An Intelligent and Powerful Data Plane Support To Enhance Future Communication

Habilitation

Kurzversion

Computer Networks Group
Institute of Computer Science
Faculty of Mathematics and Computer Science
Göttingen
Germany

vorgelegt von

Dr. rer. nat. Mayutan Arumaithurai
aus Batticaloa, Srilanka

arumaithurai@cs.uni-goettingen.de

Göttingen
im November 2017

Declaration

Ich versichere, die vorliegende Arbeit selbständig und nur unter Be-nutzung der angegebenen Hilfsmittel angefertigt zu haben.

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, November 2017.

Abstract

Users are primarily interested in obtaining content and do not care much about where they obtain the Content from. But, the Internet as it is currently designed is very host-centric and places importance on the hosts establishing connection between them. Therefore, when a certain piece of data needs to be obtained, the Internet facilitates reliable connection between the two nodes, i.e., the node interested in the data and the node with the data. If an established connection is broken, e.g. due to mobility, the infrastructure primarily focusses on re-establishing the broken connection. Recent technologies and solutions such as Content Delivery Networks (CDNs), Peer-to-peer (P2P) and cloud try to shift the focus on the content. However, they have limitations in features they could support due to the underlying reliance on the host-centric TCP/IP.

Information Centric Networking (ICN) is a new paradigm where the network provides users with named content, instead of communication channels between hosts. 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. A major assumption of many of the ICN solutions is the presence of a powerful data-plane that could be exploited to provide more functionality.

This dissertation work started at a time when research on ICN was at an early stage and many key issues were still open. Key areas that this dissertation work addressed are: 1) the shortcomings of existing solutions to provide a full-fledged solution for efficient pub/sub communication in ICN; 2) incremental deployment strategy; 3) an efficient framework to support real-time applications such as gaming; 4) a congestion control protocol for multicast communication; 5) inability of the existing solutions to support a disaster; and 6) how ICN could be used in other application scenarios such as Network Management.

This dissertation is influenced by and built on research contributions by other peers. Moreover it mainly comprises of peer-reviewed publications and therefore the solutions have been vetted by the community. However, this dissertation is by no means the final word on the proposed solutions. Instead, the main contribution of this dissertation is to provide potential solutions to key areas with the expectation that it would contribute to discussions, design and standardization effort pursued by the community and thereby help overcome the hurdles on ICN in order to make the vision of ICN a reality.

Acknowledgements

A number of people have contributed to this thesis in various ways. My work and studies towards this thesis have profited much from these positive contributions which have shaped and impacted the direction of my work.

I would like to thank Prof. Xiaoming Fu for his constant support and for giving me the opportunity to work with much freedom on these topics. I am also deeply grateful to Prof. K.K. Ramakrishnan, who was always there to support my work, especially during critical moments. His constructive criticism and honest opinions helped this work immensely.

During these years I am grateful to have had the opportunity to closely work together with very talented and passionate students. Dr. Jiachen Chen, a former PhD student of mine, deserves a special mention for his close collaboration on topics pertaining to this work. I would also like to mention Sripriya Srikant Adhatarao, Sameer Kulkarni and Eeran Maiti. The interesting and deep discussions between us on various scientific topics and problems, have had a significant influence in this work. I am also indebted to my students from various courses for their insightful questions that contributed to improving this work.

My colleagues at the Computer Network Group at the Georg-August-Universitaet Göttingenhave been a constant support and inspiration. In particular, I am thankful to the feedback received from Dr. David Koll, Dr. Konglin Zhu, Dr. Narisu Tao, Dr. Stephan Sigg, Dr. Xu Chen, Dr. Yang Chen, Dr. Lei Jiao, Osamah Barakat, Jie Li, Abhinandan S. Prasad and Dr. Lingjun Pu that help improve the quality of this work. Federica Poltronieri, Annette Kadziora, Heike Jachinke, Carmen Scherbaum and Gunnar Krull from the administrative support team deserve a special mention for helping me overcome hurdles.

Over the last several years, I have had the opportunity to cooperate with many great scientists which have had a strong impact on my work and the results presented in this document. A great thanks to the EU-Japan GreenICN team consisting of European and Japanese researchers for the great time we had while collaborating closely on research issues.

I would also like to take this opportunity to thank my wife Sunanda and my daughter Naina. Thank you for your patience, understanding and support. I would also like to thank my extended family for their never-ending support.

Finally, I would like to acknowledge funding received from: 1) the EU-JAPAN initiative by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement No. 608518 and NICT under Contract No. 167 for the project titled “GreenICN: Architecture and Applications of Green Information Centric Networking"; 2) the EU-JAPAN initiative by the Horizon 2020 initiative, Grant Agreement No. 723014 and NICT under Contract No. 167 for the project titled “ICN2020: Advancing ICN towards real-world deployment through research, innovative applications, and global scale experimentation"; and 3) the Volkswagen Foundation Project “Simulation Science Center”. These funding supported me and/or my collaborators to pursue our research interests and achieve good results that became part of this work to a large extent.

Contents

List of Figures

List of Tables

Acronyms

4WARD *Architecture and Design for the Future Internet*

ALTO *Application Layer Transport Optimization*

API *Application Programming Interface*

CCN *Content Centric Networking*

CDN *Content Delivery Network*

CDNs *Content Delivery Networks*

CNS *Content Notification System*

COMET *COntent Mediator architecture for content-aware nETworks*

COPSS *Content Oriented Publish Subscribe System*

DNS *Domain Name Systems*

FCSC *Function-Centric Service Chaining*

FIB *Forwarding Information Base*

G-COPSS *Gaming over Content Oriented Publish Subscribe System*

Hybrid-COPSS *Hybrid-Content Oriented Publish Subscribe System*

ICN *Information Centric Networking*

IP *Internet Protocol*

MMORPG *Massively Multiplayer Online Role Playing Game*

MobilityFirst *Moving towards a more robust, secure and agile Internet*

NDN *Named Data Networking*

NetInf *Network of Information*

NFV *Network Function Virtualization*

P2P *Peer-to-Peer*

PIT *Pending Interest Table*

PSIRP *Publish Subscribe Internet Routing Paradigm*

PURSUIT *Pursuing a Pub/Sub Internet*

SAID *Scalable and Adaptive Information Dissemination*

SAIL *Scalable and Adaptive Internet soLutions*

SDN *Software Defined Networking*

URI *Uniform Resource Identifier*

UHD *Ultra High Definition*

XIA *eXpressive Internet Architecture*

1 Introduction and Motivation

1.1 Motivation

The Internet is one of the key, mission-critical infrastructures of our society. Accordingly, both Internet traffic volume and the number of Internet applications are quickly growing. The data traffic increase is going to reach 1.6 zettabytes per year [1] and is mainly dominated by video content. In order to cope with such traffic growth recently Internet stakeholders have massively deployed new technologies, both in the access and in the core network.

The Internet was designed at a time when the most popular application scenario was phone conversations. Therefore, it was fundamentally designed to facilitate conversation or connection between two end devices. The Internet therefore is currently operating as a massive network of pipes that passively push bits between end-host machines, be it servers, end-user fixed or mobile devices, or sensors.

Currently, the Internet is predominantly used for content retrieval. The users are primarily interested in content and not where, i.e. the exact machine, they retrieve the content from. For example, a user watching a video on Youtube or Netflix does not care about the exact server or *Content Delivery Network* (CDN) that they receive the content from. Similarly, a user checking his Facebook update does not care about the exact Facebook server that is serving her, but more interested in the actual content. In fact, the current Internet architecture has evolved to meet the needs of a content focussed use scenario by introducing mechanisms such as CDN, *Domain Name Systems* (DNS) based load balancing and etc. However, the reliance on *Internet Protocol* (IP) restricts the network from efficiently meeting the requirements of current application scenarios. No one apart from the two communicating end-points “understand" what is being transferred, i.e., the network is not content-aware. This agnostic mode of operation affects several of the network’s key functionalities, for example, efficient content distribution and content-aware traffic engineering, but also restricts the evolution of others, e.g., mobility is not gracefully supported as attachment points change.

Information Centric Networking (ICN) [2, 3, 4, 5]emerged in recent years as an alternative to the current host-to-host communication paradigm. ICN shifts the focus to “what" is required (i.e., the actual content), instead of “where" it is obtained from (i.e., the IP location of where the content is stored). It therefore proposes direct communication between the end-hosts and the content itself. ICN puts the actual information or content in the forefront and leaves IP addresses and permanent content storage locations as secondary points of concern. In ICN, the network transfers individual, identifiable content chunks, instead of

Figure 1.1: An example depicting the difference between IP and ICN at a high level

unidentifiable data containers (i.e., IP packets). For instance, Figure 1.1 illustrates the case of a user requesting for the December 2015 version of Time Magazine. In the case of IP, he would: i) obtain a URI (via search engines or other means); ii) contact a DNS server to obtain the IP address of the server serving it; iii) use the obtained IP address to retrieve the content from the Internet. In the case of ICN, he would: i) obtain the URI (via search engines or other means); ii) use the URI directly in the network in order to retrieve the content from the best/closest source.

The basic functions of an ICN infrastructure are to: i) address content by adopting an addressing framework based on names, without a reference to the current content location (i.e., location-independent names); ii) route a user request, based only on the content-name, towards the "closest" location containing the required content; potential locations include not only the origin server of that content but also network caches or even devices of other users that downloaded the same content beforehand; iii) deliver the content back to the requesting host.

Expected advantages of the ICN paradigm include:

1. **Efficient content-routing:** ICN would enable ISPs to perform native content routing with improved reliability and scalability of content access. This would be a built-in facility of the network, which would transform the Internet to a native content distribution network, in contrast to CDN overlays as is the current practice.
2. **Ubiquitous caching:** In-network caching enabled today by off-the-shelf HTTP transparent proxies requires performing stateful operations. The burden of a stateful processing makes it expensive (both in terms of costs and in terms of operations) to deploy caches in nodes that handle a large number of user sessions. ICN would significantly improve efficiency, reliability and scalability of caching, especially for video, by judiciously utilizing caches at critical points in the network to exploit the benefits of named content.
3. **Simplified handling of mobile and multicast communication:** with ICN, in contrast to current mobility architectures, when a user changes point of attachment to the net-

work, she will simply ask for the next chunk of the content she is interested in, without the need for maintaining tunnels or re-routing from an anchor point and maintaining excessive state in the network; in fact, the next chunk may be provided by a different node than the one that it would have been used before the handover. Furthermore, multicast becomes an inherent capability in ICN, with content requested from a node being delivered to all interested receivers without the use of overlays.

4. **Support for time/space-decoupled** model of communications, simplifying implementations of pub/sub service models and allowing fragmented networks, or sets of devices to operate even when disconnected from the rest of the network (e.g. sensors networks, ad-hoc networks, vehicular networks, social gatherings, mobile networks on board vehicles, trains, planes, or networks stricken by disaster). This point is important to stimulate early take up of ICN in selected (and possibly isolated) environments.
5. **Simplified support for peer-to-peer communications:** ICN inherently supports communications between peers, without the need for application-layer overlays, as it is the case today. Users obtain the desired content from other users (or from caching nodes) thanks to location-independent content resolution and routing.
6. **Content-oriented security model:** securing the content itself, instead of securing the communication channels, allows for a stronger, more flexible and customizable protection of content and of user privacy. In addition, this is a necessary requirement for ICN: in-network caching requires embedding security information in the content, because content may arrive from any network node or user device; hence the senders cannot be trusted. Thus, end-users must be able to verify the integrity of the received data; caching nodes also make the same integrity check, to avoid caching of fake content.
7. **Content-oriented access control:** ICN can provide access to specific information items as a function of time, place (e.g., country), or profile of the user requesting the item. This functionality also allows implementing: i) access revocation (also known as digital forgetting), to ensure that content generated at one point may be removed from the system by the creator, ii) garbage collection, deleting from the network “expired"/obsolete contents.
8. **Content-oriented quality of service differentiation (and possibly pricing)**; provision of different performance in terms of both transmission and caching. Network operators (especially for mobile networks) already seek to differentiate content quality and priority, but they are forced to use deep packet inspection technologies. ICN would let operators differentiate the quality perceived by different services without complex, high-layer procedures, and offload their networks via caching.
9. **Create, deliver and consume contents in a modular and personalized way:** ICN provides opportunities for better customization of the interests of users and the content that is published by information providers. This will enable more efficient consumption of content because of better granularity in how content is described and identified,

and because of the ability of users to personalize the content that is delivered for their consumption.

10. **Network awareness of transferred content**, allowing network operators to better control information and related revenue flows, favouring competition between operators in the inter-domain market and better balancing the equilibrium of power across the entire eco-system, including over-the-top players.

A final overall advantage of ICN, which in a way comprehends the specific advantages listed above, is a simplification of network design, operation and management. Currently, content and service providers have to “patch" shortcomings and deficiencies of IP data delivery by using several “extra-IP" functionalities, such as HTTP proxies, CDNs, multi-homing and intra-domain multicast delivery, to name a few. This implies the involvement of several parties, the use of several specific protocols, the deployment of apposite devices and the interplay of different functionalities, often offered and managed by different companies and businesses. Apart from technical complexity, such operations also add management and administrative complexity. In an ICN environment, such diverse functions can be integrated in the network in a smooth and seamless way, e.g. by supporting inherently data replication, caching, multi-homing and multicast delivery. From this point of view, the design and deployment of a new functionality-rich API will play a key role.

1.2 Background

In order to satisfy the end users' need for content regardless of where they receive it from, there have been quite some patch work done to the current IP. Below, such IP based solutions that attempt to provide a content focussed solution are presented along with arguments as to why they are not capable of providing the full set of features that are envisioned by ICN. Then, some of the popular ICN approaches are described. Please take a look at the *Related Work* sections in each chapter to get a more comprehensive understanding of the short comings of the state of the art for a specific problem.

1.2.1 Content focussed application that exist in the Current Inernet

Though the Internet has been designed with a focus on end-to-end connectivity, many popular solutions try to circumvent this in order to shift the focus on to the content.

1.2.1.1 Peer-to-Peer (P2P)

Peer-to-peer (P2P) is a prime example of a content centric approach where users interested in a particular content, attempt to obtain it from other peers. Popular *Peer-to-Peer* (P2P) services such as BitTorrent make use of a tracker server to store the mapping between available content and which of the peers have it. A peer interested in a particular content contacts

a tracker server and obtains a list of peers that are serving that particular content. The advantage of P2P solutions is that the peers that are downloading a particular content can also choose to serve that content thereby increasing the number of sources for a particular content. P2P solutions thus provide users a wide range of options from where one could obtain the content. P2P services also facilitate the possibility for a requester of content to obtain it from multiple sources simultaneously.

Why they are not completely effective as a content-centric alternative? The disadvantage of a P2P based solution is that it is not topology aware and therefore proximity in terms of hops in the P2P topology does not in reality mean that they are close to each other in the routing topology. For instance, three peers that appear close to each other on the P2P topology could in fact be world apart with one of them being in the US, another in Europe and another in Japan. Therefore, in terms of the actual distance the content has to traverse, it might have to traverse a larger number of hops thereby increasing energy consumption. To overcome the problem of topology unawareness, solutions such as *Application Layer Transport Optimization* (ALTO) [6] have been proposed. The ALTO servers are envisioned to have information about the network topology and other factors and therefore support the clients in the peer selection process. The ALTO based solution looks promising, but cannot operate at small time scales since the updates it receives are usually averaged over larger time scales. Furthermore, the effectiveness of the ALTO solution depends on the level and accuracy of the information it obtains from the various network operators.

1.2.1.2 Content Delivery Network (CDN)

Content Delivery Networks (CDNs) are a distributed network of large storehouses for content and support the redistribution of content. The goal of a CDN is to serve content to end-users with high availability and high performance. CDNs help users to obtain their content faster and reduce the load on the original source of content as well as on the network. CDNs are in fact a group of servers present in data-centers that cache and serve content such as downloadable files (movies, software, documents), web-objects (images,scripts, text), location specific advertisements and other static content. They are also used by content providers to serve live-streaming and video on demand. CDNs were usually deployed in backbone networks, but recently, network operators have been deploying smaller scale CDNs closer to the edge to optimize traffic in their network as well as to provide content providers an alternative CDN service.

Why they are not completely effective as a content-centric alternative? CDNs are application layer solutions and therefore the client will have to establish connection to the content provider (e.g. HTTP), in order to receive a list of content and the corresponding CDN cache server where the content can be obtained from. Therefore, content providers might need to have their servers available for initial connection establishment and depend on CDNs to increase efficiency. Moreover, CDN based solutions can be sub-optimal since the CDN source has to be decided prior to the actual data transfer. A web server might

place data in a CDN and request users to go there, but a CDN server close to the user might not be used since the web server did not store content there. Dynamically deciding which content to cache in which CDN server is not straightforward. In the case of mobile nodes, this could result in larger inefficiency since a Uniform Resource Identifier (URI) that has been resolved earlier to a particular CDN might not be the optimal one once a node moves and since no URI resolution is involved after movement, the non optimal CDN is being used which could be a larger number of hops away.

1.2.1.3 Domain Name Systems (DNS)

DNS is another example of a content centricity approach. The DNS stores mapping between a *Uniform Resource Identifier* (URI) and the IP address where the content can be obtained. For instance, a user searching for "Google.com" can be redirected to any of the Google servers based on how the DNS is configured. The configuration could be such that the load is balanced or the request is redirected to the server closest to where the request was made. When an end user moves, he can renew his DNS request to receive from a server close to him.

Why they are not completely effective as a content-centric alternative? Nevertheless, the problem with DNS is that, it is performed at the beginning and is rarely updated during the session. This is also referred to as early-binding. Moreover, DNS updates are not possible in shorter time frames and makes sense only for big content providers.

1.2.2 Exisiting/ongoing work on ICN based solutions

1.2.2.1 Named Data Networking (NDN)

Named Data Networking (NDN) [7, 8], originally known as Content Centric Networking (CCN)[1] [3] is a popular ICN protocol where content/information is looked up and delivered according to its name without knowing the identity and location of the sender. NDN uses two packet types, `Interest` and `Data`. A consumer queries for named content by sending an `Interest` packet; a provider in turn responds with a `Data` packet. NDN requires a new forwarding engine instead of IP, which contains the *Forwarding Information Base* (FIB), Content Store (buffer memory which caches content) and *Pending Interest Table* (PIT). FIB is used to forward `Interest` packets toward potential source(s) of matching data. PIT keeps track of 'bread crumbs' of `Interest` (i.e., to support reverse-path forwarding), which the `Data` packets follow to reach the original requester(s). If multiple `Interest` packets arrive for the same data from multiple end-nodes, they will be aggregated in the PIT and served when the data arrives. The Content Store maintains a cache of the data in order to satisfy potential future requests for that data.

[1] In this work, we do not distinguish between *Named Data Networking* (NDN) and CCN since we only refer to the fundamental communication model.

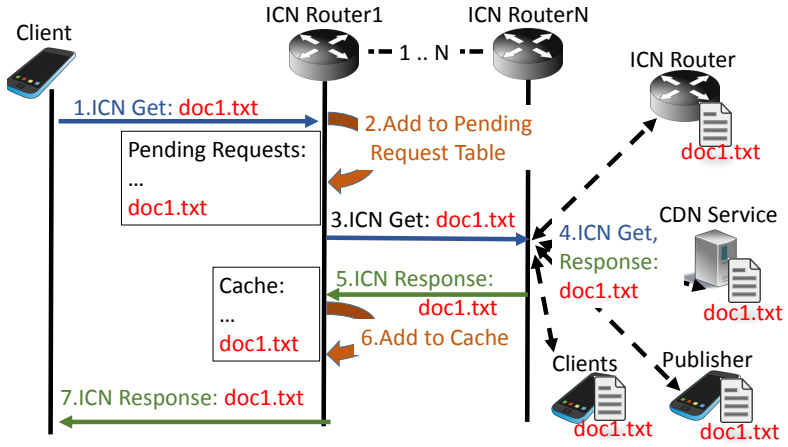


Figure 1.2: Message flow highlighting the name based data retrieval in ICN. Requested data can be obtained from one of the multiple sources of the data. In this case, the data can be obtained from the cache of an ICN router, from a CDN service, from other clients or directly from the original publisher (see Step-4). Step-2 shows that the `Interest` is added to the Pending Interest/Request Table (PIT) and in Step-6 we can observe that the data is stored in the cache of the router before being forwarded.

Figure 1.2 shows a simple message flow in an NDN architecture. Let us assume that the requester would like to get a movie file (doc1.txt) that is published by the publisher. In an IP network, the requester would make a DNS request to identify the IP address of the publisher and issue a request that would go all the way to the publisher. On the other hand, in ICN, the requester would issue a request (Step-1, i.e. ICN Get) for that movie file in an ICN environment. The first hop router has multiple options to deal with this request. If the movie is present in its cache, it could deliver it directly from it. Else, an ICN router could choose to forward it via any one of the paths such as those depicted in Figure 1.2 where one path leads to a CDN like content store, another to other clients that have the data, another to an ICN router cache and another to the publisher.

1.2.2.2 Ongoing/Completed Projects

The *Publish Subscribe Internet Routing Paradigm* (PSIRP) [9] project developed an information-centric network architecture based on a publish/subscribe paradigm. It proposed to replace the current Internet protocols entirely, applying a layer-less clean-slate architecture for routing, security, mobility and other basic network services. This was followed by the *Pursuing a Pub/Sub Internet* (PURSUIT) [10] project to address open issues such as resource control and advanced concepts for information scoping. PSIRP/PURSUIT

introduced several contributions on several aspects of information centric networking, e.g., publish/subscribe architecture (e.g., [11],[10]), fast forwarding strategies (e.g., [12]), and a new transport layer protocol (e.g., [13]), and mobility support (e.g., [14]).

The *eXpressive Internet Architecture* (XIA) [15] project proposed to create a single network that offers inherent support for communication between current communicating principals–including hosts, content, and services–while accommodating unknown future entities. For each type of principal, XIA defined a narrow waist that dictated the *Application Programming Interface* (API) for communication and the network communication mechanisms. The *COntent Mediator architecture for content-aware nETworks* (COMET) [16] project proposed a content-oriented architecture that could simplify content access and thereby support content distribution in a content and network-aware fashion.

The *Architecture and Design for the Future Internet* (4WARD) [17] project developed an ICN architecture called *Network of Information* (NetInf) [18]. It has an object model that can handle information at different abstraction levels, enabling the referencing of information independent of its encoding. The NetInf naming scheme provides name-data integrity and name persistency. The *Scalable and Adaptive Internet soLutions* (SAIL) [19] project continued developing NetInf where 4WARD left off. For instance, the naming scheme was revised, the object model was simplified, and the routing and name resolution framework has become more concrete with, for example, an inter-domain interface. *Moving towards a more robust, secure and agile Internet* (MobilityFirst) [20] is an ICN project that focusses predominantly on mobile users.

1.3 Dissertation Contributions and Approach

Vision of this thesis: The Internet evolves to an ICN based architecture so that all the stake holders can reap the benefits on an ICN architecture based Internet.

To transform this vision into reality, a key goal of this thesis was to address the open research problems of ICN. This research work was started at a time when research on ICN was at an early stage, with many key issues still open, including naming, routing, resource control, scalability, better network support for popular applications, network management and a migration path from the current Internet. The aim of this thesis therefore was to provide solutions to these pertinent research issues in ICN. Moreover, as we have observed in the aftermath of recent disasters, current communication infrastructure is incapable of supporting the scale as well as the presence of fragmentation. Therefore, this thesis believes that a futuristic architecture such as ICN must also consider disaster/ad-hoc communication in the fore front.

The thesis also intentionally steered away from proposing a concrete architecture since many well advanced and popular architectures were already present. Therefore, this thesis

focussed on solutions that are applicable to any ICN architecture that is currently available, such as NDN, *Content Centric Networking* (CCN) and PSIRP/PURSUIT and potentially to future ICN architectures with minor modifications. The proposed solutions have been generally accepted by the scientific community at large.

In order to provide solutions to these pertinent issues, this work adopted an application driven approach where applications were the focal point. These applications were used to derive requirements that were then used to design the solution and evaluate the approach.

1.3.1 Content Oriented Publish Subscribe System

Users increasingly desire access to information, ranging from news, financial markets, healthcare, to disaster relief and beyond, independent of who published it, where it is located, and often, when it was published. Publish/subscribe (pub/sub) systems are particularly suited for large scale information dissemination, and provide the flexibility for users to subscribe to information of interest, without being intimately tied to when that information is made available by publishers. With the use of an appropriate interface, users can select and filter the information desired so that they receive only what they are interested in, often irrespective of the publisher.

