service that enables end-systems to obtain the names of content of interest to users and applications. An architecture was designed to meets the requirements for a resolution service, which provides the foundation for exploiting ICN. With the efficiency derived from ICN, the overall architecture promises to make it more convenient for users, at higher performance than the current approaches of using search engines and IP networks. To demonstrate the feasibility of the design, a prototype of ORICE is implemented on an instance of ICN (CCNx with COPSS).

Chapter 10

Conclusion

The use of Internet has shifted from resource sharing to data dissemination and retrieval. ICNs are therefore proposed to meet this content-centric user behavior. However, most of the proposals like DONA, NetInf, NDN, etc., limit users in pure query/response communications. This renders pub/sub, another important content-centric communication model, end up being implemented via polling and becoming very inefficient.

After a comprehensive study on both pub/sub systems and ICN solutions, we realize that a network support for pub/sub is necessary and such network should be able to decouple pushers and subscribers, allow pushing, can accommodate user dynamicity and also be scalable and efficient.

This dissertation provides a full fledge architecture to support efficient pub/sub systems:

In the network layer, COPSS is proposed to provide basic best-effort pub/sub communication. COPSS is designed as an enhancement of NDN, therefore both query/response and pub/sub communications can be supported in a single network. With human readable, hierarchically structured CDs, COPSS allows publishers and subscribers focus on the contents instead of each others' location. The efficiency of COPSS is evaluated with Twitter and gaming – two emblematic content-oriented pub/sub applications. The result shows that with COPSS, these systems consumes less network traffic and provides better user experience (shorter latency).

In the transport layer, SAID is proposed for the sake of reliability and congestion control. Since SAID decouples reliability and congestion control, it ensures the applications to be network friendly even no matter if they need reliable delivery or not. SAID solved the long-standing difficulty of multicast congestion control. It ensures the flow only consumes fair share of bandwidth on each path of the dissemination tree. With efficient information-centric repair, SAID allows data consumers get missing packets from cache, provider and other consumers without concern of privacy and trust. The efficiency of SAID is evaluated with VoD and file delivery. The result shows that SAID can provide reliability with higher network utility and throughput.

In the application layer, ORICE is proposed to help publishers and subscribers get the CDs they are interested in. ORICE allows multiple resolution systems exist in a same network, including private ones and smaller ones that have special expertise. ORICE also suggests to have a name certification to allow the resolution systems work on a same name space.

To enable incremental deployment in the current network, hybrid-COPSS is proposed to allow COPSS (and NDN) to coexist with IP network in an efficient and scalable manner. Such a combination also provides benefit of efficiency – the information-centric functions are provided on key nodes while the other nodes focus on simpler and more efficient hash-

(IP-)based forwarding. To help ISPs identify the "key nodes", a cache deployment strategy is suggested with hybrid-COPSS for higher cache hit rate and lower data retrieval latency.

These components as a whole can provide better pub/sub support for applications like Twitter, gaming, online social network, etc. Along with the query/response communication provided by NDN, they can satisfy the users' information-centric requirements in the future.

On seeing the benefits provided by COPSS, this dissertation extends the concept of content-centricity towards a typical network management task – service chaining. FCSC is proposed to help redirect flows in service clouds. Thanks to the content-centric naming and forwarding, FCSC can provide benefits like flexibility, dynamicity, scalability and reliability.

10.1 Dissertation Impact

The author of this dissertation was the lead investigator and first author of several research papers. In particular, the work on designing, implementing and evaluating COPSS has been published in the following peer-reviewed international conference proceedings:

- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. COPSS: An Efficient Content Oriented Publish/Subscribe System. In: *Proceedings of the 7th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'11)*, New York, U.S.A., October 3–4, 2011.

The work on SAID has been published in:

- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Reliable Publish/Subscribe in Content-Centric Networks. In: *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking (ICN'13)*, Hongkong, China, August 12, 2013.

The work on hybrid-COPSS has been published in the following peer-reviewed international conference proceedings:

- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP. In: *Proceedings of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'12)*, Austin, U.S.A., October 29–30, 2012.
- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Coexist: A Hybrid Approach for Content Oriented Publish/Subscribe Systems. In: *Proceedings of*

the 2nd ACM SIGCOMM Workshop on Information-Centric Networking (ICN'12), Helsinki, Finland, August 17, 2012.

The work on ORICE has been published in:

- S. S. Adhatarao, **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. ORICE: An Architecture for Object Resolution Services in Information-Centric Environment. In: *Proceedings of the 21st IEEE International Workshops on Local and Metropolitan Area Networks (LANMAN'15)*, Beijing, China, April 22-24, 2015.

The design and evaluation of G-COPSS has been published in the following peer-reviewed international conference proceedings:

- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. G-COPSS: A Content Centric Communication Infrastructure for Gaming. In: *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS'12)*, Macau, China, June 18–21, 2012.
- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Gaming over COPSS: A Content Centric Communication Infrastructure for Gaming Applications. In: *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP'11), Poster Session*, Vancouver, Canada, October 17–20, 2011.
- **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications. In: *Proceedings of the 18th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN'11)*, North Carolina, U.S.A., October 13–14, 2011.