A couple of key requirements for a pub/sub system are efficiency and scalability. We observe that the ability to exploit multicast delivery is key to achieving efficiency, and to avoid wasting server and network resources. Scalability requirements come in multiple forms: the ability to accommodate a large number of publishers; the ability to accommodate a large number of subscribers; 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; allow for subscribers to not have to be connected to the network at all times, so that information production and reception by consumers can be asynchronous.

This work therefore proposes *Content Oriented Publish Subscribe System* (COPSS), an efficient content-centric pub/sub system that enhances CCN with push based multicast support. With the help of a Twitter-like pub/sub application, the requirements are derived and an efficient approach is proposed. See Chapter 2 for more details. It must be noted that COPSS formed the foundation for many of the follow up work performed in this thesis.

1.3.2 Hybrid Content Oriented Publish Subscribe System

Change is difficult. It is especially difficult in the Internet that spans the entire World. This work therefore believes that we must look for graceful incremental solutions, backward compatible with the current Internet, as opposed to risky clean slate and flag-day solutions. Therefore, this work examines how to evolve from an IP infrastructure to an ICN-oriented network by co-existing with the IP network. The aim is to support all the *functionality* a COPSS-enhanced ICN environment could provide (both Query/Response and Publish/Sub-

scribe) and provide users with name-oriented/content-oriented access to information. The network could however exploit cheaper IP-like 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), this 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, this work believes that a tightly integrated approach as proposed here provides the best of both worlds, with ICN routers at the edge and at selected points, and the core routers in the network only performing IP forwarding. See Chapter 3 for more details.

1.3.3 Gaming Over Content Oriented Publish Subscribe System

Massively Multiplayer Online Role Playing Game (MMORPG)[2] 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's 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

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

possibly unknown set) of participants poses difficult challenges even for a P2P oriented environment.

This work proposes the use of an ICN based architecture to provide network support for a MMORPG gaming application. While the *game logic* resides with the application server, the aim of this work is to ease the load on the game server. This is achieved by enhancing the network with ICN features to support the dissemination of the information to the correct users without the sender having to worry about who to send it to. COPSS (see Chapter 2) enhances CCN with a push based multicast capability and uses the notion of hierarchical Content Descriptors (CDs) that are employed by users to subscribe to information that is published by any end-system in the network. This facilitates a highly dynamic and large scale pub/sub environment and is able to deliver content in a timely manner. This focus of COPSS on content distribution rather than host-to-host connectivity removes the need for receivers to know and establish context with specific sources of information and for publishers to have an apriori knowledge of the intended recipients. This work proposes *Gaming over Content Oriented Publish Subscribe System* (G-COPSS), an extension to COPSS to make it suitable as a content centric communication infrastructure for a decentralized gaming environment. See Chapter 4 for more details.

1.3.4 Name Based Multicast Congestion Control Framework

In the application space, the Internet is fast becoming a multimedia information delivery platform. Although modern applications span a wide spectrum, multimedia delivery in general and video in particular is increasingly demanding of bandwidth (e.g., *Ultra High Definition* (UHD)) and extra computation (e.g., policy, transcoding, caching and shaping to accommodate user, device and last-mile characteristics) with minimal impact on latency. According to Cisco's white paper [1], video streaming and downloads are the primary contributors to the worldwide network traffic growth and it is expected that it will grow to more than 80% of all consumer Internet traffic by 2019. Mobile video traffic alone is expected to grow at an annual rate of 59% until 2019.

While video delivery is at the forefront of attention at the moment, this work envisions that demand will increase for multi-party interactive multimedia with very stringent quality of experience guarantees from mission-critical applications, e.g., crowdsourcing mobile multimedia applications (for sharing videos and experiences at large-scale events such as the Olympics), eBusiness (HQ-less businesses), eHealth (remote healthcare) and eEducation (virtual classes/campuses) and Security/access-control. These applications are expected to satisfy various requirements from perspectives of end-users, business and scalability and therefore offer many challenges to the underlying infrastructure. First and foremost, the end-users would expect a high quality of user experience (low latency, UHD, w/wo privacy, w/wo mobility). Providers of video delivery such as broadcasters and providers of CDN and user generated content would require efficient means to deliver the videos. Moreover, higher quality video encoding schemes such as 4K and 8K require more bandwidth

and higher quality in best-effort based inter-domain networks. While early versions of all of these applications are already being used today with the support of large server farms and data centers, this work argues that the Internet architecture in its current form will neither encourage nor be able to support widespread adoption of such applications/platforms. Moreover, these applications could have a large number of heterogenous receivers with varying receive rates and would have limited loss tolerance.

To promote efficient content delivery, in-network caching has been deployed in most ICN architectures. This allows the consumers to obtain the copies of transferred contents from the closer routers without visiting the original source. With in-network caching, ICN can facilitate multicasting of content, and can shorten the content delivery distance between content copies and consumers, reducing unnecessary network traffic. This work highlights a particularly thorny problem of receivers going *out-of-sync* that results in inefficiency and unfairness with heterogeneous receivers, when using existing ICN congestion control mechanisms with in-sequence delivery. Moreover, in this work, we enhance COPSS (COPSS enhances ICN solutions with push based multicast capability and also provides two-step communication for video delivery. See Chapter 5 for more details.) and other ICN solutions with a very efficient congestion control mechanism. The solutions developed for video communication are also application to other applications such as news and social media. See Chapter 5 for more details.

1.3.5 Name Based Disaster Communication Framework

An enormous earthquake hit Northeastern Japan (Tohoku areas) on March 11, 2011, and caused extensive damages including blackouts, fires, tsunamis and a nuclear crisis. The lack of information and means of communication caused the isolation of several Japanese cities. This impacted the safety and well-being of residents, and affected rescue work, evacuation activities, and the supply chain for food and other essential items. Even in the Tokyo area that is 300 Km away from the Tohoku area, more than 100,000 people became 'returner' refugees, who could not reach their homes because they had no means of public transportation (the Japanese government has estimated that more than 6.5 million people would become returner refugees if such a catastrophic disaster were to hit the Tokyo area).

That earthquake in Japan also showed that the current network is vulnerable against disasters and that mobile phones have become the lifelines for communication including safety confirmation. The aftermath of a disaster puts a high strain on available resources due to the need for communication by everyone. Authorities such as the President/Prime-Minister, local authorities, police, fire brigades, and rescue and medical personnel would like to inform the citizens of possible shelters, food, or even of impending danger. Relatives would like to communicate with each other and be informed about their well being. Affected citizens would like to make enquiries of food distribution centres, shelters or report trapped, missing people to the authorities. Moreover, damage to communication equipment, in addition to

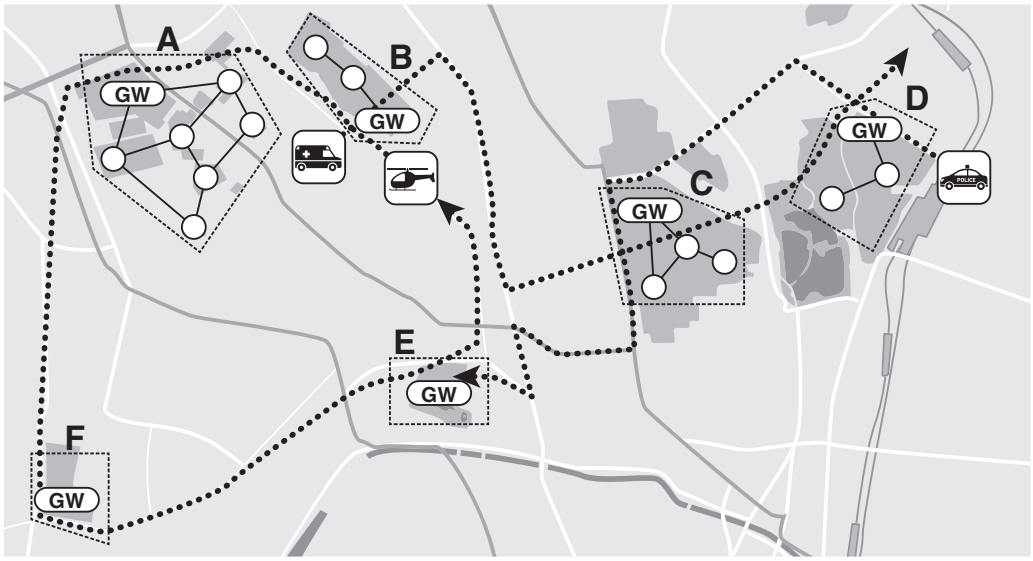


Figure 1.3: Fragmented networks identified by letters A – F are depicted at specific locations in a metropolitan area. Each fragmented network consists of one or several nodes that are able to communicate among themselves and has a gateway that is responsible for communication with other networks.

the already existing heavy demand for communication highlights the issue of fault-tolerance and energy efficiency.

Additionally, disasters caused by humans such as a terrorist attack [21] may need to be considered, i.e. disasters that are caused deliberately and willfully and have the element of human intent. In such cases, the perpetrators could be actively harming the network by launching a Denial-of-Service attack or by monitoring the network passively to obtain information exchanged, even after the main disaster itself has taken place. Unlike some natural disasters that are predictable using weather forecasting technologies and have a slower onset and occur in known geographical regions and seasons, terrorist attacks may occur suddenly without any advance warning. Nevertheless, there exist many commonalities between natural and human-induced disasters, particularly relating to response and recovery, communication, search and rescue, and coordination of volunteers.

The timely dissemination of information generated and requested by all the affected parties during and the immediate aftermath of a disaster is difficult to provide within the current context of global information aggregators (such as Google, Yahoo, Bing etc.) that need to index the vast amounts of specialized information related to the disaster. Specialized coverage of the situation and timely dissemination are key to successfully managing disaster situations.

As shown in Figure 1.3, a key assumption and a differentiating factor to related work on adhoc scenarios is that the government can play a huge role in facilitating communication during disasters. The government can plan and prepare for mechanisms and solutions that could come handy in the case of disaster management.

Although, a reliable and efficient communication infrastructure is most needed at times of disaster; network designers usually consider disaster management as an after thought. This results in communication breakdown in the aftermath of disasters. This work therefore wanted to place disaster applications at the forefront of the design. A key aspect to note is that ICN's reliance on reverse path forwarding in fact renders the core ICN design incapable of handling the presence of network fragmentation in the aftermath of disasters. Solutions designed for disaster management could also be used for ad-hoc communication environments. See Chapter 6 for more details.

1.3.6 Name Based Enhancement For Network Management

Service provider networks (and networks in general) are becoming increasingly complex. *Software Defined Networking* (SDN) aims to manage the network by decoupling the decision making from forwarding, *i.e.* by separating the control plane from the data plane. The logically centralized SDN controller(s) possess a global view of the network and can therefore provide a powerful tool for network management as compared to the traditional distributed architectures typical of the Internet. The data plane could therefore have simpler but more efficient switches that can focus on forwarding without having to worry about managing routing protocols.

However, this work argues that placing all the decision making functionality on the centralized controller is not efficient either. We, as network researchers, keep going through these cycles of designing a centralized/de-centralized/distributed version of the functionalities provided by the network. Let us for a moment, try to understand the thought process behind this. Any functionality can be designed in either a centralized or a decentralized/distributed manner. Depending on how the functionality is designed: i) the information available to make decisions vary; ii) the granularity and frequency of receiving/using the information vary; iii) different hardware constraints apply; iv) support to applications and network operators vary; v) cross layer interactions differ and solutions to bridge across layers are required; vi) will have different pros and cons, complexity, benefits; vii) and its flexibility for future applications vary.

Figure 1.4 illustrates the various factors that contribute to the design choices that influence if a functionality should be implemented in a centralized/de-centralized/distributed manner. **Application requirements** such as desired SLA, latency, throughput, availability, are a major driving force towards the design choice, they are used to derive a set of generic requirements that need to be supported by the network. Secondly, **Infrastructure/Network requirements and constraints** such as how long would it take to replace all the key components are another major contributing factor. *E.g.* the constraints to shift from 3G to

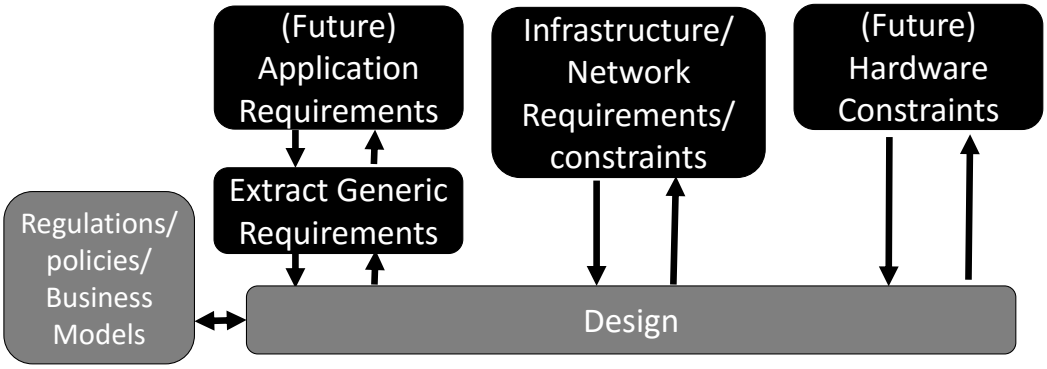


Figure 1.4: Factors that influence the design of a centralzied/de-centralized/distributed functionality

4G or from IPv4 to IPv6. Thirdly, **hardware constraints** such as available cache, TCAM space, memory and CPU resources of each and every key component such as the routers, base-stations and end-devices, is a key contributing factor. Finally, regulations, policies and business models should be factored in while making these design choices. Usually, these requirements and constraints are derived from what is available currently and what we envision would be the future. Therefore, when the assumptions made in determining the current and envisioned requirements and constraints change, it is time to revisit the design. A note of caution here is that current/envisoned constraints should not cripple innovation. A case for example is the advent of electric cars that need to circumvent challenges on various fronts such as technology, charging infrastructure, performance (in terms of speed and mileage) in order to compete with the well developed petrol/diesel cars. However, as more and more effort is invested, these challenges can be overcome. Similarly, clean slate solutions in the network infrastructure allow for such futuristic dreams. Moreover, hardware and infrastructure can be pushed beyond current boundaries (increase innovation) with compelling applications and good design.

While SDN researchers argue for the need to have simpler, but efficient switches while placing the complexity in the logically centralized controller, ICN argues for the exact opposite. ICN shifts the focus of the network from node location (IP, MAC, *etc.*) to data names. Such design enables name-based routing which forwards the requests of a specific name towards a best source of the data in terms of latency, available bandwidth, source load and *etc.* NDN [7] is one of the popular ICN solutions. NDN uses human-readable, hierarchical names such as `/video/cartoon/finding-nemo`. The forwarding engines perform the longest-prefix matching in the FIB to find the next-hop router closer to the data provider. The key difference of NDN to SDN is that NDN aims to exploit powerful router/switch hardware by pushing more functionality to the network layer. In essence, while SDN pro-

poses the need for *simpler switches*, NDN supports the need for *smarter routers/switches*.

This work therefore argues against the trend to place most of the intelligence on the logically centralized controller by taking an exemplary popular application – *Service Function Chaining (SFC) in the presence of Network Function Virtualization based middleboxes*. With this example application, this work argues that a centralized decision making entity curtails the potential that could be achieved by 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 *Network Function Virtualization* (NFV). This work then proposes a distributed decision making solution, *Function-Centric Service Chaining* (FCSC), that is designed to exploit ICN principles. See Chapter 7 for more details.

1.4 List of articles included in this work

The following publications contain the scientific work which represents this habilitation thesis. They are listed in the respective chapter they appear.

1.4.1 Chapter 2: Content Oriented Publish Subscribe System

- *COPSS: An Efficient Content Oriented Publish/Subscribe System*, Jiachen Chen, **Mayutan Arumaithurai**, Lei Jiao, Xiaoming Fu, K. K. Ramakrishnan, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ACM/IEEE ANCS 2011), Brooklyn, NY, USA, October 2011 (Acceptance rate: 32.2%) [22].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2011.ANCS/COPSS.pdf`

1.4.2 Chapter 3: Hybrid Content Oriented Publish Subscribe System

- *Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP*, Jiachen Chen, **Mayutan Arumaithurai**, Xiaoming Fu, K. K. Ramakrishnan, ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ACM/IEEE ANCS 2012), Austin, Texas, U.S.A, October 2012 (Acceptance rate: 22.9%) [23].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2012.ANCS/Hybrid-COPSS.pdf`
  Other related publications co-authored by the author: [24]

1.4.3 Chapter 4: Gaming Over Content Oriented Publish Subscribe System

- *G-COPSS: A Content Centric Communication Infrastructure for Gaming*, Jiachen Chen, **Mayutan Arumaithurai**, Xiaoming Fu, K. K. Ramakrishnan, The 32nd IEEE International Conference on Distributed Computing Systems (IEEE ICDCS 2012), Macau, China, June 2012 (Acceptance rate: 13.8%) [25].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2012.ICDCS/G-COPSS.pdf`
  Other related publications co-authored by the author: [26]

1.4.4 Chapter 5: Name Based Multicast Congestion Control Framework

- *SAID: A Scalable and Adaptive Information Dissemination Protocol in ICN*, Jiachen Chen, **Mayutan Arumaithurai**, Xiaoming Fu, K. K. Ramakrishnan, 3rd ACM Conference on Information-Centric Networking (ACM ICN 2016), Kyoto, Japan, To Appear in September 2016 (Acceptance rate: 27%) [27].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2016.ICN/SAID.pdf`
  Other related publications co-authored by the author: [28, 29, 30]

1.4.5 Chapter 6: Name Based Disaster Communication Framework

- *CNS: Content-oriented Notification Service for Managing Disasters*, Jiachen Chen, **Mayutan Arumaithurai**, Xiaoming Fu, K. K. Ramakrishnan, 3rd ACM Conference on Information-Centric Networking (ACM ICN 2016), Kyoto, Japan, To Appear in September 2016 (Acceptance rate: 27%) [31].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2016.ICN/CNS.pdf`
  Other related publications co-authored by the author: [32, 33, 34, 35, 36, 37, 38, 39, 40]

1.4.6 Chapter 7: Name Based Enhancement For Network Management

- *Exploiting ICN for Flexible Management of Software-Defined Networks*, **Mayutan Arumaithurai**, Jiachen Chen, Edo Monticelli, Xiaoming Fu, K. K. Ramakrishnan, 1st ACM Conference on Information-Centric Networking (ACM ICN 2014), Paris, France, September 2014 (Acceptance rate: 17%, Received the Best Paper Award) [41].
  Link to Paper: `https://projects.gwdg.de/projects/mayutan-public/repository/raw/0.Conferences/2014.ICN/FCSC.pdf`
  Other related publications co-authored by the author: [42, 43]

2 Content Oriented Publish Subscribe System

COPSS: An Efficient Content Oriented Publish/Subscribe System

Jiachen Chen†, Mayutan Arumaithurai†, Lei Jiao†, Xiaoming Fu†, K.K.Ramakrishnan‡
† Institute of Computer Science, University of Goettingen, Germany.
‡ AT&T Labs Research, Florham Park, NJ, U.S.A.
email: jchen3,arumaithurai,jiao,fu@cs.uni-goettingen.de,
kkrama@research.att.com

ABSTRACT

Content-Centric Networks (CCN) provide substantial flexibility for users to obtain information without regard to the source of the information or its current location. Publish/subscribe (pub/sub) systems have gained popularity in society to provide the convenience of removing the temporal dependency of the user having to indicate an interest each time he or she wants to receive a particular piece of related information. Currently, on the Internet, such pub/sub systems have been built on top of an IP-based network with the additional responsibility placed on the end-systems and servers to do the work of getting a piece of information to interested recipients. We propose Content-Oriented Pub/Sub System (COPSS) to achieve an efficient pub/sub capability for CCN. COPSS enhances the heretofore inherently pull-based CCN architectures proposed by integrating a push based multicast capability at the content-centric layer.

We emulate an application that is particularly emblematic of a pub/sub environment—Twitter—but one where subscribers are interested in content (e.g., identified by keywords), rather than tweets from a particular individual. Using trace-driven simulation, we demonstrate that our architecture can achieve a scalable and efficient content centric pub/sub network. The simulator is parameterized using the results of careful microbenchmarking of the open source CCN implementation and of standard IP based forwarding. Our 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 infrastructure.

1. INTRODUCTION

Users increasingly desire access to information, ranging from news, financial markets, healthcare, to disaster relief and beyond, independent of who published it, where it is located, and often, when it was published. Content centric networks (CCN) are intended to achieve this functionality with greater ease for users, greater scalability in terms of the amount of information disseminated as well as number of producers and consumers of information, and greater efficiency in terms of network and server resource utilization. Publish/subscribe (pub/sub) systems are particularly suited for large scale information dissemination, and provide the flexibility for users to subscribe to information of interest, without being intimately tied to when that information is made available by publishers. With the use of an appropriate interface, users can select and filter the information desired so that they receive only what they are interested in, often irrespective of the publisher.

A consumer may not wish (or 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, the consumer would rather prefer obtaining the data based on **Content Descriptors (CD)** such as a keyword, a tag, or a property of the content, such as the publisher identity, published date *etc*.

Intelligent end-systems and information aggregators (e.g., Google News and Yahoo! News, cable and satellite providers) have increasingly adapted their interfaces to provide a content-oriented pub/sub-based delivery method. However, these mechanisms are built on top of a centralized server-based framework and can also result in a waste of network resources as shown in [1, 2], since the Internet protocol 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. Having a network that is capable of delivering the information from any of the producers to all subscribers may overcome such limitations. However, unlike using multicast at the IP layer which can result in a substantial amount of duplicate information being delivered to the receiving end-system (which will have to be filtered out), it is desirable for the network to assist in delivering unique information to the subscriber.

There have been several recent proposals for CCN [3, 4, 5, 6]. One such effort is that of Named Data Networking (NDN) [3, 7]. NDN provides a substantial degree of flexibility for users and end-systems to obtain information without regard to their location or source. Exploiting caching, NDN improves the efficiency of content delivery. Subscribers can obtain the 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 the router, oblivious to the data source. However, this makes the content centric routers somewhat more heavy-weight as we will observe in our micro-benchmarking of such functionality. Moreover, due to its intrinsic design, we observe that enhancements are needed to efficiently support pub/sub applications using the NDN design. In the rest of the paper we use the term CCN to refer to the general content centric

based networking paradigm and use the term NDN to refer to the specific proposal named NDN [7].

A couple of key requirements for a pub/sub system are efficiency and scalability. We observe that the ability to exploit multicast delivery is key to achieving efficiency, and to avoid wasting server and network resources. Scalability requirements come in multiple forms: the ability to accommodate a large number of publishers; the ability to accommodate a large number of subscribers; 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; allow for subscribers to not have to be connected to the network at all times, so that information production and reception by consumers can be asynchronous.

In this paper, we develop COPSS, an efficient content-centric pub/sub system leveraging the advantages provided by CCN. We evaluate the performance of COPSS by using a decentralized Twitter-like application and show performance gains in terms of aggregate network load, publisher load and subscriber experience.

The key novelties of COPSS to provide a full fledged and efficient content delivery platform for pub-sub applications include:

- COPSS supports the notion of Content Descriptor (CD) [8, 9] based publishing and subscription. A CD goes beyond name-based [3] and topic-based [10] content identification and allows for contextual identification of information and supports ontologies and hierarchies in specifying interests.
- COPSS provides support for 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 published) and facilitates the creation of new publishers and subscribers. This is analogous to recent events in Twitter wherein people belonging to the affected region were able to behave as publishers.
- COPSS provides a push based multicast capability to be able to deliver the content in a timely manner in addition to leveraging the NDN's inherent pull-based information delivery model. COPSS does that in a scalable and reliable manner.
- COPSS is designed to provide additional features for subscribers that are offline and a 2-step delivery model that allow information publishers to exercise policy control, access control (i.e., which subscribers are allowed to access which information) and a snippet based dissemination of large pieces of content in a scalable manner.
- COPSS also addresses the need to evolve from our current IP-centered network infrastructure to a content-centric network.

We review related work in §2. In §3, we identify the requirements of an efficient pub/sub system, provide a short background of NDN, results of a microbenchmark test performed and discuss its shortcomings as an efficient pub/sub system. We present the COPSS design in §4 and evaluation results are given in §5. We conclude our work and outline further work in §6.

2. RELATED WORK

Existing work on pub/sub systems 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 proxy) 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 mechanisms require the knowledge of the identity (DNS/IP address) of publishers (or servers acting as the proxy).

In contrast, traditional push-based approaches maintain long-lived TCP connections (Elvin [11]) or notify subscribers via other means such as instant messaging (Corona [2]) or Rendezvous nodes (PSIRP [12]). Both approaches have scalability issues since it requires 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 Network Address Translators (NATs) makes it impractical for every subscriber to have global visibility, thereby complicating push based mechanisms. Overlay based pub/sub approaches like Astrolabe [13] and SpiderCast [10] are agnostic of the underlying topology and therefore cause a lot of extra overhead.

To overcome the limitation of these approaches where a subscription requires the knowledge of every content source, approaches such as ONYX [14], TERA [15], SpiderCast [10], and Sub-2-Sub [16] have been proposed as *topic/content-based* systems. In such systems, users express their interest in content rather than sources (*e.g.*, to a publisher in Twitter[1]). COPSS adopts a **Content Descriptor (CD)** based approach wherein a CD could refer to a keyword, tag, property of the content and *etc.*; similar to that adopted by XTreeNet [8] and SEMANDEX [9]. RSS feeds and XMPP pub/sub [17] are used to publish frequently updated content such as news headlines, blog entries and etc. and allows 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.

To our knowledge, there is no prior work which aims to build the content delivery network for efficient pub/sub. NDN [7] and native IP multicast [18, 19, 20] also provides an efficient delivery mechanism, but are not able to serve as an efficient full-fledge content-based pub/sub system as shown in §3. This paper proposes COPSS to fill this gap.

3. PROBLEM STATEMENT

We first describe the requirements that an efficient pub/sub content delivery system has to address. Then, we examine why existing IP multicast, overlay multicast and the current NDN solutions may be inadequate.

[1]http://twitter.com/

3.1 Requirements

An efficient pub/sub information delivery system ("the target system") needs to support:

- **Push enabled dissemination:** To ensure that subscribers receive information in a timely manner, the target system must provide the ability for publishers to push information to online subscribers interested in it. Such timely dissemination is useful in many scenarios such as disaster (e.g., Tsunami) warnings, stock market information, news and gaming.

- **Decouple publishers and subscribers:** As the number of publishers and subscribers increases, it is important for the network to be content-centric (using content names rather than addresses for routing), while still providing the appropriate association between them (publishers need not know who the subscribers are, and vice versa). Furthermore, each subscriber may be a publisher as well (e.g., Twitter allows users to be both subscribers and publishers of data).

- **Scalability:** The target system must handle a large number of publishers and subscribers. Minimizing the amount of state maintained in the network, ensuring the load on the publisher grows slowly (sub-linearly) with the number subscribers, the load on subscribers also grows 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. We also recognize the need to accommodate a very large range in the amount of information that may be disseminated, and the need for all elements of the target system in a content centric environment to scale in a manageable way.

- **Efficiency:** The system must utilize network and server resources efficiently. It is desirable that content is not transmitted multiple times by a server or on a link. Furthermore, the overhead on publisher and subscriber end-points to query unnecessarily for information must be minimized.

- **Incremental deployment:** It is desirable that the system be incrementally deployable as we transition from an IP (packet-based) to a content-centric environment. The target system must ensure that its features are beneficial for early adopters, and provide a seamless transfer from an IP dominated environment to a content-centric environment.

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

- **Support hierarchies and context in naming content:** We believe 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 [8]).

- **Supporting two-step dissemination for policy control and efficiency:** We recognize the 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 target system 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.

- **Subscriber offline support:** Another typical characteristic of pub-sub environments is that subscribers could be offline at the time the data is published. There is clearly a need for asynchronous delivery of information in a pub/sub environment in an efficient, seamless and scalable manner. The system needs to allow users who were offline to retrieve the data that they have missed. It should also allow new subscribers to retrieve previously published content that they are interested in. We envisage a server that stores all the content published. While important, the storage capacity and policy for replacement is beyond the scope of what we are able to address here in this paper.

3.2 Why Does IP/Overlay Multicast Fall Short as an Efficient Pub/Sub Platform?

IP multicast [18] is another 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 [19] construct and maintain a tree from each sender to all receivers of a multicast group. However, IP multicast isn't an efficient pub/sub delivery mechanism for several reasons: 1) IP multicast is designed for delivery of packets to connected end-points. Dealing with disconnected operation (when subscribers are offline) would have to be an application layer issue. Overlay multicast solutions such as [21, 22, 23] are agnostic of the underlying network topology, usually relying on multiple unicasts in the underlay path and are therefore also inefficient as a pub/sub delivery mechanism. 2) The somewhat limited multicast group address space makes it difficult to support a direct mapping of CDs to IP multicast addresses. 3) Current IP multicast is not able to exploit relationships between information elements, such as CDs. CDs may be hierarchical or may have a contextual relationship, which enables multiple CDs to be mapped to a group. For example, consider a publisher that sends a message to all the subscribers interested in $football$, and subscribers who are interested in receiving messages about all $sports$. The message from the publisher will have to be sent to two distinct IP multicast groups. If there happens to be a subscriber of messages on $sports$ and $football$, (s)he will receive the same message twice and will have to perform redundancy elimination in the application layer. The result is a waste in network traffic and processing at both ends.

3.3 State of the Art: Named Data Networks

NDN: Technical background

NDN [7] has been proposed as a content centric network architecture. Content sources register their availability of content by prefix (akin to a URL), and these prefixes are announced for global reachability (in a manner similar to BGP in inter-domain IP routing). There are two kinds of

packets: *Interest* and *Data* (i.e., content). 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. Both the Interest packet and a Data packet have a content name. For an Interest packet, this name is the name of the requested data; for a Data packet, the name identifies the data contained in this packet. The current design of NDN adopts a URL-like scheme for the content name, e.g., a multimedia item may be named as */uni-goettingen/introduction.mp3*. An NDN router has three data structures (see Fig. 2): 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 (CS)* that caches content from a provider upstream. The router forwards an Interest by doing a longest-match lookup in the FIB on the content name in the Interest. When forwarding an Interest, the router also records the name of this Interest and the (inter)face from which this Interest comes into PIT. NDN only routes Interest packets. Data packets follow the reverse path established by the corresponding Interest. When an Interest packet arrives at a router, first the CS 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 is no matched entry, then the Interest is sent out on all the corresponding outgoing faces. When a Data packet arrives, the CS is checked first. A match implies that this data is a duplicate of what is cached and the packet is discarded. Otherwise, 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 CS and sent out on each face from which the Interest arrived.

Difficulties with NDN for pub/sub systems: Multicast

NDN has limited intrinsic support for pub/sub systems, a critical need in a content centric environment. The aggregation of pending Interests at routers achieves efficient dissemination of information from NDN nodes. But this aggregation is similar to a cache hit in a content distribution network (CDN) cache, which occurs only if subscribers send their Interests with some temporal locality. Thus it avoids multiple Interest queries having to be processed directly by the content provider. Note however that this is still a pull-based information delivery method and depends both on temporal locality of interests and a large enough cache to achieve effective caching in the (content centric) network. On the other hand, native multicast support allows for a much more scalable push-based pub/sub environment, since it is not sensitive to issues such as the cycling of the cache when a large amount of information is disseminated. In COPSS we strive to achieve a full fledged push-based multicast capability in addition to the efficient query based information dissemination available in NDN.

Table 1: Forwarding performance (μs)[std. dev]

	CCNx	UDP-K	UDP-U
200B/Interest	2295.6[1106.42]	6.4[0.52]	37.4[8.21]
4096B/Data	2135.4[876.04]	4.0[0.82]	75.2[16.31]

Difficulties with NDN: Obtaining information from unknown publishers

NDN is essentially built as a query-response platform where subscribers are required to know the publishers or the precise identity of the content (URL) to send an interest. Since we believe it is important for subscribers to not know who are the publishers of a particular information item (i.e., a certain CD), NDN poses difficulties in achieving a true pub/sub environment. Consider a 2-publisher, 1-subscriber scenario. *Sub* sends a query "/query/sports/football" because (s)he is interested in football. On receiving the query, if Pub_1 and Pub_2 happen to have different new items of information, they will return their items to R. But only the response that arrives first will be returned by R since it will consume the PIT entry in R. If *Sub* wants the second update, he/she would either have to know the exact content name of Pub_2's response, or find a means to exclude Pub_1's response. Obviously, the first option is infeasible, since it is undesirable to negotiate a global name space across all the (possibly unknown) publishers. For the second option, the subscriber will have to specify a large number of names in the exclude field, especially if there are a large number of publishers (one can imagine thousands of such publishers). Moreover, since *Sub* does not know of the number of available publishers, it needs to send the Interest repeatedly till it stops receiving data from the various publishers and will have to repeat this process periodically. This model thus becomes similar to polling, with its associated overhead (as we will show in §5).

3.4 NDN Performance: Micro Benchmarking

We performed measurements to study the processing overhead of CCN compared to a pure IP-based forwarding (albeit recognizing that the functionality offered by a CCN node is significantly different). We ran simple benchmarks on the open source CCNx implementation[2] and standard IP based forwarding. The measurements were performed on Linux 2.6.31.9 machine (3.0 GHz Intel® E8400, 4GB Memory). Note CCNx is currently implemented as a user-space overlay, using UDP or TCP encapsulation for exchanging CCNx protocol packets between different nodes. To have a reasonably fair comparison, we use two kinds of packets in our measurements: 200-byte UDP packet compared to an NDN Interest packet, and a 4096-byte UDP packet to compare with an NDN Data packet (both with the same UDP payload size). The time between the incoming and outgoing instants of each packet is measured using Wireshark[3]. Three forwarding behaviors are measured: NDN, kernel-space UDP (UDP-K), and user-space UDP (UDP-U).

From Table 1, we observe that to achieve the functionality of name-based routing, NDN routers are about 500 times slower than UDP-K and about 50 times slower than UDP-

[2]http://www.ccnx.org/
[3]http://www.wireshark.org

U. Though a hardware-level implementation would be able to achieve better performance, the requirements placed on an NDN router is higher than that placed on IP router. Since COPSS supports NDN functionality, we understand the need to minimize overhead unless required. Further, we also explore the possibility of a hybrid COPSS + IP approach where the COPSS aware nodes at the edge support the content-centric pub/sub functionality while the intermediate nodes are just IP routers seeking to preserve forwarding efficiency.

4. COPSS ARCHITECTURE

COPSS is designed to be a content centric pub/sub platform that meets the requirements listed earlier - efficient, scalable, exploiting a push-based delivery using multicast for timely but efficient delivery, allow publishers and subscribers to be unaware of each other's identity, and support hierarchies in categorizing information. COPSS leverages the benefits provided by NDN for efficient content delivery and enhances it to provide a full fledged pub-sub platform. Publishers 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. COPSS naturally deals with substantial churn in subscription state, allowing a large number of users to join and leave and frequently change their subscriptions. The topics may change frequently as well (*e.g.*, in a Twitter-like publishing environment, where the popular topics change frequently).

4.1 COPSS Overview

COPSS introduces a push-based delivery mechanism using multicast in a content centric framework. At the content centric forwarding layer, COPSS uses a multiple-sender, multiple-receiver multicast capability, in much the same manner as PIM-SM, with the use of Rendezvous Points (RP). Users subscribe to content based on CDs. Subscriptions result in 'joins' to the different CDs that are part of the subscription. Each CD is associated with an RP, and the CCN may have a set of RPs to avoid traffic concentration. Although not necessary, the number of RPs may potentially be as large as the number of CCN nodes. Publishers package the CDs related to the information being published by them. As with PIM-SM, the published information is forwarded along the COPSS multicast tree towards the RPs, taking advantage of short-cuts towards subscribers where appropriate. COPSS makes use of hierarchical and context based CDs to aggregate content. COPSS aware routers are equipped with a *Subscription Table (ST)* that maintains CD-based subscription information downstream of them in a distributed, aggregated manner, as in IP multicast. An incoming publication is forwarded on an (inter)face if there are subscribers downstream for any one of the CDs in that publication. However, only one copy is forwarded on a given link (to gain the same advantage as multicast). This also ensures that subscribers subscribed to various groups do not receive duplicate content, to the extent possible. We note that our use of Bloom filters (described below) may result in false positives that would have to be filtered out at the first hop router next to the subscriber.

Additional capabilities in COPSS include the means to perform a two-step data dissemination capability to provide control of policy and access at publishers and to manage delivery of large volume data. Publishers send snippets of the content (which includes the CDs) and subscribers interested in the content query for it. Additionally, the COPSS architecture supports an efficient delivery mechanism for subscribers who were offline when the data was published. Agents/servers are responsible for also storing published content. Thus, subscribers who were offline could seamlessly query the network with the ID of the last received data and the COPSS aware network delivers content from such an agent/server. It also allows new subscribers to receive previously published information of interest.

In order to support pub/sub operation, COPSS introduces two additional types of packets, namely *Subscribe* and *Publish*, and are used in much the same manner as NDN's existing CCN packets *Interest* and *Data* being used for query/response interactions. By issuing a **Subscribe** for a CD, a COPSS subscriber will then receive updates when publishers **Publish** new content. We have attempted to make COPSS backward compatible with NDN as much as possible (e.g., taking advantage of caching in the CCN aware routers etc.). We now address each of the main aspects of COPSS in detail.

4.2 CDs: Hierarchical and Context Based Names

A CD can be any legal Content Name. For efficiency though, we exploit hierarchies in the structure of CDs. For example, a name $/CD/CD1/CD2$ includes CD, $CD1$ and $CD2$ as part of a hierarchy and could have multiple levels. It facilitates COPSS aware routers to aggregate subscription information and avoiding the forwarding of duplicate content. An example name may be $/sports/football/Germany$, where the CDs are $/sports$, $/sports/football$ and $/sports/football/Germany$. A subscription therefore can be at different granularities, taking advantage of this hierarchy.

4.3 Basic One-Step Communication

The basic one-step communication in COPSS is used for information dissemination via a push-based delivery using multicast. This is suitable for a 'pure' pub/sub environment, i.e., sending information where there is no need for policy control or for small volume content (e.g., Twitter-like short messages). The key operations of COPSS in this one-step communication model are *Publish* and *Subscribe*.

4.3.1 Publish using Rendezvous Nodes

COPSS supports sparse mode multicast at the content layer. This is done by introducing a rendezvous node (RN) as the root of a CD's subscription tree. As with an RP in PIM-SM, the RN receives content associated with a CD from a multitude of publishers and forwards it to all subscribers of the CD. The RN is a logical entity and handles information for more than one CD (possibly load-balanced across RNs using a policy for allocation of CDs to RNs) and resides on a physical COPSS aware router. a) An RN needs to be reachable from all publishers sending *Publish* packets with header prefix $/rendezvous/$ b) RNs receive packets from the publishers, strip the prefix $/rendezvous/$ and forward it to the interest subscribers.

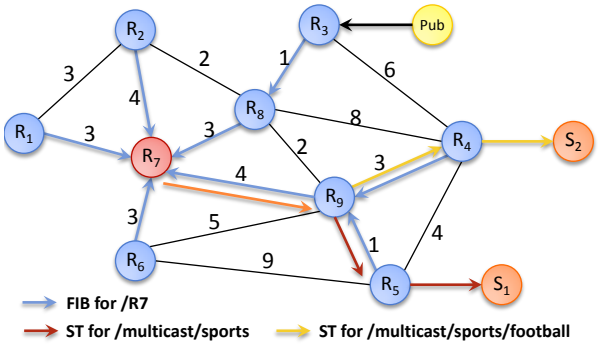


Figure 1: Multicast in COPSS aware network

Note that a publisher does not expect a reply to a *Publish* packet and sets the timeout value to 0. Therefore COPSS aware routers only forward it, rather than putting it into the Pending Interest Table (PIT). See §4.3.3 for a detailed example. The job of a RN is to receive and forward and can therefore be easily replaced whenever necessary.

4.3.2 Subscribe

A subscription received in a *Subscribe* packet (which may include one or more CDs) is treated just like a join in IP multicast. Subscription state is retained in a COPSS aware router, and the *Subscribe* is forwarded (if needed) towards the RN.

Forwarding (via Subscription table)
COPSS uses NDN's forwarding engine with an additional *Subscription Table (ST)* (see Fig. 2). The subscription table is used to maintain a list of downstream subscribers and the outgoing faces towards the subscribers. The *Subscribe* packet is forwarded towards the RN based on the forwarding table entry at a COPSS aware router. Along the path, the COPSS aware routers add this entry in their *ST*. If the COPSS aware router has an entry in the *ST* (the equivalent of being an on-tree node in IP multicast), the *Subscribe* information is aggregated and the incoming interfaces are included in the list of subscribers downstream. If not, the *Subscribe* is also forwarded towards the RN. *Data* messages with CDs matching the entries in the *ST* are forwarded on the list of interfaces that have subscribers downstream. The *ST* can be implemented as a <face:bloomfilter> [24] set. Every incoming data packet is tested against the bloomfilters and will be sent to the faces whose bloomfilter contains CD(s) the packet satisfies. This prevents multiple copies of the same data being sent out on the same interface. A COPSS user needs to subscribe to (i.e., declare an interest in) a CD. The subscription of a *user* to a certain *CD* is performed by creating an ST entry {Prefix="/*CD*"; Face=*user*} in COPSS. This allows a publisher's *Publish* packets with the corresponding CD to reach subscribers.

4.3.3 Overall Example

Rendezvous node set up: Assume that *R*7 in Fig. 1 is the rendezvous node assigned to handle the CD groups */sports/* and */sports/football*. *R*7 is assigned the name */rendezvous* and the COPSS aware network propagates this

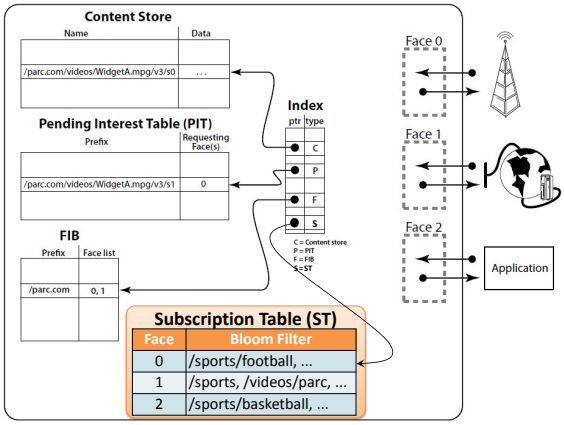


Figure 2: NDN's Forwarding engine adapted with subscription table

information and updates the forwarding tables (FIB) in the COPSS aware routers.

Subscription set up: Lets assume that *S*1 and *S*2, in Fig. 1, are subscribers interested in */sports/football* and */sports* respectively. Therefore *S*1 and *S*2 will forward a subscribe packet towards *R*7. COPSS aware router *R*9 along the path would store both the subscriptions in its *ST* (see Fig. 2) and forward the *Subscribe* packets towards the rendezvous node *R*7. *Subscribe* messages are similar to *Interest* packets in NDN, and can be considered 'standing queries' and uses the FIB for fowarding towards the RN and also updates the *ST*.

Content Delivery: A publisher just sends the content using *Publish* packets (the *Publish* packet is semantically similar to a *Data* packet, carrying data). However, it will be forwarded by looking up the forwarding table (FIB) unlike the PIT in NDN) with the header */rendezvous/sports* and/or */rendezvous/sports/football*. The COPSS aware network will first deliver it to *R*7, the RN. *R*7 on receiving this data will strip the header */rendezvous* and disseminate the *Publish* packet to subscribers downstream. Based on the *ST*, *R*7 realizes that there are subscribers for */sports* as well as */sports/football* and forwards this packet downstream. Intermediate router *R*9, then duplicates the packet and forwards it to *S*1 and *S*2. Hence, the reduction in bandwidth consumption and router processing overhead is achieved at *R*7.

4.4 Two step communication

With the basic push based communication model using multicast, COPSS uses *Publish* packets to carry the published content. However, to provide the flexibility for publishers to exercise policy and access control, as well as to efficiently distribute large volume content, we propose a two-step data dissemination method. Publishers first distribute snippets or a portion of the content (e.g., preview) as part of the payload. Subscribers then explicitly query for the content (an NDN *Interest*). This two-step model avoids subscribers from being overwhelmed with large content that may not be of interest (and saves network bandwidth).

A **snippet** is a payload carried in the *Announcement* instead of the actual content. It contains the meta information (the CDs), message abstract(s), pricing information and anything else that helps a subscriber decide if he would like to have the whole content. Additionally the snippet consists of the *contentName* that would help the subscriber obtain the data from the 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* can belong to multiple CDs, e.g., a news article about "An injury to an American football player" could belong to a CD for */news/america* 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. But the *contentName* carried in the snippet would be the same allowing for the possibility of PIT hits (if requests arrive at nearly the same time) and content store hits (if content is available in cache), when requests arrive from the subscribers of the two groups. Below, we detail our two-step(see Fig. 3) communication design.

Publisher: Announce
When a publisher has new content to publish, he multicasts an *Announcement* with the payload carrying a snippet of the data. The *Announcement* is implemented by sending a *Publish* packet, with the format for the name being "*/rendezvous/CD/*". Rendezvous nodes (RNs) do not differentiate between a normal *Publish* packet and an *Announcement* in their processing (i.e., eliminating the prefix "*/rendezvous*" and then forwarding the packet). When a subscriber receives multiple *Announcements* pointing to a same piece of data (identified by the same *contentName* portion), he would only need to query for that data once.

Publisher: Register
With two-step communication, a subscriber generates a query in response to a snippet, if he is interested in the data. For the query to reach the publisher(s) that send(s) the *Announcement*, publisher(s) must first propagate to the network the name prefixes (e.g., *contentName*) of all the content to be published (similar to the propagation of the content sources in NDN). This is called *Register* with the network. For example, a publisher who issues an *Announcement* of "*/rendezvous/CD/*" is expected to propagate the name prefix of "*/contentName*" (in essence the name of the content) in order to make the network aware of the availability of that content. 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).

Subscriber: Retrieve Data
On receiving an *Announcement*, the subscriber sends a query using an NDN *Interest* packet whose content name is the *contentName* portion in the snippet received in the payload. The publisher responds with the *Data* associated with the query. This mechanism benefits from NDN's communication paradigm and has the advantages of a potential cache hit on the way (which reduces the access latency of this content) and potential PIT hit (which reduces the query traffic) if

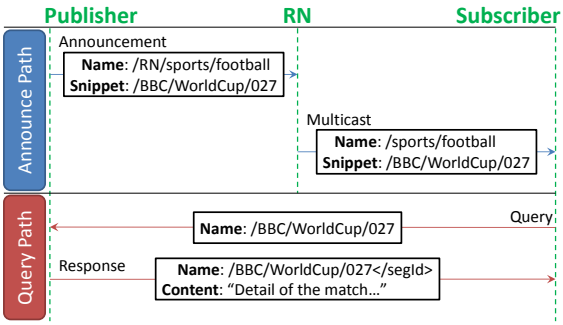


Figure 3: **Platform overview of the one-step and two-step communication.**

the same data has been requested by some other subscriber through the same router.

4.5 Supporting Asynchronous Data Dissemination: Subscribers Going Offline

A true pub/sub environment needs to be able to support 'asynchronous' data dissemination. By this we mean that when a subscriber goes offline (turns off the end-system or moves to a different location), the pub/sub environment will still enable the user to receive messages that were missed while being offline. Furthermore, the user should not have to know who the publishers were, or even whether they are still connected to the network. For instance, the original publisher may no longer be online (e.g., transmitted a warning before being disabled). COPSS supports this by having a dedicated *broker/server* that acts as a store for all COPSS multicast messages. It is likely that a very large amount of information would have to be stored at the broker, which poses scalability challenges. Our solution to this is the natural one: allow the logical broker to be a set of collaborating, distributed servers (i.e., a *broker cloud*). Load is shared among them and they provide some level of redundancy. But most importantly, such a design offers the desired scalability. During COPSS's bootstrapping, the FIB information with a pointer to the broker cloud is propagated to the whole network – just like the FIB pointing to rendezvous nodes is propagated – so that the broker cloud is reachable from everywhere in the network.

The broker subscribes to any newly created CD based on *Announcement* and *Publish* messages received from publishers. Thus, it obtains 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.