The work on FCSC has been published in:

- M. Arumaithurai, **J. Chen**, X. Fu, and K. K. Ramakrishnan. Exploiting ICN for Flexible Management of Software-Defined Networks. In: *Proceedings of the 1st ACM Conference on Information-Centric Networking (ICN'14)*, Pairs, France, September 24-26, 2014. (Best paper award).

Bibliography

[1] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content Based Routing with Elvin4. In *Proceedings AUUG2k*, 2000.

[2] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-based Networking. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004.

[3] 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 Computer Communication Review*, 2007.

[4] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J.D. Thornton. Named Data Networking (NDN) Project. Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, PARC, 2010.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009.

[6] K.V. Katsaros, G. Xylomenos, and G.C. Polyzos. MultiCache: an Overlay Architecture for Information-Centric Networking. *Elsevier, Computer Networks*, 55(4):936–947, 2011.

[7] V. Ramasubramanian, R. Peterson, and E. G. Sirer. Corona: A High Performance Publish-Subscribe System for the World Wide Web. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation*, 2006.

[8] PARC. Ccnx project. `http://www.ccnx.org/`.

[9] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax . In *IETF, RFC*, number 3986, January 2005.

[10] Wikipedia. Domain Name System. `http://en.wikipedia.org/wiki/Domain_Name_System`.

[11] C. Esteve, F. Verdi, and M. Magalhaes. Towards A New Generation of Information-Oriented Internetworking Architectures. In *Proceedings of the 2008 ACM CoNEXT Conference*, 2008.

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

[13] MobilityFirst Internet Architecture Project. `http://mobilityfirst.winlab.rutgers.edu/`.

[14] D. R. Cheriton and M. Gritter. TRIAD: A New Next-Generation Internet Architecture. 2000.

[15] Y. Diao, S. Rizvi, and M. J. Franklin. Towards An Internet-Scale XML Dissemination Service. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004.

[16] W. Fenner, D. Srivastava, K. K. Ramakrishnan, D. Srivastava, and Y. Zhang. XTreeNet: Scalable Overlay Networks for XML Content Dissemination and Querying. In *Web Content Caching and Distribution, 2005. WCW 2005. 10th International Workshop on*, 2005.

[17] Cache Capacity-aware CCN: Selective Caching and Cache-aware Routing, author=Lee, S. W. and Kim, D. and Ko, Y. B. and Kim, J. H. and Jang, M. W., booktitle=Global Communications Conference (GLOBECOM), 2013 IEEE, year=2013,.

[18] C. Bernardini, T. Silverston, and F. Olivier. Towards Popularity-based Caching in Content Centric Networks. In *RESCOM 2012*, 2012.

[19] D. Rossi and G. Rossini. Caching Performance of Content Centric Networks under Multi-path Routing (and more). *Relatório técnico, Telecom ParisTech*, 2011.

[20] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou. Modelling and Evaluation of CCN-Caching Trees. In *NETWORKING*, pages 78–91. 2011.

[21] Q. Wu, Z. Li, and G. Xie. CodingCache: Multipath-aware CCN Cache with Network Coding. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 2013.

[22] D. Perino and M. Varvello. A Reality Check for Content Centric Networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*,

2011.

[23] S. Arianfar, P. Nikander, and J. Ott. On Content-Centric Router Design and Implications. In *Proceedings of the Re-Architecting the Internet Workshop*, 2010.

[24] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, KC Ng, V. Sekar, and S. Shenker. Less Pain, Most of The Gain: Incrementally Deployable ICN. In *ACM SIGCOMM Computer Communication Review*, 2013.

[25] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[26] A. Adams and W. Siadak J. Nicholals. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). In *IETF RFC*, number 3973, January 2005.

[27] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). In *IETF RFC*, number 4601, August 2006.

[28] H. Holbrook and B. Cain. Source-specfic Multicast for IP. In *IETF RFC*, number 4607, August 2005.

[29] M. Macedonia and D. Brutzman. MBONE, the Multicast Backbone. *Naval Postgraduate School*, 1994.

[30] Y. Cui, B. Li, and K. Nahrstedt. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. *Selected Areas in Communications, IEEE Journal on*, 22(1):91 – 106, 2004.

[31] J. Jannotti, D. K. Gifford, and K. L. Johnsonand. Overcast: Reliable Multicasting with an Overlay Network. In *OSDI*, 2000.

[32] R. V. Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM TOCS*, 21:66–85, 2001.

[33] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg. SpiderCast: A Scalable Interest-Aware Overlay for Topic-Based Pub/Sub Communication. In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, 2007.

[34] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni. TERA: Topic-Based Event Routing for Peer-to-Peer Architectures. In *Proceedings of the*

2007 inaugural international conference on Distributed event-based systems, 2007.

[35] S. Voulgaris, E. Riviére, A. M. Kermarrec, and M. Van Steen. Sub-2-Sub: Self-Organizing Content-Based Publish and Subscribe for Dynamic and Large Scale Collaborative Networks. Research report, INRIA, December 2005.

[36] M. Ott, L. French, R. Mago, and D. Makwana. XML-Based Semantic Multicast Routing: an Overlay Network Architecture for Future Information Services. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, 2004.

[37] Xep-0060: Publish-subscribe. Xep-0060 (standard track, v 1.13), XMPP Standards Foundation, July 2010.

[38] S. Deering. Host Extensions for IP Multicasting. In *IETF RFC*, number 1112, August 1989.

[39] Z. Zhu and A. Afanasyev. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking. In *ICNP*, 2013.

[40] K. C. Almeroth, M. H. Ammar, and Z. Fei. Scalable Delivery of Web Pages Using Syclic Best-Effort Multicast. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1998.

[41] V. Aggarwal, R. Caldebank, V. Gopalakrishnan, R. Jana, K. K. Ramakrishnan, and F. Yu. The Effectiveness of Intelligent Scheduling for Multicast Video-on-Demand. In *Proceedings of the 17th ACM international conference on Multimedia*, 2009.

[42] D. Towsley. An Analysis of a Point-to-Multipoint Channel Using a Go-Back-N Error Control Protocol. *IEEE Transactions on Communications*, 33(3):282–285, 1985.

[43] K. Sabnani and M. Schwartz. Multidestination Protocols for Satellite Broadcast Channels. *Communications, IEEE Transactions on*, 33(3):232–240, 1985.

[44] A. Erramilli and R. P. Singh. A Reliable and Efficient Multicast Protocols for Broadband Broadcast Networks. In *ACM SIGCOMM Computer Communication Review*, 1987.

[45] S Ramakrishnan and BN Jain. A Negative Acknowledgement with Periodic Polling Protocol for Multicast over LANs. In *Proc. IEEE Infocom'87*, 1987.

[46] S. Paul, K. K. Sabnani, and D. M. Kristol. Multicast Transport Protocols for High Speed Networks. In *Network Protocols, 1994. Proceedings., 1994 International Conference on*, 1994.

[47] B. Adamson, C. Bormann, M. Handley, and J. Macker. Nack-Oriented Reliable Multicast (NORM) Transport Protocol. In *IETF RFC*, number 5740, November 2009.

[48] Luigi Rizzo. pgmcc: a TCP-Friendly Single-Rate Multicast Congestion Control Scheme. In *ACM SIGCOMM Computer Communication Review*, 2000.

[49] S. Floyd, V. Jacobson, C. G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application level Framing. *IEEE/ACM Transactions on Networking (TON)*, 5(6):784–803, 1997.

[50] Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000.

[51] S. Ratnasamy, A. Ermolinskiy, and S. Shenker. Revisiting IP Multicast. In *ACM SIGCOMM Computer Communication Review*, 2006.

[52] A. Malekpour, A. Carzaniga, F. Pedone, and G. Toffetti Carughi. End-to-End Reliability for Best-Effort Content-Based Publish/Subscribe Networks. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, 2011.

[53] J. C. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, 1996.

[54] D. M. Chiu, S. Hurst, M. Kadansky, and J. Wesley. TRAM: A tree-based reliable multicast protocol. 1998.

[55] N. G. Duffield, M. Grossglauser, and K. K. Ramakrishnan. Distrust and Privacy: Axioms for Multicast Congestion Control. In *Proceedings NOSSDAV'99*, 1999.

[56] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-Driven Layered Multicast. In *ACM SIGCOMM Computer Communication Review*, 1996.

[57] L. Vicisano, J. Crowcroft, and L. Rizzo. TCP-Like Congestion Control for Layered Multicast Data Transfer. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1998.

[58] G. Carofiglio, M. Gallo, and L. Muscariello. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, 2012.

[59] G. Carofiglio, M. Gallo, and L. Muscariello. Joint Hop-by-Hop and Receiver-Driven Interest Control Protocol for Content-Centric Networks. In *Proceedings of the second*

edition of the ICN workshop on Information-centric networking, 2012.

[60] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi. Transport-Layer Issues in Information Centric Networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, 2012.

[61] S. Arianfar, P. Nikander, L. Eggert, and J. Ott. ConTug: A Receiver-Driven Transport Protocol for Content-Centric Networks. In *ICNP (poster session)*, 2010.

[62] L. Saino, C. Cocora, and G. Pavlou. CCTCP: A Scalable Receiver-Driven Congestion Control Protocol for Content Centric Networking. In *Communications (ICC), 2013 IEEE International Conference on*, 2013.

[63] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos. Developing Information Networking Further: From PSIRP to PURSUIT. In *Broadband Communications, Networks, and Systems*, pages 1–13. 2012.

[64] W. C. Feng, F. Chang, W. C. Feng, and J. Walpole. A Traffic Characterization of Popular On-line Games. *Networking, IEEE/ACM Transactions on*, 13(3):488–500, 2005.

[65] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim. Second life and the new generation of virtual worlds. *IEEE Computer*, 41(9):46–53, 2008.

[66] F. Stenio, K. Carlos, S. Djamel, M. Josilene, and A. Rafael. Traffic Analysis Beyond This World: the Case of Second Life. In *17th International workshop on Network and operating systems support for digital audio and video, University of Illinois, Urbana-Champaign*, 2007.

[67] M. Varvello, S. Ferrari, E. Biersack, and C. Diot. Exploring Second Life. *IEEE/ACM Transactions of Networking*, 19(1):80–91, 2010.