Querying for missed messages
In order to query for missed messages, COPSS requires a subscriber to remember the content name of the final message he received when he was last online. We also require that the broker cloud order all received messages based on their arrival using its local time. When a subscriber goes offline and later rejoins the system, he queries the broker cloud

with two pieces of information: the group he subscribed to and the message he received last (by sending a *Subscribe* packet with the header */broker/CD*). The COPSS aware routers forward the *Subscribe* based on the header */broker/* to the broker cloud. The broker cloud has to look up in the log to find the match to the message from the subscriber. For each multicast message received after this, the broker cloud checks whether it belongs to any of the CDs provided by the *Subscribe*. The broker then sends all the matched messages to the subscriber. Below, we address scalability and reliability issues:

Scalability: Retrieving missed content
A subscriber may have subscribed to a very large number of CDs. Some of the CDs subscribed to could be related to infrequent events such as a "disaster warning", or popular, frequently updated information such as sports. When dealing with asynchrony, a subscriber coming back online would have to send a query for every CD that he has subscribed, 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, COPSS seeks to aggregate processing by grouping content across multiple CDs. Instead of querying for content related to individual CDs, the subscriber queries for the group. The tradeoff is that the subscriber might receive false positives, which have to be eliminated at the receiving end-point. Aggregation functions include traditional hashing schemes (e.g., based on the CD string), but have the disadvantage that their decision does not take the semantics related to CDs into consideration. COPSS's use of the hierarchical structure of CDs allows subscribers and/or brokers to exploit it for aggregation. We believe that this will help to minimize the retrieval overhead and in reducing false positives. For example, a subscriber who is subscribed to a large number of disaster warning related CDs could aggregate them to a higher level CD such as */disaster/* and issue one query to the broker. It is particularly attractive when these updates are infrequent and the number of false positives are small.

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. But this also provides opportunities to optimize network traffic. We propose using markers in the message sequence to allow for batching responses. E.g., assume a subscriber of a CD requests for content that he missed since 9 pm and another subscriber of the same CD requests content that he missed since 11 pm. Using a marker to delineate the message sequence into batches allows the broker to multicast the overlapping message sequence between the subscribers. This reduces both network and broker load.

Reliability: Possible loss of sequence
COPSS multicast messages may arrive at the subscribers and the broker cloud in different order. This can be caused e.g., by varying 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. We solve this problem 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 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 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 probability of recovering all the messages when the subscriber is offline, depending on the message frequency and user online/offline pattern.

4.6 A Hybrid Model for Incremental Deployment of CCN Capability

We have so far considered an ideal case where all routers are COPSS aware. However, our micro benchmarking of the forwarding performance of CCN routers (see §3.4) indicated that the processing a CCN packet in a CCN router can be substantially more expensive than forwarding a packet at a normal IP router. Given that forwarding a *Data* packet involves examination of the CCN headers (to parse the CDs, examine if it is in the cache etc.), it is desirable that these functions be performed on when essential.

Therefore, we develop a practical yet effective solution for a hybrid environment (COPSS + IP): COPSS aware routers are present only at the edge of the network and connected to the native IP network. We provide the necessary COPSS functionality at the edge while still achieving efficiency of forwarding packets in the core. Parsing of content centric names and forwarding decisions are made by the COPSS aware (edge) routers, while leveraging the high performance of IP forwarding in the core IP network. Such an architecture naturally allows subscribers to subscribe to CDs and allows publishers to publish data based on CDs. Furthermore, the deployment cost may be reduced by only replacing edge routers with COPSS aware routers.

To facilitate the push-based multicast in COPSS, we make use of native IP multicast capabilities in the core of the network and perform the mapping from CDs to IP multicast group addresses. This allows the network to maintain the advantages provided by the COPSS for both publishers and subscribers such as maintenance of the subscription list, CD-based subscription and publishing, one-step and two-step dissemination and support of users going offline but still allowing asynchronous communication between publishers and subscribers. The downside is given that the limited availability of IP multicast addresses compared to the envisioned scale of the number of CDs in a COPSS aware edge router, multiple CDs may be mapped to a IP multicast address.

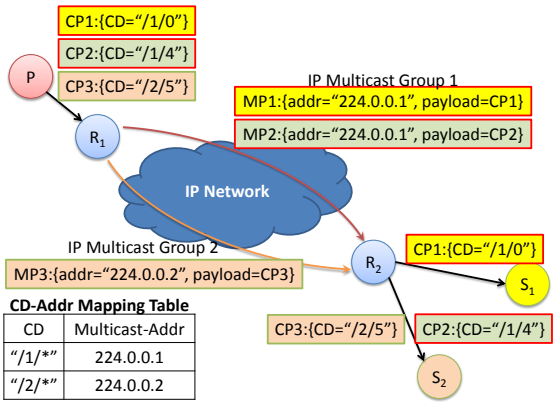


Figure 4: **Multicast in COPSS + IP.**

Such a mapping may resulting in messages being unnecessarily delivered to subscriber end-nodes that then have to discard irrelevant messages.

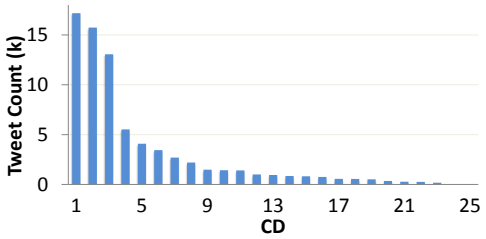
An example: In Fig. 4, we give a brief example about how hybrid-COPSS works. S_1 (subscribed to "/1/0") and S_2 (subscribed to "/1/4" & "/2/5") are connected to R_2, which is a COPSS-aware edge router. On receiving the subscription request, R_2 translates these CDs to IP multicast groups according to the mapping table. As a result, it joins groups 224.0.0.1 and 224.0.0.2 in the IP network. When a publisher P (connected to R_1) multicasts a COPSS packet, R_1 will encapsulate the packet according to the CDs it contains. In this example, $CP1$ is encapsulated as a UDP/IP packet using the IP multicast address 224.0.0.1 based on the mapping table. When R_2 receives the packets, it disseminates the packets to the end hosts according to the subscription table, ST as described above. This way, S_1 then receives $CP1$ and S_2 receives $CP2$ and $CP3$. However, according to the mapping table, R_2 can also receive packets with CD "/2/3". Since there is no subscriber downstream, R_2 will discard this packet. This results in some wasted network traffic being transmitted on some links. This is the tradeoff because of the aggregation of CDs.

5. EXPERIMENTAL EVALUATION

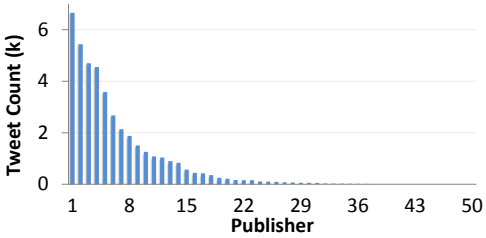
We use simulations to evaluate the benefit of COPSS. We show how pub/sub capability of COPSS achieves improved performance compared to a pull-based NDN implementation as well as pure IP multicast. We use the results of our micro-benchmarking measurements to parameterize the simulations. We built an event-driven simulator in $C\#$. To drive our simulator, we use a set of traces collected from Twitter. We evaluate the performance of COPSS in multiple dimensions: forwarding performance with both one-step and two-step (delivering a snippet and allowing subscribers to query for content) communications, subscribers going offline and then retrieving messages missed, as well as the benefits of a hybrid model.

5.1 Experimental Setup

Dataset: We use a Twitter data trace of tweets on technical topics obtained from the public Internet during a one-week



(a) # of tweets per CD



(b) # of tweets per pub

Figure 5: **Dataset information**

period in 2010, which totaled 68,695 tweets sent by 38,481 users. We identified and selected 25 hot keywords such as *iphone*, *ipad*, *blackberry*, *smartphone* as CDs. We filter this data trace and retrieve a subset in which every tweet contains at least one of the 25 keywords. This subset amounts to 41,613 tweets from 22,987 users (4.13 tweets/minute, up to 48 tweets posted at the same time) and is used as our simulation input. We then filtered additional hot keywords from these 41,613 messages in order to obtain sub-CDs for the selected 25 CDs. The sub-CDs range from 1 to 25 for the selected 25 keywords. By this method, we have a total of 407 distinct CDs that can be hierarchically grouped into 25 CDs. Fig. 5a shows the number of tweets associated with each CD respectively. To make the publishers of our system tweet more frequently, we assign the 22,987 users to 50 publishers by hashing them based on a power-law function. Fig. 5b shows the number of tweets per publisher. Without the means to inferring CD-Subscriber relationship from the data trace, we assume the more popular the CD is, the more subscribers there will be (based on [25]). Thus, we distributed the number of subscribers per-CD based on the CD-Tweet relationship. Subscribers can subscribe to multiple CDs, but a subscriber who is subscribed to a top level CD will not be subscribed to a lower level CD belonging to it. To simplify the simulation, subscribers will query the data as soon as they get announcements.

Network topology: We use the Rocketfuel [26] backbone topology (id=3967) for the core routers. Besides, we put 200 edge routers on the 79 core routers, with each core router having 1-3 edge router(s). We 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 5 ms; the delay between each the host and its associated edge router is

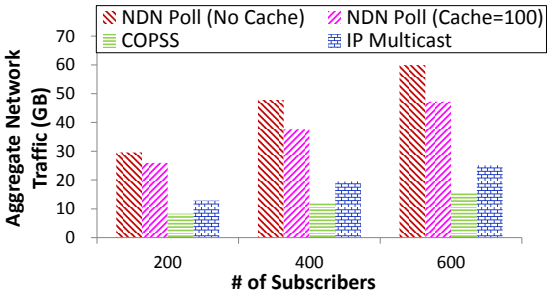


Figure 6: Aggregate Network load (NDN (cache=0, cache=100) vs. COPSS one-step and IP Multicast)

set to 10 ms.

Aggregate network load: We study the impact of COPSS on the network by measuring the *aggregate network load* calculated by:

$$\sum_{i=1}^{packetCount} packetSize_i \times hopCount_i, \qquad (1)$$

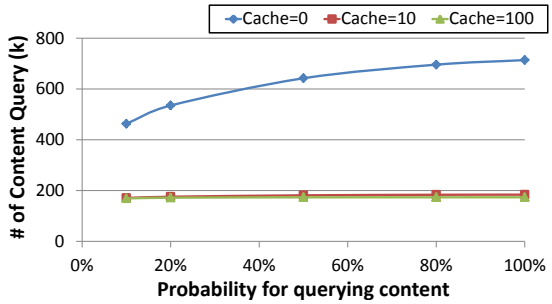
i.e. when a packet with size 1kB is sent from host A through router R to host B, 2kB is added to the aggregate network load. For a fair comparison, COPSS packets are encapsulated into UDP packets when transmitted over an IP underlay. The encapsulation overhead is therefore the same as in a CCNx implementation.

5.2 Performance of COPSS's Basic Communication Model, NDN and IP Multicast

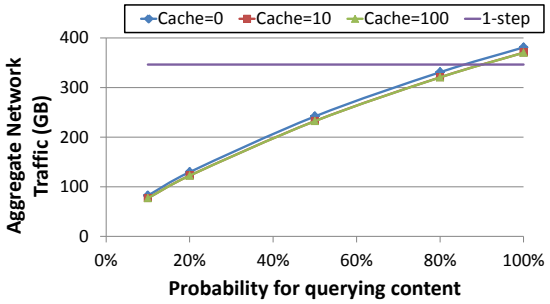
Fig. 6 illustrates the aggregate network load driven by our emulated Twitter-like application over the standard pull-based NDN (both without a cache as well as a cache of 100 packets), using COPSS's one-step communication model as well as native IP multicast. We observe that the COPSS has a lot less load in this case, because of its design of a content centric push mechanism that improves upon the NDN proposal by adding a multicast capability. In addition: 1) Polling is not used by COPSS unlike NDN. With NDN, subscribers need to poll periodically (every 30 minutes in the default setting) introducing additonal overhead. 2) Network traffic is further reduced because of multicast with COPSS even compared to the use of caches in NDNs. Moreover, COPSS performs no worse than native IP multicast, because of the use of the hierarchy based grouping of 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).

5.3 Performance of COPSS (Two-Step)

Here, we evaluate the performance of the COPSS to transfer large volume content using its two-step communication model. Large files that are 128 times larger than the original Twitter messages are created from the dataset. We



(a) Publisher Load



(b) Aggregate Network Traffic

Figure 7: **COPSS in two-step mode**

Table 2: **COPSS + IP performance vs COPSS-everywhere and IP-multicast**

	IP-Multicast	COPSS-everywhere	COPSS + IP-Multicast
Content Dissemination Latency (ms)	78.76	82.73	77.62
Aggregate Network Traffic (GB)	19.50	12.32	13.15

study the load on the publisher (see Fig.7a) by calculating 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.We observe that a cache size of 10 packets is sufficient to reduce the load on the publisher significantly.

Fig. 7b illustrates the aggregate network traffic when a varying number of subscribers request for the full content on receiving the snippet. For reference, we have also shown the case of COPSS in one-step mode delivering the full content instead of sending a snippet. 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 sending of snippets in the first step is just additional overhead compared to the one-step delivery mode.

Table 3: **Broker load vs. router cache size**

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

5.4 Performance of Hybrid-COPSS (COPSS at Edge + IP at Core)

In §3.4, we observed that content oriented processing is more expensive than IP. The aim of hybrid-COPSS (COPSS at edge + IP in the core) is to achieve the best of both worlds by obtaining the functionality of content centricity while retaining the forwarding efficiency of IP. To validate this, we evaluate the performance of hybrid-COPSS in one-step mode (COPSS aware routers at edges and IP routers at core) to that of COPSS-everywhere in one-step mode (all routers are COPSS enabled) as well as native IP-multicast. Table 2 shows the content dissemination latency which is dominated by the processing overhead incurred at every router (the processing overhead was obtained through measurements 1). The latency incurred is high in the case of COPSS-everywhere since every COPSS aware router will have to process the *Publish* and *Subscribe* packet. Comparatively, routers enabled with IP multicast perform much better in terms of latency. Hybrid COPSS is able to achieve lower latency than both COPSS-everywhere and IP-multicast. This is due to the fact that even though the edge COPSS aware routers incur high processing overhead, they are able to send a single copy to multiple groups and are thus able to send much lower traffic into the network. IP-multicast on the other hand needs to send the same message multiple times to be able to reach subscribers belonging to different CDs. This can be clearly seen in the row titled "Aggregate Network Traffic" where we observe that IP-multicast generates higher network traffic. The aggregate network traffic of hybrid-COPSS is slightly higher than COPSS-everywhere due to the fact that CDs are mapped to IP multicast groups and therefore results in a small amount of false positive content being delivered at the last hop COPSS enabled router close to the subscriber.

5.5 Performance of COPSS for Offline Users

Based on our preliminary analysis on the twitter data trace available, we synthesize the users' offline/online pattern as follows: everyday about 75% of all subscribers randomly go offline between 2am and 3am, and then go back online randomly between 10am and 11am. For each of the remaining users, we randomly choose two time points as the start and end points of their offline period with an average offline duration of 20 minutes. Though this does not match a real world scenario, we created such a behavior to study the impact of a large portion of the subscribers coming online at nearly the same time. In Table 3, we observe that 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.

5.6 Additional Observations

Table 4 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 RN based multicast, the publishers need not propagate their entry and only the RN propagates the entry to all the (278) COPSS enabled routers. This 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.

Table 4: **The FIB entry created due to Server/RN and publishers of the specific content**

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

6. SUMMARY

In this paper we presented and evaluated COPSS, an efficient pub/sub-based content delivery system exploiting the fundamental capability of CCN for efficient information dissemination. COPSS builds on existing proposals for CCN/NDN to provide pub/sub functionality. COPSS is designed to be scalable to a large number of publishers, subscribers and CDs. We recognize that a pub/sub platform needs to be able to accommodate users going offline, and allow them to retrieve content that has been published during the time the subscriber was offline. We also explicitly address the need for incremental deployment of CCN capability in traditional IP networks. We use trace-driven simulations to evaluate the COPSS architecture. COPSS reduces the aggregate network load and the publisher load significantly. The additional CCN 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. COPSS is substantially more efficient than existing pull-based CCN proposals (such as NDN) because of its inherent pub/sub capability.

7. REFERENCES

[1] K.V. Katsaros, G. Xylomenos, and G.C. Polyzos. Multicache: an overlay architecture for information-centric networking. *Elsevier, Computer Networks*, 55(4):936–947, March 2011.

[2] Venugopalan Ramasubramanian, Ryan Peterson, and Emin Gün Sirer. Corona: a high performance publish-subscribe system for the world wide web. In *NSDI*, 2006.

[3] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J.D. Thornton. Named data networking (ndn) project. Tech. report ndn-0001, PARC, 2010.

[4] 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 *SIGCOMM*, 2007.

[5] S. Arianfar, P. Nikander, and J. Ott. On content-centric router design and implications. In *ReArch*, 2010.

[6] 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 *ReArch*, 2008.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F.

Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT*, 2009.

[8] W. Fenner, D. Srivastava, K. K. Ramakrishnan, D. Srivastava, and Y. Zhang. Xtreenet: Scalable overlay networks for xml content dissemination and querying. In *WCW*, 2005.

[9] M. Ott, L. French, R. Mago, and D. Makwana. Xml-based semantic multicast routing: an overlay network architecture for future information services. In *GLOBECOM*, 2004.

[10] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *DEBS*, 2007.

[11] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content based routing with elvin. In *AUUG2K*, 2000.

[12] C. Esteve, F. Verdi, and M. Magalhaes. Towards a new generation of information-oriented internetworking architectures. In *ReArch*, 2008.

[13] 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.

[14] Yanlei Diao, Shariq Rizvi, and Michael J. Franklin. Towards an internet-scale xml dissemination service. In *VLDB*, 2004.

[15] Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. Tera: topic-based event routing for peer-to-peer architectures. In *DEBS*, 2007.

[16] Spyros Voulgaris, Etienne Riviére, Anne-Marie Kermarrec, and Maarten Van Steen. Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. Research report, INRIA, December 2005.

[17] Xep-0060: Publish-subscribe. Xep-0060 (standard track, v 1.13, XMPP Standards Foundation, http://xmpp.org/extensions/xep-0060.html, 2010.

[18] S. Deering. Host extensions for ip multicasting. RFC 1112, August 1989.

[19] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2362 (Experimental), June 1998.

[20] T. Pusateri. Protocol Independent Multicast - Sparse Mode (PIM-SM) IETF Proposed Standard Requirements Analysis. RFC 4602 (Informational), August 2006.

[21] Y.-H. Chu, S.-G. Rao, and H. Zhang. A case for end system multicast. In *SIGMETRICS*, 2000.

[22] S. Banerjee, B. Bhattacharjee, and C. Kommareddyand. Scalable application layer multicast. In *SIGCOMM*, 2002.

[23] J. Jannotti, D.-K. Gifford, and K.-L. Johnsonand. Overcast: Reliable multicasting with an overlay network. In *USENIX OSDI*, 2000.

[24] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[25] H. Kwak, C.Lee, H.Park, and S.Moon. What is twitter, a social network or a news media? In *WWW*, 2010.

[26] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *IMW*, 2002.

3 Hybrid Content Oriented Publish Subscribe System

Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP

Jiachen Chen*, Mayutan Arumaithurai*†, Xiaoming Fu*, K.K. Ramakrishnan‡
* Institute of Computer Science, University of Goettingen, Germany,
Email: {jiachen, arumaithurai, fu}@cs.uni-goettingen.de
† NEC Laboratories Europe, Heidelberg, Germany, Email: arumaithurai@neclab.eu
‡ AT&T Labs-Research, Florham Park, NJ, U.S.A., Email: kkrama@research.att.com

ABSTRACT

Content-Centric Networking (CCN) seeks to meet the *content-centric* needs of users. In this paper, we propose hybrid-COPSS, a hybrid content-centric architecture. We build on the previously proposed Content-Oriented Publish/Subscribe System (COPSS) to address incremental deployment of CCN and elegantly combine the functionality of content-centric networks with the efficiency of IP-based forwarding including IP multicast. Furthermore, we propose an approach for incremental deployment of caches in generic query/response CCN 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 CCN overlay. Our hybrid approach would also be applicable to the Named Data Networking framework.

To demonstrate the benefits of hybrid-COPSS, we use a multiplayer online gaming trace in our lab test-bed and microbenchmark the forwarding performance and queuing for both COPSS and hybrid-COPSS. A large scale trace-driven simulation (parameterized by the microbenchmark) on a representative ISP topology was used to evaluate the response latency and aggregate network load. Our 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.

Categories and Subject Descriptors

C.2.1 [**Computer-communication Networks**]: Network Architecture and Design

General Terms

Design, Performance

Keywords

CCN, NDN, COPSS, Pub/Sub, Multicast, Incremental Deployment

ANCS'12, October 29–30, 2012, Austin, Texas, USA.

1. INTRODUCTION

Content Centric Networking (CCN) [1–7] seeks to transform content as a first-class entity. Rather than the current *location-centric* network architecture, CCN presents a networking paradigm that enables users to issue a query for content instead of contacting a specific end host for that content. It allows the network to provide the content from any of the multiple sources where it is available (including the in-network cache), and meets the user's need to send and receive content, irrespective of where it is or who generated it. Named Data Networking (NDN) [4] is one of the more popular examples of CCN.

Publish/subscribe (pub/sub) systems are particularly suited for large scale information dissemination, and provide the flexibility for users to subscribe to information of interest, without being intimately tied to when that information is made available by publishers. This temporal separation between information generation and indication of interest is a highly desirable content centric feature. Our proposal, Content-Oriented Publish/Subscribe System (COPSS) [7,8] is built on top of NDN with a new CCN-oriented multicast capability, to meet the efficiency and scalability needs of large scale pub/sub systems. Examples we address include a content-focused Twitter-like pub/sub system [7] and applications with tight timeliness requirements such as online multiplayer gaming and content streaming [8].

A major component of the NDN design is the extensive use of caching at every hop of the network. Specifically, NDN requires every NDN router 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 discussed in [7], we demonstrated that supporting content centricity entails significant additional processing in the forwarding engine. Thus, the excitement of the new content centric network architecture has to be tempered by the performance, cost and complexity consequences of the architecture. Furthermore, there is an important aspect on the incremental deployment

and co-existence of CCN with the current IP-oriented network world.

In this paper, we examine how to evolve from an IP infrastructure to a CCN-oriented network by co-existing with the IP network. Hybrid-COPSS attempts to support all the *functionality* a COPSS-enhanced CCN network 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-like forwarding capability where appropriate. Cache hits from the key CCN nodes enable fast response to content requests, but this needs to be balanced against the cost of having a large number of complex CCN nodes. Therefore, additionally, by a judicious choice of placing a limited number of full-fledged CCN nodes that can also cache content at key points in combination with a larger number of hash-based forwarding (similar to IP forwarding), we address the problem of efficient migration to an Information Centric future. The NDN implementation treats CCN as an overlay using TCP/UDP between CCN overlay nodes. However, we believe a tightly integrated approach as proposed here provides 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. Additionally, we also propose a new FIB propagation mechanism and an incremental deployment strategy to increase the cache hit rate with the limited amount of memory (cache) size. Our contributions in this paper include:

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

- Addressing the challenges of inter-domain multicast, through hybrid-COPSS's use of CCN overlay nodes at individual administrative domain edges. Moreover our 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 CCN network may use, onto a limited IP multicast group address space. The choice of the address mapping has a direct impact on network efficiency. We provide 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 CCN deployment. We propose 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.

- Evaluating the performance of hybrid-COPSS against COPSS and IP multicast in an experimental test-bed. We use a gaming trace to microbenchmark the forwarding performance and queueing in our lab test-bed and use a representative ISP topology in a simulation environment to compare the response latency, network traffic and rendezvous point throughput of the alternate approaches. The results show that hybrid-COPSS can be integrated into the current IP network architecture, and significantly improves latency compared to a CCN-only environment. While network traffic can be higher as a result of an extremely limited IP multicast group address space, we observe that even with a larger address space, an IP multicast-like approach can only perform as good as hybrid-COPSS. However, hybrid-COPSS significantly outperforms IP-multicast-like approaches in multi-domain environments, in terms of reducing inter-domain traffic.

Our paper is organized as follows. In §2 we discuss related work and background. Then, we give the basic pub/sub communication of hybrid-COPSS in §3. In §4, we describe our optimization on query/response in hybrid-COPSS. §5 solves the inter-domain multicast problem and §6 discusses scalability and management issues. Evaluation results are reported in §7 before concluding in §8.

2. RELATED WORK

Publish/subscribe systems (*e.g.* [6, 9–14]) are attractive because they relieve the consumers from the strict synchronization in time and location when the information is generated by publishers [15]. Although meant to be content-centric in nature, current systems require the users to know the location or identity of the publisher of the information of interest. 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.