[68] J. Keller and G. Simon. SOLIPSIS: A Massively Multi-Participant Virtual World. In *Proc. Int. Conf. Parallel & Distributed Techniques & Applications (PDPTA)*, 2003.

[69] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A Distributed Architecture for Online Multiplayer Games. In *NSDI*, 2006.

[70] S. Y. Hu, J. F. Chen, and T. H. Chen. VON: A Scalable Peer-to-Peer Network for Virtual Environments. *IEEE Network*, 20(4):22–31, 2006.

[71] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games.

In *ACM SIGCOMM Computer Communication Review*, 2008.

[72] M. Varvello, S. Ferrari, E. Biersack, and C. Diot. Network and Systems Support for Games (NetGames), 2009 8th Annual Workshop on. In *Network and Systems Support for Games (NetGames), 2009 8th Annual Workshop on*, 2009.

[73] M. Varvello, C. Diot, and E. Biersack. P2P Second Life: Experimental Validation using Kad. In *INFOCOM'09, IEEE*, 2009.

[74] A. R. Bharambe, S. Rao, and S. Seshan. Mercury: A Scalable Publish-Subscribe System for Internet Games. In *Proceedings of the 1st workshop on Network and system support for games*, 2002.

[75] M. Walfish, J. Stribling, M. N. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. In *OSDI*, 2004.

[76] B. E. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. In *IETF, RFC*, number 3234, February 2002.

[77] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011.

[78] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for The Internet. In *ACM SIGCOMM Computer Communication Review*, 2004.

[79] I. Castineyra and M. Steenstrup. The Nimrod Routing Architecture. In *IETF, RFC*, August 1996.

[80] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. In *IETF, RFC*, number 4423, May 2006.

[81] A. Myles, D. B. Johnson, and C. Perkins. A Mobile Host Protocol Supporting Route Optimization and Authentication. *Selected Areas in Communications, IEEE Journal on*, pages 839–849, 1995.

[82] B. Ford. Unmanaged Internet Protocol: Taming the Edge Network Management Crisis. *ACM SIGCOMM Computer Communication Review*, 2004.

[83] M. Walfish and H. Balakrishnan. Untangling the Web from DNS. In *Networked System Design and Implementation (NSDI)*, 2004.

[84] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection

Infrastructure. In *ACM SIGCOMM Computer Communication Review*, 2002.

[85] CISCO. Policy-Based Routing. Technical report.

[86] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patney, R. Subrahmaniam, M. Shirazipour, C. Truchan, and M. Tatipamula. StEERING: A Software-Defined Networking for Inline Service Chaining. In *ICNP*, 2013.

[87] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *ACM SIGCOMM Computer Communication Review*, 2013.

[88] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-Deployed Software Defined Wan. In *ACM SIGCOMM Computer Communication Review*, 2013.

[89] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. NLSR: Named-data Link State Routing Protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, 2013.

[90] H. Dai, B. Liu, Y. Chen, and Y. Wang. On Pending Interest Table in Named Data Networking. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, 2012.

[91] W. So, A. Narayanan, and D. Oran. Named Data Networking on a Router: Fast and DOS-resistant Forwarding with Hash Tables. In *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*, 2013.

[92] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The Bloomier Filter: an Efficient Data Structure for Static Support Lookup Tables. In *the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[93] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a Decentralized Network Coordinate System. In *ACM SIGCOMM Computer Communication Review*, 2004.

[94] M. Yuksel, K. K. Ramakrishnan, R. D. Doverspike, Rakesh K. Sinha, G. Li, K. N. Oikonomou, and D. Wang. Cross-layer Failure Restoration of IP Multicast with Applications to IPTV. *Computer Networks*, 55(9):2329–2351, 2011.

[95] Named Data Networking Project. `http://named-data.net/`.

[96] W. So, A. Narayanan, and D. Oran. Named data networking on a router: Fast and

dos-resistant forwarding with hash tables. In *Proceedings of the 9th ACM/IEEE symposium on Architectures for networking and communications systems (ANCS'13)*, 2013.

[97] Twitter. `http://www.twitter.com`.

[98] Wireshark. `http://www.wireshark.org`.

[99] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, 2010.

[100] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring Link Weights using End-to-End Measurements. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.

[101] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewar, J. D. Thornton, and R. L. Braynard. VoCCN: Voice-over Content-Centric Networks. In *Proceedings of the 2009 workshop on Re-architecting the internet*, 2009.

[102] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang. ACT: Audio Conference Tool over Named Data Networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, 2011.

[103] On-line Games. `http://www.thefengs.com/wuchang/work/cstrike/`.

[104] M. Claypool and K. Claypool. Latency and Player Actions in Online Games. *Communications of the ACM*, 49(11):40–45, 2006.

[105] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM Computer Communication Review*, 1988.

[106] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *ACM Transactions on Computer Systems (TOCS)*, 8(2):158–181, 1990.

[107] S. Oueslati, J. Roberts, and N. Sbihi. Flow-Aware Traffic Control for a Content-Centric Network. In *INFOCOM, 2012 Proceedings IEEE*, 2012.

[108] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 7(4):458–472, 1999.

[109] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, 1993.

[110] J. Gettys and K. Nichols. Bufferbloat: Dark Buffers in the Internet. *Communications of the ACM*, 55(1):57–65, 2012.

[111] R. J. Simcoe. Test Configurations for Fairness and Other Tests. In *ATM Forum/94-0557*, 1994.

[112] D. Rayburn. How Dynamic Site Acceleration Works, What Akamai and Cotendo Offer. `http://blog.streamingmedia.com/2010/10/how-dynamic-site-acceleration-works-what-akamai-and-cotendo-offer.html`, October 2010.

[113] D. Rayburn. It's Official: Akamai To Acquire Cotendo, Good For Akamai, Bad For Customers. `http://blog.streamingmedia.com/2011/12/its-official-akamai-to-acquire-content-good-for-akamai-bad-for-customers.html`, December 2011.

[114] J. Erman, A. Gerber, K. K. Ramakrishnan, S. Sen, and O. Spatscheck. Over the Top Video: The Gorilla in Cellular Networks. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011.

[115] J Erman and K. K. Ramakrishnan. Understanding the Super-sized Traffic of The Super Bowl. In *Proceedings of the 2013 conference on Internet measurement conference*, 2013.

[116] C. Cui, H. Deng, D. Telekom, U. Michel, H. Damker, I. Guardini, E. Demaria, R. Minerva, and A. Manzalini. Network Functions Virtualisation. In *SDN and OpenFlow World Congress*, 2012.

[117] J. Hwang, K. K. Ramakrishnan, and T. Wood. NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIX Association*, 2014.

[118] B. Heller, R. Sherwood, and N. McKeown. The Controller Placement Problem. *Proceedings of the first workshop on Hot topics in software defined networks*, 2012.

[119] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, pages 69–74, 2008.

[120] J. Moy. OSPF Version 2. In *IETF, RFC*, number 2178, July 1997.

[121] K. Lougheed and Y. Rekhter. A Border Gateway Protocol (BGP). In *IETF, RFC*,

number 1105, June 1989.

[122] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service. In *ACM SIGCOMM Computer Communication Review*, 2012.

[123] P. Pan, G. Swallow, A. Atlas, et al. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. In *IETF, RFC*, number 4090, May 2005.

[124] M. Yu, J. Rexford, M. J. Freedman, and J. Wang. Scalable Flow-based Networking with DIFANE. In *ACM SIGCOMM Computer Communication Review*, 2010.

[125] Y. Wang, Y. Zu, T. Zhang, K. Peng, Q. Dong, B. Liu, W. Meng, H. Dai, X. Tian, Z. Xu, et al. Wire Speed Name Lookup: A GPU-Based Approach. In *NSDI*, 2013.

[126] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang. OSPFN: An OSPF Based Routing Protocol for Named Data Networking. Tech. Rep, University of Memphis and University of Arizona, 2012.

[127] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. In *IETF, RFC*, number 1195, December 1990.

[128] J. Abley and K. E. Lindqvist. Operation of Anycast Services. In *IETF, RFC*, number 4786, December 2006.

[129] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: A Distributed Control Platform for Large-scale Production Networks. In *OSDI*, 2010.

[130] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically Centralized?: State Distribution Trade-offs in Software Defined Networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, 2012.

[131] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An Online Social Network with User-Defined Privacy. In *ACM SIGCOMM Computer Communication Review*, 2009.

Chapter A

Content-centric Notification System (CNS)

A.1 Introduction

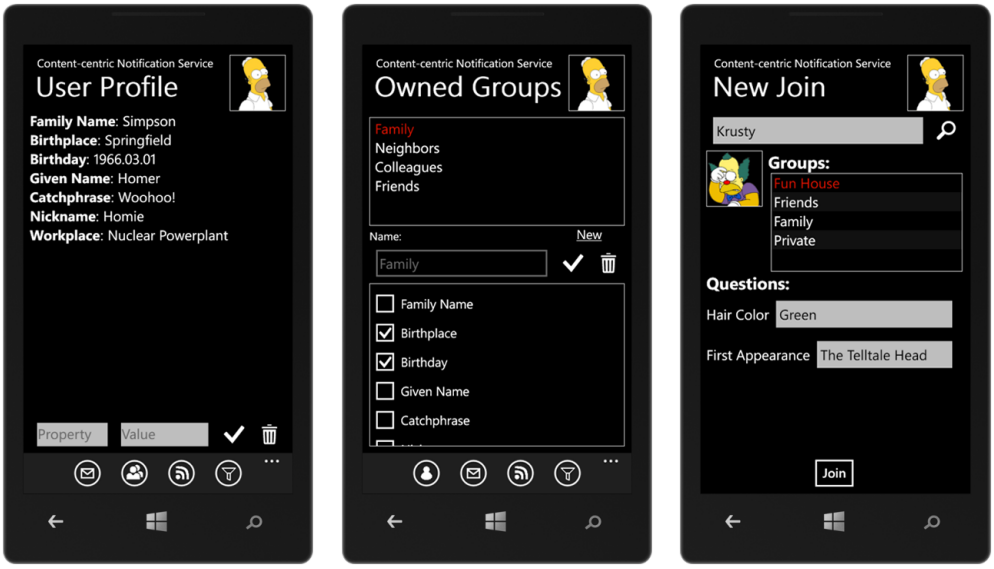
This chapter describes an ongoing work of *Content-centric Notification System* (CNS) that can be used for timely notification of essential information from a user to a group of subscribers that have requested (or potentially will request) that information. Such capabilities are envisaged to be desirable in a variety of circumstances, including disaster management, event notifications and many other scenarios where timely notification is key.