NDN [4] is an instantiation of CCN. With NDN, a human readable *ContentName* is used to identify a data chunk. Request and response are represented by two basic kinds of packets *Interest* and *Data*. Information consumers send Interests (queries) to request for content. NDN routers forward Interests towards potential information providers according to the Forwarding Information Base (FIB) (we use $FIB(name)$ to denote the outgoing face(s) in FIB for ContentName *name*). Interests not matching an entry in the Content Store (CS) are cached at the Pending Interest Table (PIT) of each router they traverse. Information providers subsequently send matched Data (response) packets along the reverse path of the pending Interests towards information consumers. Data packets will be cached at each router's Content Store for future use, and will consume each pending Interest when traversing the path. Proposals such as VoCCN [16, 17] try to support online live streaming and group communication over NDN architecture. However, such systems might incur significant overhead due to the inherent query/response communication model, when a simple pub/sub form of information dissemination is desired.

In [18], the authors motivate the need for channel-like (including push-based) communication in NDN. IP multicast solutions such as PIM-DM [19], PIM-SM [20] and SSM [21]

provide multicast group communication to applications like IPTV. However, a lot of ISPs do not support IP multicast, especially for inter-domain traffic, due to complexity and charging concerns. Overlay multicast [22–24] is another alternative for multicast group 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 multicast is agnostic to the underlying topology, likely resulting in forwarding inefficiencies.

COPSS [7, 8], created as an enhancement to NDN, provides an efficient pub/sub capability in the content-centric network architecture and effectively integrates pub/sub with query/response. COPSS supports content-centric features in the naming structure including the ability to include hierarchies. The fundamental component, a Content Descriptor (CD), is used in the CCN-based multicast framework. COPSS uses a Rendezvous Point (RP) based multicast structure, along with automatic RP balancing to avoid traffic concentration. However, the performance of COPSS may be improved in environments where incremental deployment or co-existence with IP networks is desired, by exploiting native IP multicast, to reduce latency. In this paper, we present a complete COPSS overlay based solution that is more efficient and scalable than the solution outlined in [7,8] that consisted of the 1st hop router simply mapping a Content Descriptor (CD) to an IP multicast group. Hybrid-COPSS facilitates the dynamic management of the CD to IP multicast and CD to RP mapping, which helps in reducing the unnecessary traffic being sent in the network. The solution we study in this paper also provides the means for a careful partitioning of the functionality at different nodes so as to allow the nodes to maintain the functionality (as we incrementally deploy) as much as possible when the network finally evolves into pure CCN/COPSS environment.

While we presented the basic philosphy for co-existence and graceful migration in a workshop paper [25], we present in this paper a detailed architecture that handles pub/sub, optimizes query/response with a 2-stage dissemination and RP balancing in a multi-domain scenario so that such RP balancing "affects only the first domain downstream". Furthermore, we provide a thorough evaluation to illustrate the cache hit rate and response latency with the optimized query/response solution. In this paper we look at the total update latency from microbenchmarking to illustrate that the use of an efficient IP forwarding, hybrid-CCN (hybrid-COPSS) can have better performance even in a simple topology. Our approach also offers increased scalability for the pub/sub topology with varying numbers of end-hosts.

3. Hybrid-COPSS PUB/SUB

In this section, we describe the multicast-based delivery model of hybrid-COPSS with our approach for incremental deployment, leveraging an IP network's efficient forwarding. We retain the content centric functionality from both the user's and the end-system's perspective. In [4, 6], the authors proposed that content centric network could be built as an overlay to achieve the CCN's 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, we explore an integrated approach. We allow hybrid-COPSS 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, we seek 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 we do not place strict requirements on where the pure COPSS or pure IP routers need to be placed. In fact, we expect that the COPSS enabled nodes can be used either with their full CCN 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 CCN capability of intermediate nodes when and where needed. The overlay-underlay design of the nodes implies that where needed, there are CCN routers that provide query aggregation and caches (thereby the benefit of cache hits).

3.1 Packet Forwarding in COPSS

To provide a flexible publish and subscribe functionality, COPSS adopts a Content Descriptor (CD) based approach where a CD could refer to a keyword, 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). However, in NDN, a piece of data is identified by a globally unique ContentName. But, in COPSS a piece of data (*e.g.*,a document) can have multiple CDs and at the same time there may exist multiple data items (*e.g.*, multiple documents) that are identified by a given CD.

COPSS is designed to use a Rendezvous Point (RP)-based multicast as the basic communication model. To reduce the (well-known) traffic concentration at an RP, several RPs are setup in the network based on the workload, and every RP serves a set of CDs. Considering the problems related to RP balancing and node failure, we do not constrain an RP to reside only at a given physical machine. It is a module identified by a ContentName that can be dynamically placed at a router. RN_{CD} is the ContentName that identifies the module implementing the RP serving the CD. CD to RP mapping is stored in the RP_Table on every edge router ($RN_{CD} = RP\_Table(CD)$). When an RP is setup (or moved), it will propagate the list of the CDs it is responsible for along with its own ContentName throughout the network. The COPSS routers would then have an FIB entry storing the ContentName and the outgoing face towards the RP. We use $FIB(RN_{CD})$ to denote the outgoing face towards the RP that serves CD.

COPSS routers are equipped with a Subscription Table (ST) to store the outgoing faces to reach subscribers downstream. The ST is a dictionary of `{Face:Bloom-Filter(CDs)}` such that if any CD in a Publish packet has a hit in the

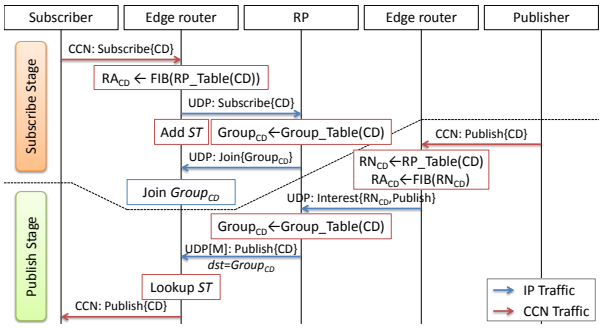


Figure 1: Basic Protocol Exchange

bloom filter for a face, the packet will be sent (and will only be sent once) through that face.

When an end host subscribes to a CD, it sends a Subscribe packet to its 1^{st} hop router. The 1^{st} hop router modifies its own ST and forwards it upstream, *i.e.*, $FIB(RN_{CD})$, after checking if it has not already subscribed to this CD. The upstream routers change their local ST and forward the Subscribe packet until it reaches the RP or a router that has already subscribed to the CD (essentially an 'on-tree' node). On receiving a Publish packet (containing multiple related CDs and the content) from an end host, a 1^{st} hop router encapsulates the Publish packet into an Interest packet with ContentName RN_{CD}. This packet will be forwarded to the RP that serves the CD. The RP recognizes the Interest packet and decapsulates it. The decapsulated Publish packet is forwarded downstream according to the routers' STs until it reaches all the subscribers.

3.2 Packet Forwarding in Hybrid-COPSS

In hybrid-COPSS, we seek 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 RP_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, we denote RA_{CD} as the address of the RP module that serves CD. The RA_{CD} can be calculated by $FIB(RN_{CD})$ because the RPs and the edge routers are directly linked at the CCN overlay layer. Then, this packet will be forwarded to the RP through the underlay. CD to 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 5.

On receiving a Subscribe packet from an end host that seeks to subscribe to a CD, the 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. 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).

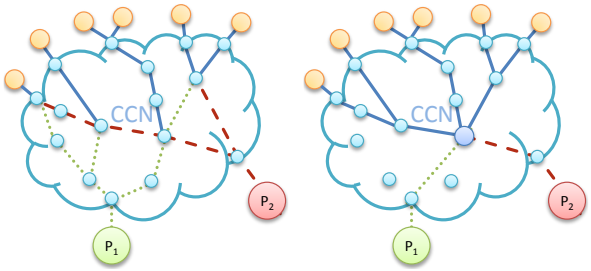
To publish content, the publisher sends a Publish packet to the edge router (the user behavior is unchanged). The protocol exchange is shown as the "Publish Stage" in Fig. 1. The edge router encapsulates the packet using an Interest with RN_{CD} and sends it to RA_{CD}. When the RP receives this packet, it will decapsulate it and forward it based on the Group_Table, instead of the ST. We can also use the ST on the RP 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 Publish 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 CCN 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.

We believe that to a significant extent ISPs support IP multicast within their domain. The primary challenge is in supporting IP multicast across domains. However, in those cases where IP multicast is not supported within a domain, we rely on a pure COPSS overlay. We then exploit an overlay multicast tree to minimize the number of copies sent from the RP and on each overlay hop.

4. Hybrid-COPSS QUERY/RESPONSE

Query/response based dissemination is an essential part of content delivery, in addition to the pub/sub delivery mechanisms. 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 2-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 2-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 he subscribed to. At the same time, this strategy can help 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 1^{st} stage, we use COPSS to minimize the announcement latency, and for the 2^{nd} stage, we use query/response to get the best use of in-network cache, as is typically performed in NDN. This work 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 we do not place strict requirements on where CCN-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 CCN aware in the basic case. The simplest approach is to enable query/response functionality on the COPSS nodes. But for achieving improved overall efficiency, we believe that having more CCN-aware routers can help because of the in-network cache and FIB aggregation



(a) Original FIB Status **(b)** New FIB Status

Figure 2: FIB propagation for Query/Response

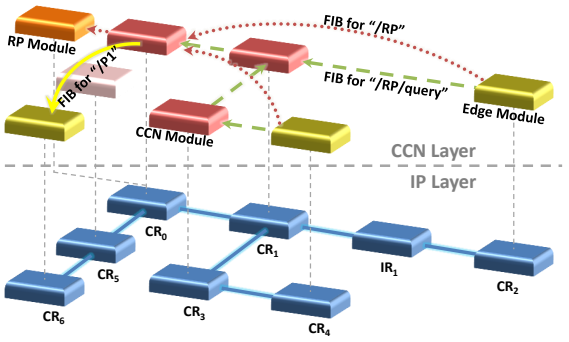


Figure 3: Overlay for Query/Response

capability. However, for the dissemination of the content, traversing a larger number of CCN-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, we explore optimization strategies for placement of CCN caching nodes versus nodes that forward based only on IP.

4.1 RP-based Query/Response

To provide content-oriented query/response in an incremental way, we propose an optimized RP-based query/response architecture that allows for reducing the size of the FIB while increasing the cache hit rate. To allow the subscribers to obtain the content, the publishers need to send a globally unique ContentName along with the snippet and also propagate that he serves (the prefix of) the ContentName beforehand. The propagation results in an FIB entry being added either at the RP or in the CCN aware routers depending on the approach (see Fig. 2). In COPSS, the globally unique ContentName is divided into 2 parts: globally unique publisher name and the publisher's locally unique data ID. In the case of the NDN-based query/response, a globally unique name is used. NDN requires the network to know how to reach the publisher by creating a source-based tree rooted (*i.e.*, starting from the edge of the network inwards) at the publisher's 1^{st} hop router. Fig. 2a shows the FIB for `/P1` (green dotted lines) and `/P2` (red dashed lines). The blue solid lines are the shared part of the trees. Since there might be subscribers everywhere in the network, every router needs to know the outgoing face for all the publishers. The total FIB entry size in the network can be calculated as:

$$Size_{FIB} = (n_{rp} + n_{pub}) \times n_r, \tag{1}$$

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

However, in a pub/sub system environment, the RPs are already well-known to the complete network (for multicast). If we aggregate the query/response tree at the RP, we can reduce the FIB entries stored on every router: only RPs need to know how to reach the publishers. In Fig. 2b, the blue solid lines from the subscribers to the RP are the FIB for `/RP`. In the hybrid world, FIB values are `{IP:port}` pairs, so we can point $FIB(P1)$ on RP directly to the 1^{st} router of P_1. The green dotted line in Fig. 2b is an overlay link that skipped an intermediate router in the underlay. The FIB entry size in the network can be calculated as:

$$Size'_{FIB} = n_{rp} \times n_r + n_{pub} \times n_{rp}. \tag{2}$$

Since $n_{rp} \ll n_r$, the new FIB size can be much smaller than the original (in equation 1). Though 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, RPs would be responsible for aggregating FIBs from data providers and would advertise themselves. The CCN routers would forward the query to the RPs, which in turn would forward it to one of the data providers in case a cache hit has not already occurred. The benefits of such an architecture for the NDN-like query response is that it limits the FIB sizes that are being propagated in the network. In the rest of the section, we detail the effect of placing CCN aware nodes on the pub/sub model.

FIB Propagation from Subscribers

In §3, we required the RP and all the edge routers be CCN-aware routers. In the overlay, we created FIB entries from the edge routers directly to the RP since there are no CCN-aware routers in between. When we have more CCN-aware routers deployed in the network to increase the cache-hit rate for query/response interactions, we slightly modify the "Subscribe Stage" to allow intermediate CCN-aware routers be part of query/response tree as well. But, in the "Publish Stage", the packet flow remains the same.

We use Fig. 3 as the example of our description. In Fig. 3, IR_1 is an IP router. CR_{0-6} are CCN-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, we have the CCN module at CCN-aware routers. The edge routers have CCN modules with additional functionality (*e.g.*, encapsulating, decapsulating and ST for end-hosts). Since a CCN-aware router can serve as 0, 1 or more RPs, we design the RP as a logically separate module for multicast and renaming. Thus, on router CR_0, there exists both CCN module and RP module. When an RP is setup on a router, the RP module registers a FIB entry on the router's CCN module: name=`/RP`, face=$R_{x:RP}$. We use $CR_{x:CCN}$ to denote the `{address:port}` pair of the CCN/edge module on CR_x and $CR_{x:RP}$ for RP module on CR_x.

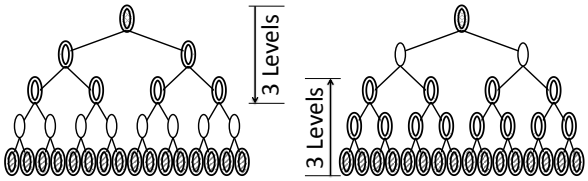
On receiving a Subscribe packet from the subscriber (S_1), the edge router ($CR_{2:CCN}$) will add ST and forward it using UDP packet whose destination is $CR_{0:CCN}$ according to FIB(RP_Table(CD)). Here, we use a flag bit in the UDP packet to mark the special message type. When IR_1 receives the packet, 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 $CR_{1:CCN}$. Its CCN module treats the Subscribe packet similar to an Interest packet. An entry (name=`/RP/sub`, face=$CR_{2:CCN}$) will be added into the Pending Interest Table (PIT) that contains the requests that are yet to be served. When $CR_{1:CCN}$ sends the UDP packet out, the *from* field of the UDP packet is changed from $CR_{2:CCN}$ to $CR_{1:CCN}$. When $CR_{0:CCN}$ receives the Subscribe packet, it does the same as $CR_{1:CCN}$, but the face of the PIT entry is $CR_{1:CCN}$. 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 §3.2). The CCN module treats this packet similar to Data packets with extra FIB add action. $CR_{0:CCN}$ adds $FIB(/RP/query) = CR_{0:RP}$, $CS(/RP/sub) = Join$. Subsequently, this Join packet will consume $PIT(/RP/sub)$ and be forwarded to $CR_{1:CCN}$. $FIB(/RP/query)$ will be created along the path to $CR_{2:CCN}$ and Content Store entry for `/RP/sub` will be stored in the intermediate routers. When edge module ($CR_{2:CCN}$) 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 §3.2). When another subscriber S_2 also tries to join CD, $CR_{1:CCN}$ 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, $R_{4:CCN}$ will add $ST(CD/x) = S_2$ but subscribe to the CD upstream, so as to still get hit on $CR_{1:CCN}$. Fig. 3 shows the FIB for `/RP` (in red dotted lines, for multicast) and FIB for `/RP/query` (in green dashed lines, for query/response).

FIB Propagation from Publishers

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 an FIB entry for every possible publisher on the RPs is not advisable in the pub/sub environment. We therefore require FIB creation information to be piggybacked with the first Publish packet. If a publisher needs to serve a prefix (he wants the subscribers to issue a query for the whole data), he will encapsulate the prefix in a Publish packet. In Fig. 3, P_1 will encapsulate `/P1` in the Publish packet. On seeing the piggybacked information in the Publish packet, the $CR_{6:CCN}$ will setup $FIB(/P1) = P_1$ and forward the packet to $CR_{0:CCN}$ ($CR_{5:CCN}$ will not see the packet). $CR_{0:CCN}$ will add $FIB(/P1) = CR_{6:CCN}$. The FIB is shown in yellow solid line in Fig. 3.

Data Dissemination According to FIB

On receiving a snippet with CD and data ID `/P1/Data1`, S_1 queries data with `/P1/Data1` and include CD as a reference. When $CR_{2:CCN}$ receives the packet, it adds $PIT(/RP/query/P1/Data1) = S_1$ and forwards it according to entry $FIB(/RP/query)$, which is $CR_{1:CCN}$. Then, $CR_{1:CCN}$ forwards it to $CR_{0:RP}$ through $CR_{0:CCN}$. $CR_{0:RP}$ removes the `/RP/query` prefix and forwards the Interest (ContentName=`/P1/Data1`) to P_1 through $CR_{0:CCN}$. 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. CCN 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 CCN module CR_0, CR_1 and CR_2.



(a) Top Down **(b)** Bottom Up

Figure 4: Possible Incremental Deployment Strategies

4.2 Strategy to Enable CCN-aware Routers

Although deploying a larger number of caches in the network can increase the cache hit rate, incremental deployment of the CCN 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 CCN 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 we assume the ISPs have a limited maximum amount of cache that can be deployed across the various nodes in the network. This then raises the question of which routers should be replaced/enabled with the CCN (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 leaves of the tree with the RP as the root node. In order to better understand the trade-off, we analyze 2 possible ways of deploying CCN-enabled routers in the network, considering the logical multicast topology: top down (from the RP down) and bottom up (starting from the end-hosts/subscribers). Fig. 4 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 CCN-enabled routers. The CCN enabled routers are marked as nodes with a double circle in the figure. The *top down* strategy deploys CCN enabled routers starting from the routers directly connected to the RP in the logical tree. Fig. 4a shows the structure with 3 levels of CCN-enabled routers according to top down strategy. The *bottom up* strategy deploys CCN enabled routers starting from the routers directly linked to edge (leaf) routers in the tree. Fig. 4b shows the structure with 3 levels of CCN-enabled routers based on the bottom up strategy. Note Fig. 4 is for illustration purposes (we realize here the number of CCN 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

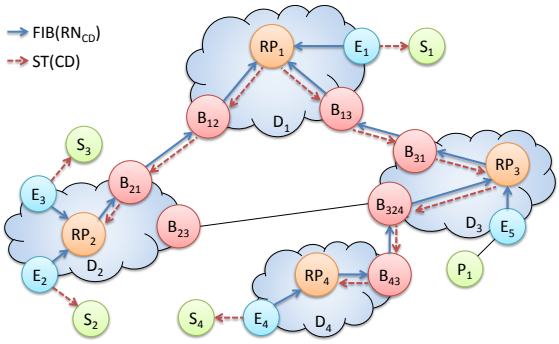


Figure 5: Inter-domain multicast

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 CCN enabled routers. A smaller cache at each node may result in a lower cache hit rate. Alternately a smaller number of CCN 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. We will compare these two strategies in our evaluation.

5. Hybrid-COPSS INTER-DOMAIN

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). We combine overlay multicast at the COPSS 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 us to have different CD to IP multicast mapping in each domain based on considerations such as the load and subscriber count for each CD. We take 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. 5, similar to the requirements of the single domain solution, the edge routers and the RP in each domain are COPSS aware routers. Additionally, in the multi-domain case, we require the boundary routers (marked B_x) to be COPSS 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.

5.1 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 the 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, we suggest that the local RP 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. 5, 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. 5 shows the result of RP setup started at domain D_1, triggered by S_1. Notice 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.

5.2 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. 5 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.

5.3 Publish

For the intra-domain multicast, the local RP multicasts the packet using local $Group_{CD}$. But on the overlay, we use a shortcut-enabled multicast tree 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. 5 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 Multicast (overlay) 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.

5.4 Automatic Rendezvous Point Balancing

In G-COPSS [8], an automatic RP balancing method was proposed to relieve traffic concentration ("hot spots"). We adopt this solution for 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. MANAGING ADDRESS MAPPINGS

In this section, we examine efficiency and scalability related issues of hybrid-COPSS. First is the issue of management of the mapping table between the CDs and the Multicast groups on RPs (*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. The second issue is the management of the table containing the mapping of CDs to a Rendezvous Point (*i.e.*, $RN_{CD} = RP\_Table(CD)$). Although hybrid-COPSS requires that RN_{CD} is maintained in every COPSS edge router, we seek to limit the size of this table to enable the solution to scale better even when we have millions of CDs.

6.1 CD to Multicast Group Mapping

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 id. The mapping of the unbounded, hierarchical CCN namespace onto a bounded, flat IP multicast group address space will naturally result in wasteful traffic being sent on links in the network. But different mapping functions can result in varying amounts of wastage. Imagine that there are 2 CDs: `FireAlarm` (subscribed to by almost everyone but does not have many updates) and `CCNMailingList` (subscribed to by only a few people but with frequent updates). If we map them both onto one IP Multicast group, this group will be subscribed by almost everyone. This implies that the updates in the `CCNMailingList` will be received and discarded by almost every edge router, resulting in a large amount of wasted traffic 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 a Content-Centric Network, we assume that neither publishers nor RPs 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. We suggest 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 our approach, 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 a COPSS 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, although we have not explored this in detail yet. 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. Notice 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. Our ongoing work is to examine the details of this mapping and its effect.

6.2 Management of the RP Table

To limit the size of table containing the mapping of a CD to an RP ($RN_{CD} = RP\_Table(CD)$) at every edge router and for ease of management, we use a broker to maintain the complete database in each domain. The smaller table for RN_{CD} at the edge routers are treated as a cache. When a router has a cache miss, it goes to the broker. Similar to the NDN design, intermediate routers can also respond to this request. The data structure of storing RN_{CD} on the router can be chosen from one of two options. The first is a traditional CD to RP mapping table. The index of CDs can be grouped into a tree structure to optimize search performance. The router would only have to map the CD once before it sends the packet. The second option is a bloom filter based RN_{CD} table. For every RP, there is a (counting) bloom filter storing the hash of the CDs it serves. This can compress the index greatly and reduce the cache miss rate. However, since we can have false positives, the router will have to test all the bloom filters before it can forward the packet. This could 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.

Table 1: Avg. Forwarding Latency(95%CI)

(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)**
Rendezvous Point	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)

7. EVALUATION

In this section, we microbenchmark a hybrid-COPSS implementation on our test-bed compared with pure COPSS for the forwarding efficiency and queuing. We then use an online gaming trace and a Twitter trace to evaluate the performance of these architectures under load with a simulation parameterized by the microbenchmark results.

7.1 Microbenchmarking

We implemented hybrid-COPSS on our lab test-bed and compare it to the implementation described in [8]. Similar to [8], 62 players are used to load the test-bed implementation, but with a longer period ($2min$ warm up and $10min$ evaluation) from the gaming trace to get statistically significant results. We also traced every packet using Wireshark and calculate 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, we do not breakdown the performance of individual encapsulation, decapsulation and forwarding functions. These routers are treated as black boxes. However, for internal routers, we separately measure the functionality of unicast (Interest in COPSS; UDP unicast in hybrid-COPSS) and multicast (Multicast in COPSS; UDP multicast in hybrid-COPSS) forwarding.

Microbenchmarking Results

We show the average forwarding latency on different routers along with the 95% confidence intervals in Table 1. The results confirmed the observation in [7] that CCN (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/IP 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 quick. In hybrid-COPSS, although it incurs a 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 shows the CDF of the total update latency. Observe 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 we used a simple test-bed topology, with only 1-2 internal routers between the RP and edge routers, we expect higher performance gains in a typical network topology with more intermediate hops.

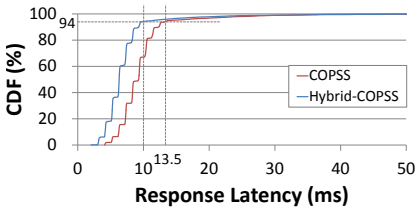


Figure 6: Test Bed Response Latency (CDF)

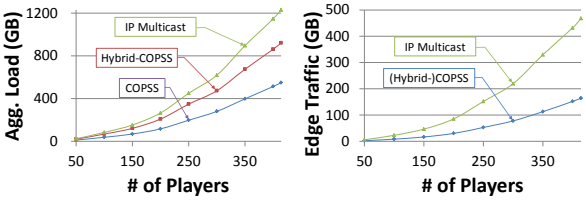
7.2 Large Scale Simulation

To further evaluate the performance of hybrid-COPSS in a large scale environment with realistic network topologies, we use trace-driven simulation, with two traces: one from a multiplayer online game, and the second from Twitter. 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. We do this to emulate scenarios where the ratio of publishers to subscibers 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.

Data Traces

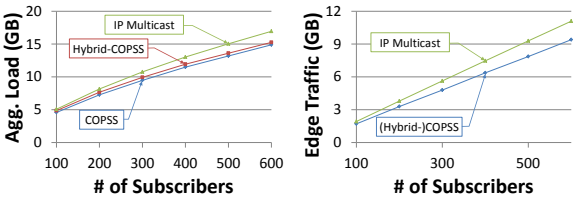
Game Trace: We first modified a Counter-Strike (CS) trace obtained on a busy CS server in a $7h05m25s$ period [26]. 414 unique players who published $10,686,950$ packets (average publish frequency is around $2.39ms/packet$) were in the trace. We also created several subsets of the trace (the # of players varied from 50 to 400) to evaluate the scalability of our architecture. All the players share a global hierarchically partitioned map divided into 5 regions. Each region is further divided into 5 zones. A player is able to see and modify objects based on his location in the game and the hierarchy of the area he belongs to. So the RP will actually be busier than the original server (where at most 22 players may share an instance of the game, based on the CS server configuration). 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

Twitter Trace: We used a Twitter trace on technical topics obtained from the public Internet during a one-week period in 2010. We identified and selected 25 popular keywords such as *iphone*, *ipad*, *blackberry*, *smartphone* as 1^{st} level CDs and created a subset of the trace containing $41,613$ tweets from $22,987$ users that contain these keywords. These 25 key words are treated as the 1^{st} level CDs. We further identify secondary popular keywords (up to 25 keywords) that are associated with these 1^{st} level CD keywords. We build a tree structure off of all these CDs. There are a a total of 407 distinct leaf CDs. To have an adequate number of tweets originating from each publisher in our system, we assigned the $22,987$ users to 50 publishers. The individual users were hashed to a publisher using a power-law. Every 1^{st} level CD uses an IP multicast address. To have a suffi-



(a) Aggregate Network Load **(b)** Edge Traffic

Figure 7: Single Domain Performance: Game Trace



(a) Aggregate Network Load **(b)** Edge Traffic

Figure 8: Single Domain Performance: Twitter Trace

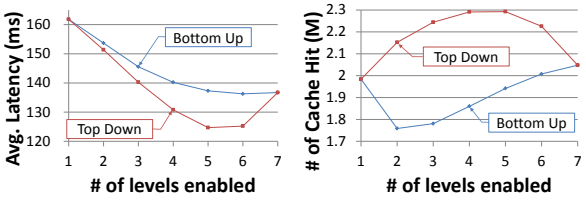
ciently large message for the query/response case, we scaled the tweet size by a factor of 128. A publisher publishes the original message size as a snippet first and then the subscribers probabilistically query for the complete message on receiving the initial snippet (tweet).

Single Domain: Pub/Sub

In single domain case, we use RocketFuel 3697 [27] (79 routers) as the core topology. A total of 200 edge routers are randomly assigned to the core routers (1-3 edge router(s) per core router). For the evaluation of hybrid-COPSS in this case, we used both the Twitter and Gaming traces. 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 (in [8], we showed that 3 RPs are sufficient to efficiently handle the gaming trace and we use the same environment for the Twitter trace). 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 update latency of around $73.7ms$, compared to $84.6ms$ with COPSS. However, because of an insufficient # of IP multicast groups (we map 7 CDs onto 1 IP multicast group), hybrid-COPSS causes a larger load in the network compared to COPSS, but less than pure IP multicast, as shown in Fig. 7a when we vary the number of players. This demonstrates that our solution can be integrated into the current IP architecture without substantial performance degradation.

With the same # 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



(a) Avg. Response Latency **(b)** Cache Hit Rate

Figure 9: Performance of Different Incremental Deployment Strategies

is shown in Fig. 7b. 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. 8 illustrates the same trend for the Twitter trace, except that the network load(Fig. 8a) and edge traffic(Fig. 8b) grow sub-linearly. This is due to the fact that although there is an increase in the # 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.

Single Domain: Query/Response

We ran a simulation to compare the performance of the two strategies. To understand the results better, we view the topology in the abstract as a 8-level binary tree (composed of 255 routers). 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 $10ms$). We used the Twitter trace, where snippets are sent as an announcement and the subscribers then query for data they are interested in (*e.g.*, video clips) [7]. 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 CCN enabled routers in the network. The total published data size is around $4.5GB$.

The results are shown in Fig. 9, where the x-axis $level = n$ denotes the number of levels of routers that have their CCN cache enabled. Therefore $level = 2$ implies that 2 levels of routers have their 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 CCN enabled, and for $level = 7$, all the routers in the network are CCN enabled.

There are multiple criteria interacting here that affect the tradeoff of where and how many CCN 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 CCN routers), with a per node cache size of 20.24 MB, we see a lower overall cache hit rate than with $level = 2$ (RP, 128 edge and 2 extra CCN routers) where the per node cache size is smaller at 19.93 MB. The higher cache hit rate is because of the aggregation of requests across uesrs at the second level in the tree that the

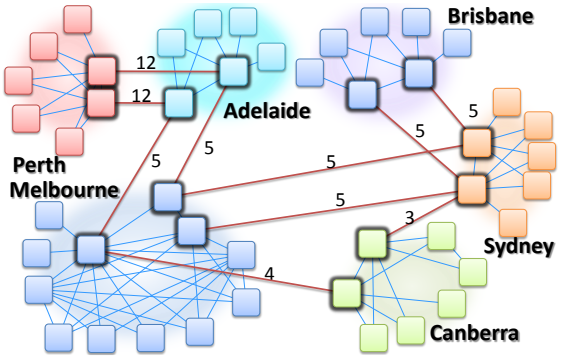
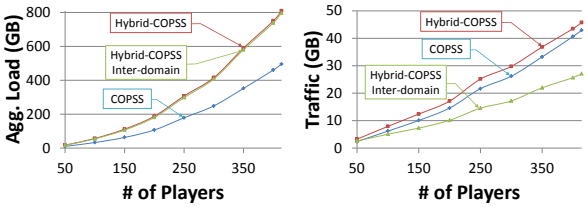


Figure 10: Multi-domain Topology: RocketFuel 1221



(a) Aggregate Network Load **(b)** Inter-domain Traffic

Figure 11: Multi-Domain Performance: Game Trace

2 extra CCN routers can respond to from their caches. The latency also reduces as we increase the number of levels of CCN routers. 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 we go to $level = 6$ or $level = 7$, the cache hit rate reduces because the individual cache sizes are now smaller than the working set size. Correspondingly, the response latency goes up, even though there are more CCN routers further down the tree. When we look at the bottom-up strategy, we see that 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 CCN routers goes up. This is because the total cache is divided across more routers (RP, 128 edge and 64 extra CCN routers), because the extra CCN routers do not see a benefit of any aggregation of requests and do no yield an increase in overall cache hits. As we 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 we compare 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 CCN routers may be used (*e.g.*, top-down $level = 2$ has only 3 extra CCN routers, bottom-up $level = 2$ has 64 extra CCN 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.

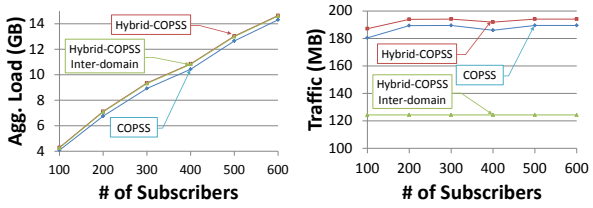
This result suggests that when an ISP has a fixed number of CCN nodes to deploy in the network (i.e., building an overlay network), it is better to take a top-down approach, enabling the CCN functionality starting from the nodes directly linked to the RP, but make sure every router having enough cache size compared to working set size. The top-down strategy can achieve lower response latency with fewer expensive CCN-enabled routers. This serves as an indication that the RP-based dissemination tree structure is better than a publisher focused, source-based tree (see Fig. 2a) 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.

Multi-Domain Simulation

We then investigate hybrid-COPSS in a multi-domain scenario using RocketFuel 1221 Australia, which has clear domain structure, as our core topology. According to [27], Telstra has hubs in major cities (Sydney, Melbourne, Perth) with spokes elsewhere. We consider every city as a domain and the routers at these major cities as the core routers. The topology (Fig. 10) 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. We then add 207 edge routers 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 $8ms$. 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.

Similar to the single domain, for the multi-domain Gaming case, hybrid-COPSS achieves lower average update latency (around $52.09ms$ compared to $61.13ms$ 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 multicast routing can take advantage of shortcuts across domains, a multicast packet doesn't have to go all the way to the global RP to be forwarded to subscribers. Our solution reduces the average update latency by about 2.46ms. Hybrid-COPSS also cuts the inter-domain traffic almost by half (Fig. 11b), and reduces the aggregate network load slightly compared to hybrid-COPSS (Fig. 11a). This means the inter-domain solution is much "cheaper" for ISPs even compared to a pure COPSS solution. Moreover, as described previously, 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 ($51.35ms$ vs. $59.05ms$) and inter-domain solution reduces it by an additional $1ms$. 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



(a) Aggregate Network Load **(b)** Inter-domain Traffic

Figure 12: Multi-Domain Performance: Twitter Trace

subscribers (Fig. 12a). But, we can observe 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. 12b).

8. SUMMARY

In this paper, we present hybrid-COPSS, an architecture to integrate CCN functionality with the current IP architecture. We present a detailed solution to integrate both the pub/sub and the query/response based content-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. In this paper we have addressed the 3-way tradeoff that arises when considering the incremental deployment of CCN, 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 we go more towards a pure IP environment; there is increased latency in a pure CCN environment because of the additional per-hop processing; finally there is an increase in processing cost for each CCN hop because of the additional complexity. Our evaluations suggest that hybrid-COPSS strives to achieve a proper balance in this trade-off by putting CCN functionality at key points and hash-based forwarding at the other routers. Moreover, we optimize the query/response dimension of hybrid-COPSS and show that an RP based top down approach provides the best means for service providers to incrementally deploy caching-capable CCN nodes. The hybrid-COPSS inter-domain solution recognizes the current challenges in having inter-domain IP multicast and overcomes it with the use of CCN overlay nodes at the domain edges. The inter-domain hybrid-COPSS cuts inter-domain traffic almost by half even compared to our earlier CCN proposal, COPSS.

9. REFERENCES

[1] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps, "Content Based Routing with Elvin," in *AUUG2K*, 2000.

[2] A. Carzaniga, M. Rutherford, and A. Wolf, "A routing scheme for content-based networking," in *INFOCOM*, 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 *SIGCOMM*, 2007.

[4] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J. Thornton, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *CoNEXT*, 2009.

[6] W. Fenner, D. Srivastava, K. K. Ramakrishnan, D. Srivastava, and Y. Zhang, "XTreeNet: Scalable Overlay Networks for XML Content Dissemination and Querying," in *WCW*, 2005.

[7] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System," in *ANCS*, 2011.

[8] J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "G-COPSS: A Content Centric Communication Infrastructure for Gaming," in *ICDCS*, 2012.

[9] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: a high performance publish-subscribe system for the world wide web," in *NSDI*, 2006.

[10] A. R. Bharambe, S. Rao, and S. Seshan, "Mercury: a scalable publish-subscribe system for Internet games," in *NetGames*, 2002.

[11] C. Esteve, F. Verdi, and M. Magalhaes, "Towards a new generation of information-oriented Internetworking architectures," in *ReArch*, 2008.

[12] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM TOCS*, pp. 332–383, 2001.

[13] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "SpiderCast: a scalable interest-aware overlay for topic-based pub/sub communication," in *DEBS*, 2007.

[14] 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," INRIA, Research Report, December 2005.

[15] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.

[16] 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 *ReArch*, 2009.

[17] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: audio conference tool over named data networking," in *ICN*, 2011.

[18] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *ICN*, 2011.

[19] A. Adams and W. S. J. Nicholals, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," RFC 3973, January 2005.

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

[21] H. Holbrook and B. Cain, "Source-specfic Multicast for IP," RFC 4607, August 2005.

[22] H. Eriksson, "Mbone: the multicast backbone," *Commun. ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[23] Y. Cui, B. Li, and K. Nahrstedt, "ostream: asynchronous streaming multicast in application-layer overlay networks," *JSAC*, vol. 22, no. 1, pp. 91 – 106, 2004.

[24] J. Jannotti, D.-K. Gifford, and K.-L. Johnsonand, "Overcast: Reliable Multicasting with an Overlay Network," in *OSDI*, 2000.

[25] J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "Coexist: A Hybrid Approach for Content Oriented Publish/Subscribe Systems," in *ICN*, 2012.

[26] W. Feng, "On-line Games," http://www.thefengs.com/wuchang/work/cstrike/.

[27] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring Link Weights using End-to-End Measurements," in *IMW*, 2002.

4 Gaming Over Content Oriented Publish Subscribe System

G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications

Jiachen Chen*, Mayutan Arumaithurai*†, Xiaoming Fu* and K.K. Ramakrishnan‡
*Institute of Computer Science, University of Goettingen, Germany,
Email: {jiachen, arumaithurai, fu}@cs.uni-goettingen.de
†NEC Laboratories Europe, Heidelberg, Germany, Email: arumaithurai@nw.neclab.eu
‡AT&T Labs Research, Florham Park, NJ, U.S.A., Email: kkrama@research.att.com

***Abstract*—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 (MMORPG). Currently, MMORPG is 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. We propose Gaming over COPSS (G-COPSS), a distributed communication infrastructure using a Content-Oriented Pub/Sub System (COPSS) to enable efficient decentralized information dissemination in MMORPG, 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.**

We have implemented G-COPSS on top of the open-source CCNx implementation. We use a simple game with a hierarchical map to carefully microbenchmark the implementation and the processing involved in managing game dynamics. We have also microbenchmarked the game based on NDN and a server with an IP infrastructure. We emulate an application that is particularly emblematic of MMORPG – Counter-Strike – but one in which all players share a hierarchical structured map. Using trace-driven simulation, we demonstrate that G-COPSS can achieve high scalability and tight timeliness requirements of MMORPG. The simulator is parameterized based on microbenchmarks of our implementation. Our evaluations 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.

I. Introduction

Massively Multiplayer Online Role Playing Games (MMORPG) 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 position-s/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's 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 communication requirement is eminently satisfied by a content-centric networking (CCN) paradigm [1]: players publish updates to an area/object without regard to who's supposed to see it; at the same time, players subscribe to the areas/objects they can see, without knowing who else is trying to modify them. Publish/subscribe (pub/sub) systems are particularly suited for large scale information dissemination, and provide the flexibility for users to only subscribe to the information of interest to them. However, a straight-forward pub/sub system will still depend on a server for tracking the source of the information. But, a content-oriented pub/sub further decouples the information delivered to the subscribers from knowing the specific publishers that

publish a piece of information (e.g., in a game scenario, related players' positions/actions). With the use of an appropriate interface, users can select and filter the information desired, irrespective of the publisher of this information. Such a capability is eminently supported by a content oriented pub/sub system such as COPSS [2]. COPSS is an implementation of content-oriented pub/sub which exploits multicast and a hierarchical Content Descriptor (CD) naming framework. In this work, G-COPSS fine-tunes and optimizes COPSS to support the specific needs of a game environment.

The key contribution of G-COPSS is to provide an efficient, distributed communication infrastructure for MMORPG. In more detail, our 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.
- We evaluate the performance of G-COPSS by performing a micro-benchmark of our implementation of G-COPSS and compare it to a NDN and an IP-server based solution. Moreover, we use the data trace of the Counter-Strike game to perform large scale tests to demonstrate the performance gains in terms of aggregate network load, server load and improved player experience.

We review the related work in §II and present a brief background of CCN and COPSS. We present the design of the G-COPSS infrastructure in §III and the game specific add-ons in §IV. Evaluation results are given in §V. We conclude our work and outline future work in §VI.

II. Related Work

Modern fast-paced action games run on a Client/Server (C/S) architecture which limits the number of players who can interact simultaneously, since the server needs to handle the frequent updates and disseminate it. Feng *et al.* [3] observed that due to this reason, 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 (SL), has multiple dedicated servers to support each of its 18,000 regions [4]. Studies such as [4]–[6] show that SL 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.* [5] also show that the management of a region with only 5,000 rigid-body objects requires about 72% of the server computational power. We believe that 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.

Peer-to-peer (P2P)-based approaches (e.g., [7]–[11]) provide an alternative to C/S-based gaming approaches. E.g., Varvello *et al.* [11], [12] propose a DHT-based architecture for SL to overcome the limitations of the C/S architecture. Donnybrook [10] 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., [13]), 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 (we could have 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.

The notion of Content Centric Networking (CCN) (for which [14] is a popular example) can offer a new opportunity, where content/information is looked up and delivered according to its name without knowing the identity and location of the sender. NDN uses two packet types, $Interest$ and $Data$. A consumer queries for named content by sending an Interest packet; a provider in turn responds with a Data packet which consumes this Interest packet. NDN requires a new forwarding engine instead of IP, which contains the FIB (Forwarding Information Base), Content Store (buffer memory which caches content) and PIT (Pending Interest Table). FIB is used to forward Interest packets toward potential source(s) of matching Data. PIT keeps track of 'bread crumbs' of Interest (i.e., reverse-path forwarding) which the Data packets follow to reach the original requester(s). However NDN alone does not support an efficient pub/sub-way of information dissemination, which is ideal for an MMORPG. Content-Oriented Publish/Subscribe System (COPSS) was designed to achieve communication efficiency for pub/sub applications. In this paper we propose G-COPSS, which extends COPSS, to address the unique needs of gaming

like applications such as sensitivity to latency and dynamic traffic concentration (hot spots caused due to the increase in popularity/action of a particular zone in a game). Note that the architectural enhancements proposed in this paper can also be integrated into COPSS to improve its performance.

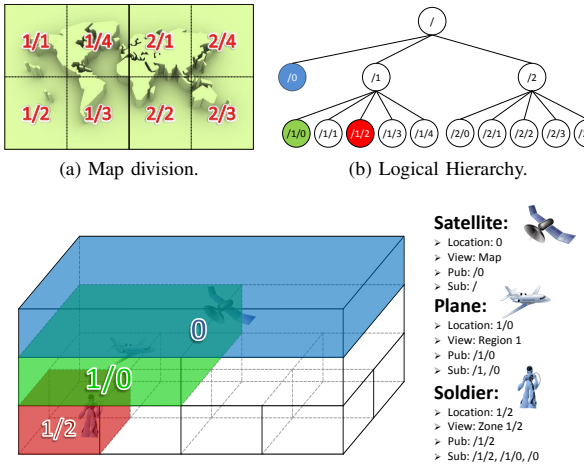
III. G-COPSS: An Efficient Communication Infrastructure for MMORPG

G-COPSS is designed as a decentralized content-centric communication framework to support MMORPG. 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. In [2] we proposed Content-Oriented Pub/Sub System (COPSS) as a means to achieve an efficient publish/subscribe capability that predominantly uses multicast for content-centric networks. G-COPSS builds on COPSS to support the specific needs of a typical game environment. G-COPSS utilizes COPSS's predominantly 'push-based' multicast framework to ensure that players receive timely updates. Using a content-centric solution (which we envisage will eventually be part of a general network layer) 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, we can take advantage of the typical schema and the natural partitioning of a game map. We generalize the game map by considering the notion of hierarchical game maps. We formulate a naming hierarchy suitable for even complex games that may include a hierarchical game map partitioned in arbitrary ways. We then describe the communication framework for G-COPSS in detail, including the implementation of G-COPSS on top of the open source CCNx code plus our enhancements to COPSS. Subsequently, we will describe efficient gaming specific optimizations.

A. *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. In G-COPSS, we take that optimization a step further by considering a multi-layer hierarchical relationship between the various areas in the environment, as shown in Fig. 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 1a shows



(a) Map division. (b) Logical Hierarchy.

(c) Hierarchical subscription.

Figure 1. **Hierarchical map partition.**

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. To match the information generated by a producer to the consumers interested, it is important to have an easily understandable nomenclature. Here, we exploit the concept of Content Descriptors (e.g., described in [15]). Content Descriptors (CDs) can be used to represent elements in a group or are part of a topic hierarchy. The schema used for the partitioning of the map forms the schema for our 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. 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`, `1/2`, `1/3` and `1/4`. G-COPSS in fact allows map designers to divide the map into arbitrary layers, but for simplicity, we only use 3 layers in this paper. CDs form the naming framework corresponding to the hierarchical location map for 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 `/1` since he would then receive updates from all the players belonging to

the zone-layer of /1. To address this issue, we introduce the concept of having every zone/area represented also as a leaf node. This allows finer-grained publication and subscription. To perform this, we create a /0 for every non-leaf CD in the hierarchy, i.e., /0 for the top map layer (/) and /1/0 for the region /1. These /0s are used to represent the areas above, e.g., where planes are flying (shown in 3D partition Fig. 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 /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. In G-COPSS, we make 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.

B. *Communication framework*

The hierarchical CDs described above form the basis of the multicast based communication model wherein virtual Rendezvous Points (RPs) are assigned to act as the core of the multicast treetowards 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 G-COPSS is able to dynamically add and delete RPs based on the demand (see Section IV). Every G-COPSS-aware router maintains a subscription table (ST) that consists of the various CDs and the outgoing interfaces for those messages. Therefore, whenever this router receives a message intended for the group /sports, it will look at its ST entry and forward it to the appropriate interfaces resulting in multicast based delivery.

Rendezvous Point Setup

We create prefix-free virtual RPs that are responsible for forwarding updates from players to the other players subscribed to the same or higher groups in the hierarchy. The term "prefix-free" mandates that a prefix is served by only one RP, e.g., if an RP is serving the prefix /1/1, this same RP would also be serving /1/1/1 and no RP could serve /1. However, another RP could serve /1/2 or /1/3. This design ensures that messages will only be sent to one RP for that CD and all the CDs below it in the naming hierarchy. Therefore a player sending an update to /1/1/1 will forward it to this RP which in turns forwards it to not only subscribers subscribed to /1/1/1, but also to those subscribed to /1/1 thereby ensuring that hierarchy based subscription is preserved. The absence of a CD /1 in this example implies that a subscriber wishing to subscribe to /1 must subscribe to those RPs serving /1/1, /1/2 and so on. The aggregation of subscriptions at the subscription table ensures that the subscriber wishing to subscribe to /1 is automatically subscribed to /1/1, /1/2, etc., till /1/n.

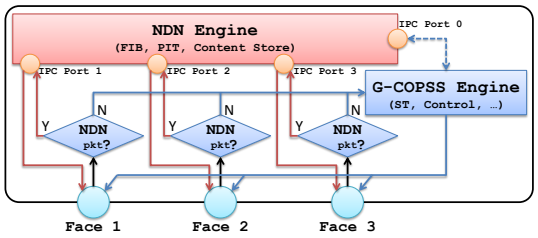


Figure 2. **G-COPSS Router Architecture**

Update Dissemination in Gaming

Using this naming hierarchy, a player can send an update using a leaf CD representing the area that contains an object (including himself) he has modified. The player can also subscribe to the leaf CDs that represents the areas he/she has visibility into (areas of interest (AoI)). Furthermore, subscriptions to CDs can be aggregated to a higher level in the hierarchy. According to [3], 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.

Hierarchical Publishing: Players need to publish the updates they make to all interested-players (interested-players are those that view the same AoI/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. 1c, he will send the message to the RP serving the CD /0; if he shoots at a plane in the green area, he will disseminate the information using the CD /1/0; and if a soldier is moving in the red area, he will disseminate it using the associated CD /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 he is in and all the /0s along the hierarchy. E.g., a player standing on 1/2 should subscribe to /0, /1/0 (the /0s along the hierarchy) and /1/2 (the area he is in). This allows him to see the units standing on 1/2, the planes flying over zone 1, and the satellite at top (so that 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 /0s along the hierarchy) and /1 (the area he is in).