Effective authorization is a very important aspect in such an environment, i.e.certain publishers would prefer having control over who has visibility to their information. E.g., a publisher might want to publish personal data only to close friends, or another set of data to office colleagues and yet another set to all his acquaintances.

In some special cases, especially in cases where energy efficiency is critical (e.g., a user device's battery has little residual power), the publisher might not want to (or be able to) do the authorization himself. The system should allow a 3^{rd} party (either the network or a friend) to perform the authorization function.

CNS explores enhancing the COPSS with a preliminary authorization framework that could be used as the building block for a full-fledged notification service that could be used in future disaster management situations.

This work demonstrates how CNS facilitates pub/sub based notification capabilities, highlighting the enhancement of the basic ICN functionality to achieve *control efficiency* achieved with the use of hierarchical CDs. *Network efficiency* is achieved with the use of mutlicast, and *timeliness* is achieved with the push-based multicast approach. Furthermore, the basic authorization capability will be described to provide access control for a publisher's information.



(a) Profile management (b) Group management (c) Follow management

Figure A.1: Personal information management in CNS.

A.2 *Content-centric Notification System* (CNS)

CNS leverages the COPSS architecture for efficient data dissemination. Different modules of CNS are described in this section.

A.2.1 Publisher Management

Every user in CNS has a unique user (publisher) ID so that with a simple combination of `/APP/%UserID%`, CNS can get a unique prefix for that user. This can be part of the Content Name that is used by both publishers and subscribers. The Content Name may have other parameters such as the publisher name, signature etc., as specified in NDN. Moreover, it may also include CDs, information that goes beyond what is in the data, so that subscribers can identify the publisher and who has signed it; it may include a hash to ensure data integrity; it may also have some additional information that indicates the version or sequence number etc. The published data (e.g., a document) itself comprises many tokens (e.g., keywords), some of which are used as CDs by the publisher.

A user will have a **profile** which is a set of key-value pairs of his personal properties to uniquely identify the user. The user interface of profile management is shown in Fig. A.1a.

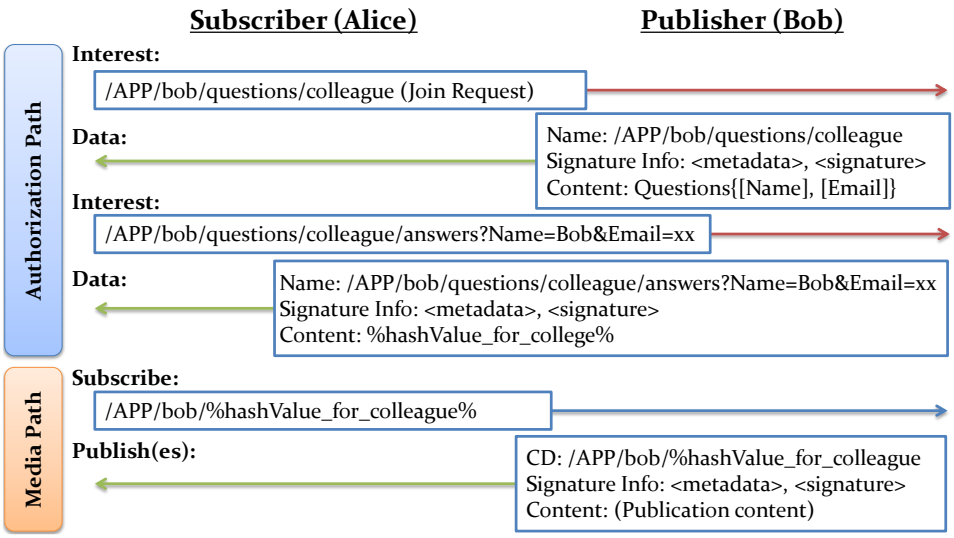


Figure A.2: **Protocol exchange for Alice follows Bob's Colleague group**.

The user may create several **groups** according to his or her preference (see Fig. A.1b). For every group, the user may choose a combination of the properties that identify the user, as the authorization token (challenge). Only the users who know the values of the authorization token associated with the group can subscribe to that group.

E.g., in the example shown in Fig. A.1, user Homer has the prefix of `/APP/homer`. He creates profile with properties of birthplace, birthday, given name, catchphrase, etc. Using a subset of these, he can have authorization tokens for groups that are his friends, colleagues, families, etc. For the "colleague" group, Homer may use his name and Workplace as the authorization token. But for people who want to subscribe to his "family" group, they need to know his birthplace and birthday as the authorization token.