C. *Design Details*

In this section, we briefly explain the implementation of our G-COPSS on top of a NDN [14] aware router. We ensure that it is backward compatible with NDN, ensuring that the query-response based applications also function seamlessly. Pub/sub capability with multicast is provided by adding 3 packet types ($Subscribe$, $Unsubscribe$ and $Multicast$) and introducing a new data structure ST (Subscription Table) to the CCNx forwarding engine. In addition, we use FIB add/remove packets to directly deal with maintaining the FIB. ST is a $<Face$, $BloomFilter<CD>>$ table that stores the subscriptions for each outgoing face, where the Bloom Filter describes the CD set. Thus, a multicast packet m will be forwarded to face f if m has CD(s) that find a match in the Bloom Filter of f. If a packet arrives for a CD `/sports/football` the Bloom Filter would be checked for `/sports` and for `/sports/football` to identify all the appropriate outgoing interfaces for both levels of subscription. One possible optimization is to calculate the hash values at the 1^st hop router and the routers forward hash values along with the names. So routers only need to perform simple bit comparison in Bloom Filter to reduce the latency.

Fig 2 shows the architecture of a G-COPSS router, where each face is served by an Inter-Process Communication (IPC) port. For original NDN packets ($Interest$, $Data$, etc.), these faces work as a tunnel to continue to provide the NDN functionality. E.g., if an $Interest$ coming from `Face 1` passes the "`is a NDN pkt?`" check, it is forwarded to NDN (`localhost:9695`) via `IPC Port 1`. On receiving the $Interest$, if NDN returns a $Data$ packet (content store/cache hit), it will send the $Data$ to the COPSS engine via `IPC Port 1`, which in turn will send it out via `Face 1`.

On receiving a FIB $add/removal$ packet from face f, we add/remove NDN FIB entries using the IPC port associated with f. For efficiency, each FIB $add/removal$ packet can contain multiple $ContentNames$. If `Face 2` receives a FIB add packet requesting to add a name prefix `/sports/football`, the NDN engine will add an FIB {prefix=`/sports/football`, face=`IPC Port 2`}. On receiving an Interest packet with the prefix from another face (`Face 1`), G-COPSS will forward it to the NDN engine via `IPC Port 1`. The NDN engine will then decide to forward it to `IPC Port 2`. Since an IPC Port will forward incoming packets to the corresponding face, `Face 2` will receive the $Interest$ as needed.

A dedicated IPC tunnel (`0`) is introduced for the communication between G-COPSS and NDN engines. On receiving a $Multicast$ packet from an end host, the G-COPSS engine will encapsulate it into an $Interest$ packet and then send it to the NDN engine via this port. When a router functions as an RP (e.g., `RPx`), a FIB entry will be added to the NDN engine: {prefix=`/RPx`, face=`IPC Port 0`}. When an encapsulated $Multicast$ packet (an $Interest$) is forwarded to RP, that router's NDN engine will forward it to the special port. On receiving an $Interest$ from that port, G-COPSS will try to decapsulate the packet and forward it using ST.

Some G-COPSS functionalities are achieved independent of the NDN engine. For example, on receiving a $Subscribe/Unsubscribe$ packet, the router updates the local ST and decides if it will join/leave some groups according to the downstream subscription status. On receiving a $Multicast$ packet from another router, it will forward the packet to the faces directly based on the ST.

D. *Incremental deployment*

COPSS+IP was introduced in [2] as an incremental deployment step wherein the COPSS enabled routers at the edge are used to provide content centric functionality while the intermediate IP routers provide the forwarding efficiency. In this section, we describe how G-COPSS can be designed to function in a COPSS+IP environment.

The key to adopting G-COPSS in a COPSS+IP environment (hybrid-G-COPSS) is to map the multitude of hierarchical CDs to the limited IP multicast space. An ideal scenario would be to perform a one-to-one mapping of the leaf CDs. But taking into consideration the fact that billions of CDs could exist in a content centric environment, this could result in more than one CD being mapped to a single IP multicast address. In G-COPSS, based on the available IP multicast address space, the COPSS enabled edge routers would hash the high level CDs on to a single IP multicast address rather than directly hashing the leaf CDs. This allows the mapping tables at the edge COPSS routers to aggregate mapping entries. Moreover, this mechanism allows a message received by `/1/1/1` to be forwarded to the players subscribed to `/1/1` and `/1` too. Due to the mapping of multiple CDs on to one IP multicast group, unwanted messages may be forwarded along branches of the network, too. E.g, if `/1` and `/2` are mapped to one IP group, a message to `/1` would be sent to all the subscribers including those subscribed to `/2`. The COPSS enabled router close to the receiver side is then entrusted with the task of filtering out the unwanted messages and forwarding only those messages intended for the receiver. In §V, we evaluate the trade-off between the G-COPSS and hybrid-G-COPSS approaches.

IV. Game Specific Add-ons

A. *Message Dissemination Upon Players Moving*

It is natural for a player to move from one sub-world to another as well as from being offline to coming online. Besides the general pub/sub support provided in COPSS for offline users, G-COPSS provides the following additional facility to support effective dissemination for player movement.

When a player enters (or approaches) a new sub-world, 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.

There are several options for a player to retrieve the latest snapshot of the area he is interested in, e.g., a query-response based approach and a cyclic-multicast based approach presented below:

Query-response based approach (QR): This approach exploits the Query-Response functionality in NDN. When a player moves/teleports to another subworld, he queries for the snapshot of that sub-world (but only the part 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.

Cyclic-multicast based: 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 $Subscribe$ packet and stops on receiving the last $Unsubscribe$ packet. 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 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.

B. Handling Hot Spots and Traffic Concentration

It is common for a team of players to collaborate and play in an area together thus resulting in some game zones becoming very popular (or hot) in a short span of time. The *hot spots* in some regions of the games are therefore likely to be subjected to high loads due to frequent updates from many players. The multicast capability in G-COPSS permits scaling to a large number of players. Nevertheless, like core-based tree approaches in IP multicast, if multiple publishers send updates to CDs served by one RP, traffic concentration will result on the links leading to that RP and could 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. As shown in Fig. 5b, 2 RPs are adequate for the first 70,000 packets of that trace. We then experience high update latencies. By using 3 RPs we satisfy the needs for this game trace (cf. Fig. 5a). However, a more general solution is required.

G-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 the popularity of a CD. When the packet queue at a router R that serves as an RP is above a certain threshold, the creation of a new RP is triggered automatically. The hierarchical map formation allows seamless decentralization and load-balancing of RPs. As the load on the RP increases, it can offload some CDs to a new RP. Since players send multicast packets associated with a CD (rather than the address of the RP), users are not directly affected by the addition/removal of RPs. The new RP only needs to indicate that it serves the new CDs. G-COPSS modifies the FIB and ST accordingly. Packets are redirected automatically to the new RP and a new multicast tree is formed. We detail the steps.

New RP & 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 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 like [16]. Here, we use a random selection process to divide the load equally among the RPs. It can be further optimized using prediction to ensure that addition and deletion of RPs are not performed frequently.

Reverse the FIB & ST entries between the old and the new RP: We seek to efficiently create a 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 RP R'. R sends a packet containing the list of CDs that R' needs to handle. Once R' receives it, it is ready to behave as the RP for those CDs. R also propagates a FIB add message to routers along the path to R' to enable the forwarding of messages that need to be multicast by R to R'. This also results in the intermediate routers reversing their ST entry so that R is in a subtree formed with R' as the root. This ensures all subscribers who are still attached to R continue to receive messages. At the end of this stage, published messages for the CDs will be sent to R' and R' forwards the multicast packets to R except for the new ST entries on R' and for those on the path between the two RPs. It is easy to prove that no packets would be missed during the dissemination (half an RTT between R and R'): all the packets that arrive before the new R' is created will be multicast via R; all the packets arrive after the creation of R' will be sent to R', and multicasted via R'; packets that travel between R and R' will be redirected to R' due to the change in FIB, and will finally be multicast by R'.

Propagate new RP information: In the final stage, R' becomes an RP independent of R. First R' propagates FIB with the related CDs throughout the network, so that packets that arrive via R can be forwarded directly according to the network topology. A shortest path tree is created. On receiving an FIB change packet, a normal router r in the network will change the ST information according to the new FIB (in COPSS, ST is built on the reverse FIB path). However, an immediate change will cause packet loss during this period, since upstream routers do not have the same ST information. Our solution to this is similar to that adopted in [17]. We put the new ST into a pending ST and send a join message upstream. During this period, multicast packets continue to arrive via the original upstream routers. On receiving a join message downstream, a normal router will check if it is already in the multicast tree, there will be 3 cases: 1) not in the tree, it will add a pending ST and send a join packet upstream; 2) already in the tree, it will send a confirm packet back; and 3) in the pending ST, it will wait for the confirm upstream and send a leave to the original upstream router and pass confirm downstream. In this step, we can see that the router does not leave the original ST branch until it is added to a new ST branch. So no packet will be missed by the subscribers during this stage.

V. Experimental Evaluation

We evaluate the performance of G-COPSS and compare 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 felt by all the candidate solutions. The simulator, which is parameterized by microbenchmark results, uses real 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. 3a. 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 could change. A player is able to see and modify these objects depending on his location in the game and the hierarchy of the area he belongs to as defined in §III.

A. *Microbenchmarking*

We implemented and microbenchmarked G-COPSS on our lab testbed to compare the three approaches, using the game map depicted in Fig. 3a. 62 players participated in the game with 2 players in every 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, we generated a 1-minute trace composed of publish records like $\{time, playerName, CD, Content\}$. In this timeframe, 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.

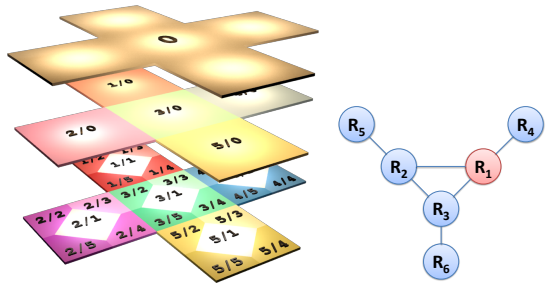
In our test-bed, we use 6 Optiplex 755 SF systems as routers, running Ubuntu 10.10 and using the public domain CCNx code version 0.4.0. One PowerEdge T300 system running Ubuntu Server 11.10 is used to run all the 62 clients and the server in an IP-server scenario. The network topology is shown in Fig. 3b. 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 [18] and assumes that players are managed using the system proposed in ACT [19], 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, we let a player 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, we send a response every t ms. All the updates within the t ms will be put into one packet. There is a tradeoff: if we set t large enough, more updates are included which saves some bandwidth, but the update latency will be longer. If we set t too small, players can see the updates immediately but incur a lot of overhead.

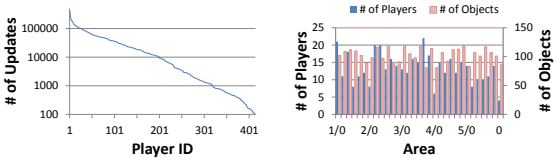
For a server-based solution, we created a data packet containing the following fields: source address, destination address and payload. All the machines use an application-level forwarding engine for a fair comparison, forwarding packets based on the destination address. All players send updates to a server. On receiving an update, the server decides whom to send it to, and sends the update using unicast.

Microbenchmark Result

We show the update latency in Fig. 4. In the evaluation, the average update latency experienced by players in G-COPSS is $8.51ms$, compared to $25.52ms$ in the IP server-based solution. The latency here is smaller than the real world situation because we do not incur much latency on the wire; almost all the latency caused is due to the packet processing at the routers. We can observe that all players experience an update latency smaller than $55ms$ in G-COPSS whereas about 8% of players experience an update latency over $55ms$ in the IP server case. The server becomes



(a) Game map. (b) Topology in benchmark.



(c) # of updates per player. (d) # of players/objects per area.

Figure 3. **Experiment Setup.**

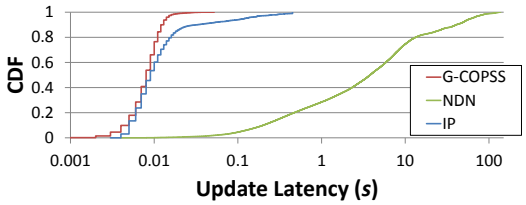


Figure 4. **Update Latency CDF of G-COPSS, NDN and IP Server.**

the bottleneck of message dissemination although IP routers are much more efficient than the G-COPSS routers.

For the NDN solution, even with the optimizations provided, we find that the average update latency is over $12s$ and the largest latency experienced reaches $150s$. 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, we conjecture that the VoCCN based NDN solution might not work for large-scale pub/sub systems like games.

B. *Large Scale Trace-Driven Experiments*

To further evaluate the performance of G-COPSS in large scale scenarios, we developed a simulator, with a game model derived from the data trace of the Counter-Strike (CS) game [20]. Since the query/response-based NDN (which we already optimized based on the state of the art [18], [19]) still experiences large delays (as shown in §V-A), the NDN solution was omitted in the rest of the evaluation and G-COPSS is compared to the IP server-based solution.

Data Trace and Experimental Setup

We use a CS data trace obtained during the peak period of one day [20]. It consists of a Wireshark 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 packets headers captured did not contain game-related information, and is for a server based game, we performed the following operations to filter out packets that represent a decentralized game: 1) discarded all packets sent from the server since G-COPSS does not require a server, 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) we used each unique address 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. 3c shows the number of updates performed by the different players. The 414 unique players share a global game-map shown in Fig. 3a with each area containing between 4 and 20 players. Furthermore, as shown in Fig. 3d, each area is further divided into objects ranging from 80 to 120.

We use the Rocketfuel [21] backbone topology (id=3967) for the core routers. We put 200 edge routers on the 79 core routers, and each core router is connected to 1-3 edge router(s). We uniformly distributed the 414 players 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 $10ms$.

Update Message Dissemination for Online players

In this section, we evaluate the performance of G-COPSS in disseminating game updates compared to the traditional approach of using an IP network with servers. Initially, we assume a case where all the players remain in their areas of interest and the events (from the trace) generated relate to actions within each AoI.

Traffic concentration elimination: We emulate the playing of the game using the first 100,000 update packets from the event trace to evaluate the average latency incurred in delivering an update from the publisher (player performing the update) to all the other subscribers (players subscribed to the CDs) for different number of RPs and servers. The average update frequency observed in the event trace is about $2.40ms$. In our simulations, an RP's processing time (including FIB lookup, packet decapsulation and ST lookup) is set to $3.3ms$ (based on our previous benchmark measurements),and the server processing time is around $6ms$ (based on our microbenchmark result, factoring in some

additional processing for other game related function like location translation and collision detection.)

Table I shows that the update latency for both the server and the G-COPSS approach. In the case of 1 or 2 RPs, it is high due congestion at the RPs. Adding another RP mitigated the congestion: using 3 or more RPs, no congestion is observed. Fig. 5a shows the minimum, maximum and average update latencies of every update in the 3-RP case. We observe that the update latency is below 1/5 second, well below the generally considered acceptable latency for such games (i.e., between 300ms and 1 second [22]).

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, we can see that multicast in G-COPSS reduces the network traffic significantly.

We then looked into detail for the congested cases. With 2-RPs, the G-COPSS latency shown in Fig. 5b, indicates that the RPs see congestion after $70,000$ packets and the latency increases dramatically. This is because of traffic concentration and queueing at the RP.

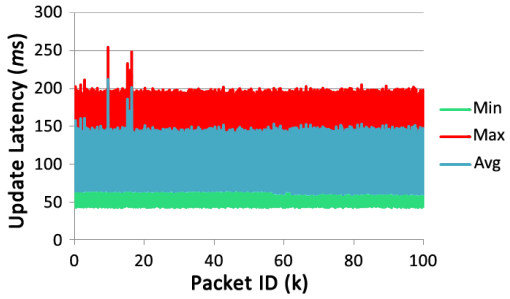
With 3 RPs or 3 Servers we examine scalability, varying the number of players in the network from 50, ..., 350, 400. Fig 6 shows the response latency and aggregate network load. Note that 3 servers are sufficient to support about 200 players playing at the same time. Fig 6a shows that the response latency 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. To overcome this, we implemented the automatic RP balancing and present simulation results. Fig. 5c shows the detail of the update latency in a 414 player test. The latency for the first $2,000$ packets are enlarged to show the further detail. The G-COPSS routers divided and moved the CDs to additional RPs twice when queuing was detected at an RP. The automated RP balancing algorithm chose 3 RPs, and as shown in Table I, the performance of the automatic RP balancing is close to the manually selected 3RP solution. With an increasing number of players, automatic RP balancing can further split the CDs to avoid congestion.

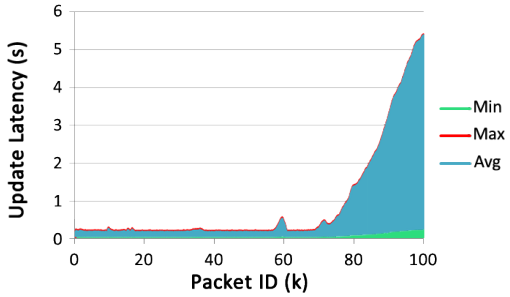
The benefit of hybrid-G-COPSS: We run the whole event trace to compare the performance of the IP-Server (with 6 servers), G-COPSS (6 RPs) and the hybrid-G-COPSS (6 IP multicast groups) approaches when there is

Table I
PERFORMANCE OF G-COPSS AND IP SERVER WITH DIFFERENT # OF RP(S)/SERVER(S) (414 PLAYERS)

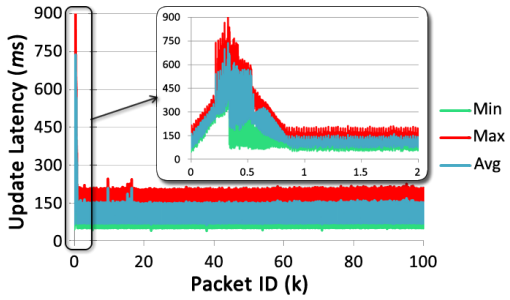
Type	# of RP/Server	Update Latency (ms)	Network Load (GB)
G-COPSS	1	47,680.29	5.59
	2	558.18	5.66
	3	94.89	5.58
	Auto	106.76	5.61
IP Server	1	249,679.70	9.74
	2	71,991.92	10.00
	3	21,448.17	9.62



(a) 3-RP Update Latency.



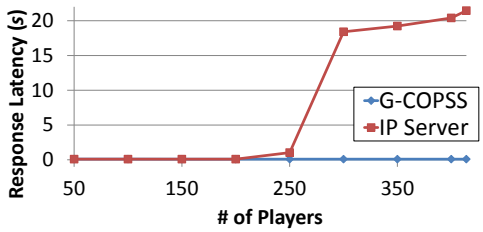
(b) 2-RP Update Latency.



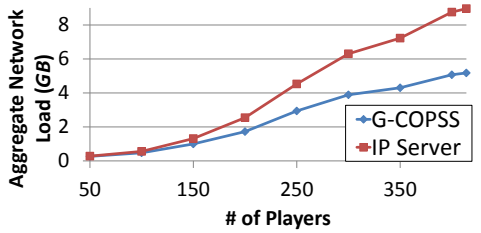
(c) Auto-balancing Update Latency.

Figure 5. **Traffic Concentration Elimination**

no congestion. In Table II, we see that with the use of hierarchical CD based multicast, G-COPSS (without the hybrid mode) places the least network load among the three solutions due to its use of multicast in a content-centric manner all along the path. Hybrid-G-COPSS performs the best in terms of update latency but introduces some more



(a) Response Latency.



(b) Aggregate Network Load.

Figure 6. **Performance of G-COPSS and IP server with different # of players. (3 RPs/Servers)**

Table II
PERFORMANCE OF IP-SERVER (6 SERVERS), G-COPSS (6 RPS) AND HYBRID-G-COPSS (6 IP MULTICAST GROUPS)

Type	Update Latency (ms)	Network Load (GB)
IP Server	90.17	1,046.62
G-COPSS	104.41	594.09
hybrid-G-COPSS	78.16	953.24

network load because multiple CDs are mapped onto one IP multicast group, which causes the dissemination of unwanted messages in the network. But the network load of hybrid-G-COPSS is still smaller than the server-based solution.

Message Dissemination for Players Moving

We evaluate 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, we did our best to emulate the activity that is likely to be generated in a real game. We perform the following operations to the data trace: 1) every player moves after an interval ranging from $5min$ to $35min$, 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, we further categorize the player movements into 6 types: 1) To a lower layer: E.g. $1/0 \rightarrow 1/1$ (plane landing). No download is required since he has the view already, 2) 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, 3) 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, 4) 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. 5) 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, 6) 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.

We uniformly assign updates of a player to the objects he can see 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 a snapshot of the area he has moved into. To model the size of the area (since the objects in the area could have been updated), we assume 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:

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

where $size(upd_i)$ stands for the size of ith update packet. For our trace, we set $\Delta = 0.95$. Therefore, the size of objects at the last version (at the end of the simulation) ranges from 579 to $1,074$ bytes.

In Table III, we compare the two solutions proposed for efficiently disseminating the current snapshot to players who have just arrived at a new area. We measure the convergence time as the time it takes for the player to receive an update after he has moved into the area. There are 3 brokers to disseminate the snapshot information.

We observe that the players in the cyclic multicast based solution see an average of $851ms$ whereas the players using the query response (QR) based solutions with a pipelining window size of 15 face an average of $2,600ms$. We can see that pipelining in QR solution optimizes the performance when we increase the window size from 5 to 15. However, there is no further benefit for a higher window size beyond 15, as we have approached the size of the pipe. Moreover, the broker might be the bottleneck in a QR based solution, as the number of players moving increases.

We observe that the convergence time grows (sub) linearly with the CD count. In cyclic multicast, a player can get the snapshot within 4 seconds even if 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

Table III
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 [different 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)

objects could begin to show first and the finer details could be downloaded after that.

We observe 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. However, we argue that neither of these two solutions is optimal. The control overhead of QR and the wasted transmissions in cyclic multicast could be further optimized using existing but better multimedia dissemination solutions (our ongoing work).

VI. SUMMARY

This work presents G-COPSS that functions as an efficient decentralized communication infrastructure for a gaming environment by leveraging the advantages of a content-centric network. We have implemented G-COPSS both on a testbed and a simulator. We obtained the micro-benchmark results and compared the performance of G-COPSS (over 60 players on the testbed) to that of an IP-server and NDN based approaches. We also show, using trace driven simulations for a system (over 400 players) that G-COPSS is able to outperform a pure IP-server based gaming environment in regards to aggregated network traffic and update latency. We also outperform the NDN approach, which depends on a query-response model for such a gaming application. We are currently working on algorithms for improving RP selection and creating a gaming environment that uses G-COPSS features to acquire real user behavior data and gaming traces.

REFERENCES

[1] V. Jacobson, D. K. Smetters *et al.*, "Networking named content," in *CoNEXT*, 2009.

[2] J. Chen, M. Arumaithurai *et al.*, "COPSS: An Efficient Content Oriented Publish/Subscribe System," in *ANCS*, 2011.

[3] W.-c. Feng, F. Chang *et al.*, "A traffic characterization of popular on-line games," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 488–500, 2005.

[4] S. Kumar, J. Chhugani *et al.*, "Second life and the new generation of virtual worlds," *IEEE Computer*, vol. 41, no. 9, pp. 46–53, 2008.

[5] F. Stenio, K. Carlos *et al.*, "Traffic Analysis Beyond This World: the Case of Second Life," in *NOSSDAV*, 2007.

[6] M. Varvello, S. Ferrari *et al.*, "Exploring Second Life," *IEEE/ACM Transactions of Networking*, vol. 19, no. 1, pp. 80–91, 2010.

[7] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," in *Proc. Int. Conf. Parallel & Distributed Techniques & Applications (PDPTA)*, 2003.

[8] A. Bharambe, J. Pang *et al.*, "Colyseus: A distributed architecture for online multiplayer games," in *NSDI*, 2006.

[9] S.-Y. Hu, J.-F. Chen *et al.*, "VON: A Scalable Peer-to-Peer Network for Virtual Environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, 2006.

[10] A. Bharambe, J. R. Douceur *et al.*, "Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games," in *Sigcomm*, 2008.

[11] M. Varvello, S. Ferrari *et al.*, "A Distributed Avatar Management for Second Life," in *NetGames*, 2009.

[12] M. Varvello, C. Diot *et al.*, "P2P Second Life: experimental validation using Kad," in *Infocom*, 2009.

[13] A. R. Bharambe, S. Rao *et al.*, "Mercury: a scalable publish-subscribe system for internet games," in *NetGames*, 2002.