A.2.2 Basic Authorization

To prevent malicious users from subscribing to (well-known) CDs, CD based group name is renamed in CNS with the use of hashes (on the group name + user ID) to provide a basic access control wherein only subscribers authorized by the publisher gain access to thes CDs. As shown in Fig. A.2, when a subscriber wants to subscribe, he sends a request to the publisher with the group name (`/APP/bob/questions/colleague` in the example), and the publisher responds with a challenge, e.g."what is my `Name` and `EmailId`?", where "Name" and "EmailId" are the two attributes in the profile the publisher selects as the authorization token used to authorize the subscribers. Assuming that the subscriber knows this informa-

tion (which means he is eligible to subscribe to that group), he responds with the appropriate values in the form of HTML parameters: `answers?Name=Bob&EmailId=bob@icn.org`. The publisher on verifying this data sends him the hash-based hierarchical CD, which the subscriber then subscribes to. If the subscriber cannot answer the questions correctly, the publisher will respond with an error instead. Fig. A.1c shows an example that Homer wants to join Krusty's "Fun House" group by answering challenge of "Hair Color" and "First Appearance".

Note that the authorization path uses the query/response approach of NDN. The network provides the functionality of caching on both challenge and the group hash value. On receiving a correct answer (in the form of an Interest with the same name as a previous correct answer), a router in NDN can respond directly if an existing copy of the hash value is already in the Content Store. This can save substantial processing and communication overhead on the publisher end (imagine the case a pop star that has to authorize a million fans who want to subscribe to his "fan" group, possibly with a cell phone!).

A.2.3 Data Dissemination and Offline Support

CNS also leverages COPSS to provide a large scale timely notification service. Along with the content published, the user includes the authorization token, which looks like a CD (e.g.`/APP/bob/%hashValue_for_colleague%`). Bob can publish to all his colleagues using a Publish packet with this authorization token. As defined in COPSS, Bob can send packets to multiple groups by putting all the needed authorization tokens in the same Publication packet. The network will ensure the dissemination of the information automatically.

To help new subscribers as well as users that come online after having been disconnected for a period of time, we leverage the 'broker' concept suggested in COPSS. The broker subscribes to all the messages published by the user (i.e., CD `/APP`) and stores them for a period of time. The new subscribers can query the broker on completing the authorization phase. The name of the Interest packet is in the form of:

`/Broker/APP/UserID/GroupHash/MessageRange`.

A subscriber can also get online periodically and request missing messages by polling the broker. This can help save energy on the subscriber's device (e.g., battery) in extreme cases. Fig. A.3 shows the groups Homer has followed and the messages he has received recently.

A.2.4 3^{rd}-Party Authorization:

In some scenarios, a publisher might want to delegate the task of authorization to a third party (a subscriber in the same group in our demo). When a publisher is not able to respond

(a) Follow management (b) Message view

Figure A.3: Group management and message view in CNS.

to authorization requests, a subscriber (S_{req}) could forward the request to another subscriber (S_{del}) that is in the same group. Since S_{del} knows the answer for joining the required group, he can decide if the answer provided by this new subscriber is correct to then provide the hash of the authorization token to S_{req}.

However, there can be security concerns associated with such transitive authorization schemes. Publishers might not want to delegate the authorization on some groups, nor do they want to delegate the authorization to some subscribers. Since our demo targets as a building block for a generic notification service, we have not yet addressed these issues in providing a 3^{rd} party authorization mechanisms, using approaches like a reputation systems. Instead, a simple solution is adopted in current version of CNS: a publisher can specify if a group is allowed to provide transitive authorization; and the 3^{rd} party authorization is done manually (i.e.S_{del} decides if S_{req} can join a group by clicking a button in our demo).

A.2.5 Group Hierarchy

It is very important for a group communication system to have the capability of distinguishing different groups of subscribers [131]. E.g., Bob might have friends, colleges, families, etc. Among all the friends, there might be football club members, book read-

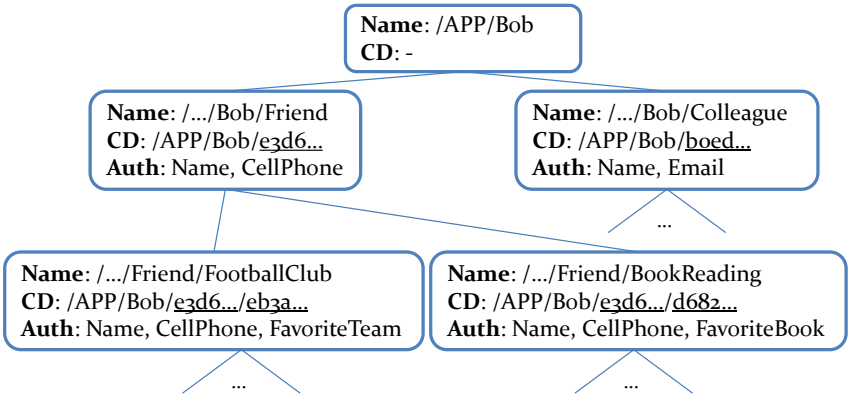


Figure A.4: Hierarchical group structure of user Bob (underlined parts are hash values).

ing group friends, etc. With the help of COPSS, it is quite simple to achieve hierarchical group management in CNS. Fig. A.4 shows the group hierarchical of user Bob. Note that he can pick different profile fields as authentication criteria for different groups. E.g., for `Friend/FootballClub`, he uses name, cell phone number and his favorite team as the authentication. So that only the friends in the club (that knows his interest in football) can join the group. Bob can publish news about football only to the `FootballClub` group, or he can also send a message to *all* of his friends via the `Friend` group.

A.3 Chapter Summary

This chapter describes CNS which leverages the benefit brought by COPSS for an efficient notification service, including the convenience of hierarchical group management, network efficiency and timeliness in delivering the notification. The work also shows a simple first-step authorization. Future work will include a more complete authorization, authentication, encryption and mobility support for a scalable notification service.

Curriculum Vitae

Contact Information

Jiachen Chen
Theodorheuss Str. 11
37075 Göttingen
jiachen@cs.uni-goettingen.de
https://www.net.informatik.uni-goettingen.de/people/jiachen_chen

Personal Information

Born on April 21st 1984 in Shanghai, China
Chinese Citizen
Marital Status: Single

Education

University of Göttingen, Göttingen, Germany

PhD Student, **Since 11/2010**
PCS Programme in Computer Science (PCS),
Georg-August University School of Science (GAUSS),
Institute of Computer Science, Computer Networks Group

- Dissertation Topic: *A Content-Oriented Architecture for Publish/Subscribe Systems*
- Dissertation Committee: Prof. Xiaoming Fu, Prof. Konrad Rieck, Prof. Edith Ngai, Prof. Carsten Damm, Prof. Folrentin Wörgötter

Fudan University, Shanghai, China

Master of Engineering, **07/2010**
Software School,
Performance and Information Research Center

- Thesis Topic: Fast Active Tabu Search and its Application to Image Classification
- Advisors: Prof. Junyu Niu, Prof. Hongyu Li

Bachelor of Science, **07/2007**
Software School

Refereed Conference Publications

S. Adhatarao, **J. Chen**, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. ORICE: An Architecture for Object Resolution Services in Information-Centric Environment. In: *Proceedings of the 21st IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN'15)*, Beijing, China, April 22-24, 2015.

M. Arumaithurai, **J. Chen**, X. Fu, and K. K. Ramakrishnan. Exploiting ICN for Flexible Management of Software-Defined Networks. In: *Proceedings of the 1st ACM Conference on Information-Centric Networking (ICN'14)*, Pairs, France, September 24-26, 2014. (Best paper award).

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. CNS: A Content-Centric Notification System. In: *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP'13), Demo Session*, Göttingen, Germany, October 7–11, 2013.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Reliable Publish/Subscribe in Content-Centric Networks. In: *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking (ICN'13)*, Hongkong, China, August 12, 2013.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP. In: *Proceedings of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'12)*, Austin, U.S.A., October 29–30, 2012.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Coexist: A Hybrid Approach for Content Oriented Publish/Subscribe Systems. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Information-Centric Networking (ICN'12)*, Helsinki, Finland, August 17, 2012.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. G-COPSS: A Content Centric Communication Infrastructure for Gaming. In: *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS'12)*, Macau, China, June 18–21, 2012.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. Gaming over COPSS: A Content Centric Communication Infrastructure for Gaming Applications. In: *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP'11), Poster Session*, Vancouver, Canada, October 17–20, 2011.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications. In: *Proceedings of the 18th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN'11)*, North Carolina, U.S.A., October 13–14, 2011.

J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. COPSS: An Efficient Content Oriented Publish/Subscribe System. In: *Proceedings of the 7th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'11)*, New York, U.S.A., October 3–4, 2011.

H. Li, J. Niu, **J. Chen**, and H. Liu. Entropy descriptor for image classification. In: *Proceedings of the 33rd Annual ACM SIGIR Conference (SIGIR'10), Poster Session*, Geneva, Switzerland, July 19–23, 2010.

Research Visits

Fudan University, Shanghai, China, **8/2013-10/2013**

- School of Computer Science, Fudan University, Shanghai, China,
- Visiting Researcher, DAAD Scholarship
- Host: Prof. Dr. Jin Zhao

Honors and Awards

- 2014: ICN 2014 Best Paper Award
- 2013: ACM SIGCOMM 2013 Student Travel Grant
- 2013: DAAD Travel Scholarship within the DAAD PPP projects
- 2011: IEEE ICDCS 2012 Student Travel Grant

Teaching Experience

Computer Networks Group, University of Göttingen, Göttingen, Germany

Teaching Assistant **Since 03/2011**

Advanced Computer Networks (Master level)

- Summer 2013 – present
- Responsible for several 2 hour weekly lectures.

Advanced Topics in Computer Networking (Master level)

- Winter 2011/12
- Responsible for several 2 hour weekly seminars.

Practical Course Advanced Networking (Master level)

- Summer 2011 – present
- Responsible for the supervision of student projects.

Seminar on Internet Technologies (Bachelor/Master level)

- Winter 2011/12 – present
- Responsible for the supervision of student projects.

Thesis Supervision **Since 09/2014**

- Sripriya Srikant Adhatarao. ORICE: Object Resolution Architecture in Information-Centric Environment. Master Thesis, ongoing.