[14] L. Zhang, D. Estrin *et al.*, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[15] W. Fenner, D. Srivastava *et al.*, "Xtreenet: Scalable overlay networks for xml content dissemination and querying," in *WCW*, 2005.

[16] F. Dabek, R. Cox *et al.*, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM*, 2004.

[17] M. Yuksel, K. K. Ramakrishnan *et al.*, "Cross-layer failure restoration of ip multicast with applications to iptv," *Computer Networks*, vol. 55, no. 9, pp. 2329–2351, 2011.

[18] V. Jacobson, D. K. Smetters *et al.*, "Voccn: voice-over content-centric networks," in *ReArch*, 2009.

[19] Z. Zhu, S. Wang *et al.*, "Act: audio conference tool over named data networking," in *ICN*, 2011.

[20] W. Feng, "On-line games," http://www.thefengs.com/wuchang/work/cstrike/.

[21] R. Mahajan, N. Spring *et al.*, "Inferring link weights using end-to-end measurements," in *IMW*, 2002.

[22] M. Claypool and K. Claypool, "Latency and Player Actions in Online Games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, 2006.

5 Name Based Multicast Congestion Control Framework

SAID: A Control Protocol for Scalable and Adaptive Information Dissemination in ICN

Jiachen Chen*†, Mayutan Arumaithurai†, Xiaoming Fu†, and K. K. Ramakrishnan‡
*WINLAB, Rutgers University, NJ, U.S.A. jiachen@winlab.rutgers.edu
†Institute of Computer Science, University of Göttingen, Germany. {arumaithurai,fu}@cs.uni-goettingen.de
‡University of California, Riverside, CA, U.S.A. kk@cs.ucr.edu

ABSTRACT

Information dissemination applications (video, news, social media, *etc.*) with large number of receivers need to be efficient but also have limited loss tolerance. The Information-Centric Networks (ICN) paradigm offers an alternative approach for reliably delivering data by naming content and exploiting data available at any intermediate point (*e.g.*, caches). However, receivers are often heterogeneous, with widely varying receive rates. When using existing ICN congestion control mechanisms with in-sequence delivery, a particularly thorny problem of receivers going *out-of-synch* results in inefficiency and unfairness with heterogeneous receivers. We argue that separating reliability from congestion control leads to more scalable, efficient and fair data dissemination, and propose SAID, a control protocol for Scalable and Adaptive Information Dissemination in ICN. To maximize the amount of data transmitted at the first attempt, receivers request *any next packet* (ANP) of a flow instead of next-in-sequence packet, independent of the provider's transmit rate. This allows providers to transmit at an application-efficient rate, without being limited by the slower receivers. SAID ensures reliable delivery to all receivers eventually, by cooperative repair, while preserving privacy without unduly trusting other receivers.

CCS Concepts

•Networks → Network protocol design; Transport protocols;

Keywords

ICN; Congestion Control; Reliability; Out-of-synch

1. INTRODUCTION

Large scale information dissemination applications like video streaming (YouTube, Netflix, *etc.*), online social networks (Facebook, Twitter, *etc.*) and news/entertainment (CNN, BBC, RSS feeds, *etc.*) have become common. Many of these applications have limited loss tolerance and depend on the network to provide efficient, fair and reliable content distribution. While IP multicast was designed for large-scale information dissemination, the inability to have an effective congestion control solution, especially in the presence of heterogeneous capacity to receivers, has been a limitation.

A key goal for publishers sending data at an application-efficient rate across the entire receiver population is to be not limited by the slower receivers. Due to the absence of a network layer mechanism to control the delivery rate at the receiver end, previous solutions have either sought to push all the data onto the path, overlooking congestion and unfairness and using end-end unicast recovery, or seek to slow down the sending rate to the slowest receiver [1]. Alternatively, the use of unicast, with the associated inefficiencies, has become the norm. The use of multicast at the application layer (*e.g.*, SCRIBE [2]) exploits TCP's congestion control mechanisms for ensuring loss-free delivery on an end-end basis. However, such solutions are network topology unaware and achieve lower efficiency (caused by the end-hosts replicating the packets rather than the routers) than what an effective multicast solution could be expected to achieve.

The advent of Information-Centric Networks (ICN) offers us an opportunity to take a fresh look at the potential solution approaches. Content-Centric Networks (CCN [3]) or Named Data Networking (NDN [4]) is a representative ICN approach. Although NDN does not mandate a congestion control mechanism, most of the proposed solutions [5–7] choose to use a TCP-like receiver-driven mechanism to limit the number of requests (window) 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 [8]. Compared to a sender-driven rate control approach, such a receiver-driven approach has the benefit of the consumer[1] being able to control the receive rate. When considering efficient large scale data dissemination where every piece of data is consumed by a large number of receivers, TCP-like mechanisms for receiver-driven multicast can have significant shortcomings, especially with the path to the receivers having different capacities. With such heterogeneity, a problem we observe is that of receivers being **out-of-synch** *even with optimal policies for managing the cache.* The out-of-synch problem can be briefly described as follows: NDN routers cache contents as they are forwarded. When there is temporal locality of requests from receivers, a router that has the cached content can respond. However, with receiver

ICN'16, September 26 - 28, 2016, Kyoto, Japan

ISBN 978-1-4503-4467-8/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2984356.2984370

[1] Consumer/receiver, and provider/sender are used interchangeably in this paper.

heterogeneity, the requests from receivers even for the same data items 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 content provider (via retransmissions) as a separate flow. These retransmissions will compete for the bottleneck bandwidth and the overall throughput can therefore dramatically reduce. This is a fundamental issue as long as there are heterogeneous receivers and routers with limited cache sizes. The problem can be exacerbated with scale, thus occurring even more often in the core of the network.

In this paper, we propose a control protocol for Scalable and Adaptive Information Dissemination (SAID) in ICN, a novel mechanism enabling large scale efficient data dissemination. We leverage the receiver-driven framework of NDN with enhancements to overcome the out-of-synch problem. SAID achieves efficiency by separating reliability from congestion control. While our design is framed in the context of NDN, we believe SAID can also be used by other multicast solutions. The contributions of this paper include:

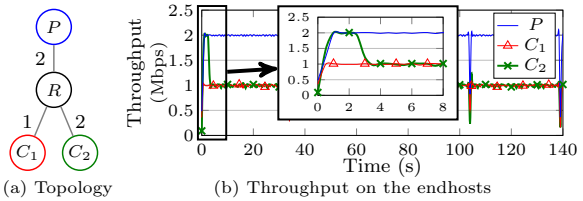
- An analytical and emulation-based study on the out-of-synch problem that shows it will occur in real networks even with large in-network caches as long as there is receiver heterogeneity (see §2);
- A new reliable multicast framework that seeks to maximize the useful throughput, by consumers requesting for *Any Next Packet* (see §3.1). Reliability is then achieved via a repair mechanism that leverages NDN's capability for receivers to request contents from any network node, while preserving privacy and data integrity (see §3.2);
- A receiver-driven congestion control mechanism tailored for ANP delivery that enables *each* receiver to obtain its fair[2] share of the bottleneck link while maintaining an application-efficient sending rate (see §4);
- Evaluations on our prototype and large scale simulations illustrate the benefit of SAID compared to solutions such as ICP and pgmcc (see §5).

2. PROBLEM WITH EXISTING CONGESTION CONTROL – OUT-OF-SYNCH

We first study the out-of-synch problem in NDN and show how it reduces the benefit of in-network caches and the use of pending interests. We demonstrate this with an emulation using CCNx 0.8.0 along with a congestion control mechanism similar to ICP [5]. We show through an analytical model that our observations on the out-of-synch problem is systemic with heterogeneous receivers, and should be expected even with other receiver-driven, in-sequence congestion control mechanisms.

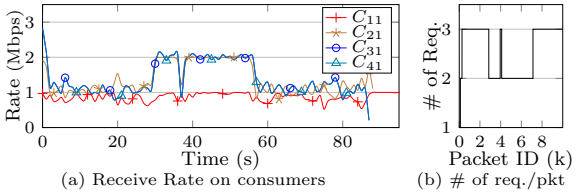
2.1 Demonstration of Out-of-Synch in An Emulated Scenario

To clearly demonstrate the out-of-synch problem and its cause, we use a simple emulation performed in Mini-CCNx. The network topology and the link rates (in $Mbps$) are shown in Fig. 1a. The latency on all the links is $2ms$. Router R has a 50 packets cache[3]. Consumers C_1 and C_2 start to



(a) Topology (b) Throughput on the endhosts

Fig. 1: Out-of-synch problem emulated in a simple topology.



(a) Receive Rate on consumers (b) # of req./pkt

Fig. 2: Out-of-synch: On larger dissemination tree.

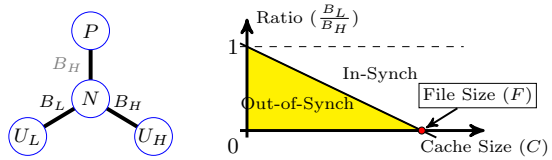
request the same content ($\sim 35MB$ in size, $8,965pkts$) from provider P at the same time. The throughput of the end hosts are shown in Fig. 1b. For the first 2.5 seconds, the PIT and the network cache benefit both receivers. Requests from C_1 either get aggregated or get a cache hit at R. Overall network throughput in this period 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, to be treated as a distinct 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 network throughput reduces to $2Mbps$, thus underutilizing the bandwidth by 33%. For the entire transfer, we observed that $< 2\%$ of the requests from C_1 see a cache hit.

Similar results occur in more complex topologies (*e.g.*, Fig. 10 without the dotted links). The consumers C_{11} – C_{41} start to request the same content from provider P_1 at the same time (C_{51} is not active in this emulation). The cache size is $100pkts$ and the content size is $10,000$ packets. The receive rates at the consumers are shown in Fig. 2a and the # of requests (transmissions) per packet observed by P_1 is shown in Fig. 2b. We can see that, similar to the previous simpler case, the consumers once again get out-of-synch soon after the transmission sequence starts. The # of transmissions increase and the receive rate drops. For the first 30 seconds, the aggregate throughput is only around $4Mbps$ (the ideal throughput is $9Mbps$). During the intervals 30–36 and 40–55 seconds, the faster receivers get in-synch again, due to the randomness in the network and cache occupancy. The packet sequences between $2,800$–$3,950$ and $4,100$–$7,100$ are transmitted twice and the throughput of C_{21} through C_{41} increases during these two time periods. But even during this time, the aggregate throughput can only reach $7Mbps$ rather than the $9Mbps$ achievable in the ideal case.

Through these emulations, we 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. This is the occurrence of 'out-of-synch' problem. When the slower receivers

[2]In this paper, we define fairness as max-min fairness based on the link capacity, which is not affected by the content popularity.

[3]We use a relatively small cache size to quickly demonstrate the out-of-synch problem. However, this is fundamental and occurs even with much bigger caches. Please see §2.2 for the relationship between the cache size and the heterogeneity allowed among the receivers.



(a) Single-branching model (b) Relationship between out-of-synch and request rate, cache size and flow size

Fig. 3: Out-of-synch in a single-branching model.

re-issue requests, these requests are seen as 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-synch problem can happen even when all the receivers start their requests for the sequence of data packets at the same time, 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.

2.2 Analysis on Out-of-Synch Occurrence

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 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. We realize that almost all the existing congestion control protocol proposed for NDN fall into this category.

To show the universality of the problem, we generalize the model for congestion control 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. We analyze the maximum heterogeneity that can be supported given a certain cache and flow size while the receivers still remain in-synch till the end of the flow. Since we are focusing on a single flow, we also 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.

We start with a more precise definition of out-of-synch.

Definition 2.1 (Out-of-synch)**.** *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-synch occurs when the difference in the amount of bytes received (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.*

We then use a single-branch model to demonstrate the relationship among the cache size, flow size and receiver heterogeneity in a simple scenario.

Lemma 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 range $[B_L, B_H]$, to avoid out-of-synch, the following condition should hold,* i.e.*:*

$$B_L/B_H \geq 1 - C/F, \tag{1}$$

where F is the size of the flow.

Proof. According to Def. 2.1, to avoid out-of-synch at N, it is sufficient to consider 2 immediate downstream links with the largest and smallest available bandwidth. That is, we can consider the single-branching topology in Fig. 3a 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-synch, 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:

$$t = F/B_H.$$

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

$$G = (B_H - B_L) \times t = (1 - B_L/B_H) \times F$$

According to Def. 2.1, to keep the consumers in-synch, we need:

$$G = (1 - B_L/B_H) \times F \leq C. \tag{2}$$

Equation (1) is equivalent to (2). □

The requirement for clients in-synch (equation 1) is presented in Fig. 3b. We can see that the requirement for being in-synch cannot be satisfied when the heterogeneity (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 form a tree structure rooted at the data provider, we study the in-synch requirements in a k level tree.

Theorem 2.1. *For a dissemination tree with k levels and every intermediate router having cache size C, all the receivers will be in-synch only when the available bandwidth between the fastest receiver and the slowest receiver follow:*

$$B_L/B_H \geq (1 - C/F)^k. \tag{3}$$

Proof. We prove this theorem by contradiction. Suppose that the request rates of the highest and the lowest receivers satisfy

$$B_L/B_H < (1 - C/F)^k, \tag{4}$$

and all the receivers are in-synch.

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 2.1, we have

$$B_L/B_{H,t} \geq 1 - C/F. \tag{5}$$

According to (4) and (5), it follows that

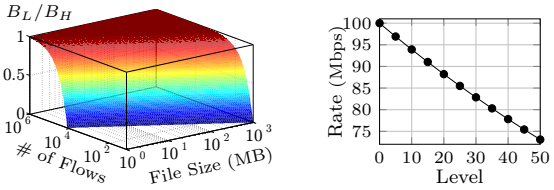
$$B_{H,t} < B_H \times (1 - C/F)^{k-1}. \tag{6}$$

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 2.1, we have

$$B_{H,t}/B_{H,t-1} \geq B_{L,t-1}/B_{H,t-1} \geq 1 - C/F. \tag{7}$$

According to (6) and (7), it follows that

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



(a) Relationship among out-of-synch criteria in real router (b) Minimum allowed download rate *vs.* hierarchical level

Fig. 4: Requirements to keep consumers in-synch.

By the similar argument, we can show that

$$B_{H,t-2} < B_H \times (1 - C/F)^{k-3},$$

$$B_{H,t-3} < B_H \times (1 - C/F)^{k-4},$$

$$\ldots$$

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

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 reach a contradiction and the proof is completed. □

From Theorem 2.1 we can see that with growth of k, 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 we show that out-of-synch is difficult to avoid in a NDN router deployment.

Remark. *The problem of receivers going out-of-synch persists with receiver-driven feedback-based in-sequence congestion control protocols with 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 [9], NDN requires a 25TB cache for a 50% hit rate on Youtube data and 175TB cache for a 50% hit rate on BitTorrent data. However, [10] suggests that deployable NDN routers (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, we assume that concurrent flows in a router share the 100Gb cache size equally. The relationship between the bandwidth ratio (B_L/B_H), file size (F) and number of flows is shown in Fig. 4a. The intersection of the curve with XY plane is where C=F (# of flows=$100Gb/F$). The region behind the curve represents the region receivers are out-of-synch. Note that both the X- and Y-axis are log-scale, which means the area when the receivers are in-synch is just a very small portion of the overall region.

Although a core router might have relatively larger 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:

$$B_L/B_H \geq 1 - (100Gb/100k)/160Mb = 0.99375$$

If B_H can reach 100Mbps, B_L should be ≥99.3Mbps. When we apply the hierarchical tree model in Theorem 2.1, the minimum download rate *vs.* level is plotted in Fig. 4b. Even with 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 multiplexed on a given link.

In this model, we assume that all the consumers start to request a same piece of content at the same time. We argue that this assumption is reasonable, especially considering the set of popular content items. With a reasonably large library of content accessed by consumers, the working set of popular content across consumers can potentially exceed the cache size at routers. It has been observed in [11] that there is a 'fat middle' with a large number of content items having multiple users accessing content at *around the same time*, even when unicast is used for VoD delivery (as is currently the case in the Internet with delivery of video over HTTP, similar to the request/response paradigm of several ICN protocols). Therefore, the best case scenario for normal query/response (sequence-specific requests) to avoid out-of-synch is when all the consumers request for the content at the same time (our assumption) and it would only worsen with the arrival of late-comers. Even with popular content, as larger numbers of heterogeneous users access the content, it will still result in cycling the cache (the symptom of 'out-of-synch') since the cache size at a router is likely to be much smaller compared to the working set size.

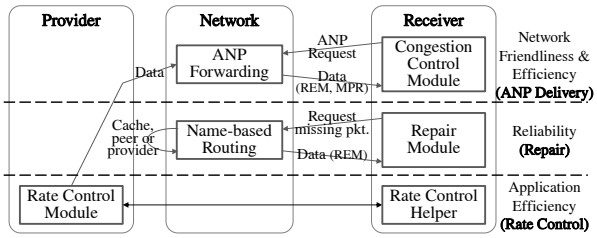


Fig. 5: SAID overview.

3. SAID FRAMEWORK

To address the out-of-synch issue at its root cause, SAID slightly modifies the request paradigm – from request for a specific-sequence to a request for *any-next packet* (ANP). The consumer-driven, network-assisted ANP delivery is the core of the SAID architecture (see Fig. 5). It intentionally decouples reliability from congestion control because *1*) the need for reliability and in-order delivery varies across applications, and *2*) the solutions that enforce in-sequence reliable delivery result in overheads (*e.g.*, receivers being out-of-synch, with NDN/ICP) or inefficiency (sending at the slowest receiver's rate, with pgmcc). The new paradigm seeks to retain consumers in-synch so as to maximize the utility of the packets sent at the first attempt. ANP delivery also ensures fairness on each branch of the dissemination tree, independent of the content provider's sending rate.

To satisfy applications that have different requirements on reliability and efficiency, we add application-specific modules to make SAID a general framework for multiparty information delivery in the ICN/NDN context. The repair mechanism (middle row in Fig. 5) fully utilizes name-based routing to help recover missing packets from the original provider, an in-network cache or neighbors, without sacrificing privacy or trust. For applications that have elastic bandwidth requirements (*e.g.*, file delivery), the rate adaptation mechanism (bottom row in Fig. 5) allows applications to achieve a tradeoff between network load and session completion time.

3.1 ANP Delivery Model

To overcome the case of heterogeneous receivers getting out-of-synch, and maximize the likelihood of delivery of a packet in the first transmission attempt, we propose a slight modification to the existing communication model of receivers asking for a sequence-specific packet (as is currently done by NDN and even TCP, *etc.*). Here, receivers ask for *any* subsequent incoming packet. Therefore, while con-

sumers with higher available bandwidth (referred to as faster consumers) can receive sequences 0, 1, 2, . . . , 100, consumers with lower available bandwidth (slower consumers) might get 0, 3, 6, . . . , 100. Further, consumers that join the group later than the others (we refer to them as late-comers) might get 66, 67, . . . , 100. We call this kind of delivery as in-synch delivery since all the consumers reach packet 100 around the same time. While it is true that the slower consumers still have to get packets 1, 2, 4, 5, . . . , and the late-comers have to get packets 0, 1, . . . , 65 (through repair, see §3.2), SAID ensures that in these cases, 1/3 of the packets are delivered in the first attempt, unlike sequence-specific requests that require retransmission of many more packets when receivers are out-of-synch. Thus we avoid cycling the cache.

An ANP request packet looks much like a normal Interest packet, except that the Name in the packet is the prefix of the file/flow instead of a sequence-specific content (*i.e.*, such as a normal Interest which carries both prefix and segment ID). To retain the current framework of NDN as much as possible, we reuse the existing Interest packet, since the name in the packet can be used for both sequence-specific and ANP requests. To distinguish between the two requests, we place an extra field (flag) into the packet header.

A pure network-based congestion control solution that divides the outgoing link bandwidth fairly for each flow going to the next hop is unable to accommodate indications of reduced demand from flows that have a limit downstream, especially with heterogeneous receiver bandwidths. This results in inefficiency, as was suggested by [12], and we show it in §5.1. Therefore, SAID chooses to use an end-system assisted (receiver-driven) solution to avoid such inefficiency. First, SAID retains the 'flow balance' property suggested in NDN, in which every ANP request packet will have at most one Data packet returned. Receivers then maintain a window of unsatisfied ANP requests, reflecting the maximum number of packets that can be in flight towards this receiver. We adapt this window using an AIMD scheme, just like TCP, thus finding the capacity of the bottleneck in the sender-receiver path.

The forwarding rules for ANP requests also need to be modified. One critical difference is that the ANP requests for a given flow would carry the same flow name (prefix). Therefore, instead of creating an entry in the PIT for each sequence-specific request, we add a counter in the PIT entries (called the Pending Request counter, PR) to keep track of the (maximum) window size of each downstream node. On receiving an ANP request, the router would: *1*) Skip the usual check of the Content Store to avoid transmitting duplicate packets. *2*) Check in the PIT if there is already an entry with the same name (prefix) and incoming face. If the entry does not exist in the PIT, add one entry with PR=1, otherwise, increment the PR by 1. *3*) If the PR value in the last step (before increment) was already the maximum PR for that prefix among all the incoming faces, an ANP request will be sent upstream based on a FIB lookup. With this mechanism, each consumer can propagate its window size up until its branching point[4], and the maximum window size is accumulated at the provider. However, the provider does not have to send at the rate of the maximum window (equivalent to aligning the sending rate at the fastest consumer), since it might result in more traffic generated because of repair. We describe how the provider should adapt its sending rate in §3.3. This modification to the forwarding rule for ANP requests is more space-efficient than the sequence-specific requests. For n requests of the same flow, SAID only needs 1 PIT entry and 1 counter compared to n separate PIT entries with the existing NDN approach.

On receiving a Data packet of the flow (which satisfies the ANP request), the router would: *1*) Save the packet in the Content Store for further (sequence-specific requests). *2*) Find all the PIT entries the Data can satisfy and decide the outgoing faces. For ANP request entries, the router would decrease the PR of each outgoing face by 1, and discard the PIT entry when PR becomes 0. *3*) Replicate and forward the Data to all the outgoing faces determined in the previous step. With this subtle change, receivers in SAID can receive data without getting out-of-synch, thus enhancing network efficiency. Since the window size of each consumer is propagated up to its branching point, we can ensure that the consumers would *1*) receive the flow at the rate indicated by the window size if that is smaller than the sending rate of the provider, or else *2*) receive at the sending rate of the provider. In §4, we describe how we control the window size on the consumer to reflect the fair share on the bottleneck between each consumer and its branching point, and thus ensure the max-min fairness in the network.

3.2 Efficient Repair

For applications that need different degrees of reliability, we depend on the application- (or transport-) layer interface to retrieve missing packets. SAID allows the application to decide *which packets* to repair and *when* the repair packets are delivered, while the network helps to find *where* copies of the needed packets. For applications like live-streaming, they can start the repair during the time period when the flow's new packets are being sent for the first time, although that might aggravate congestion at the bottleneck. The benefit is that the user experience (in terms of stall time) will be improved. This is a tradeoff that the application has to decide, based on policy and/or periodically based on perceived QoE. QoE is also influenced by repair being made through other paths. For applications like file download, repair can start after an ANP round to minimize the network congestion while new packets are flowing. Applications can also decide which packets need to be repaired, *i.e.* fetched. *E.g.*, applications using Raptor Codes [13] can reduce the amount of repair since the content of some missing packets might be recovered by ANP delivery of future packets.

Once an application decides to repair a packet, SAID uses NDN's sequence-specific (normal) data retrieval capability to retrieve the missing packets from the network. Since in-network caches may have stored packets as they are forwarded, some of the repair requests can be satisfied by the caches. However, if the required packets are no longer in the cache (have been replaced), consumers can still get the packets from the original provider. However, this repair has the potential to aggravate the congestion at a bottleneck link in the path from the provider to the receiver. Therefore, we propose a peer repair mechanism (described below) that can potentially mitigate the impact of repair on an already congested link by making use of other paths.

With SAID, receivers who have received (some or all of the packets) the flow can propagate FIB entries for the pre-

[4]We define the branching point of consumer C (BR_C) as the router closest to C that has another consumer that has bandwidth higher than C. The branching point for the fastest consumer is up at the first hop router of the provider.

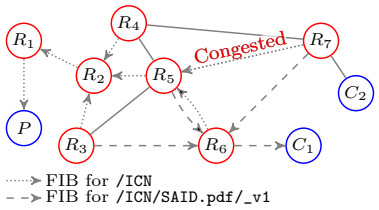


Fig. 6: Example of different repairs.

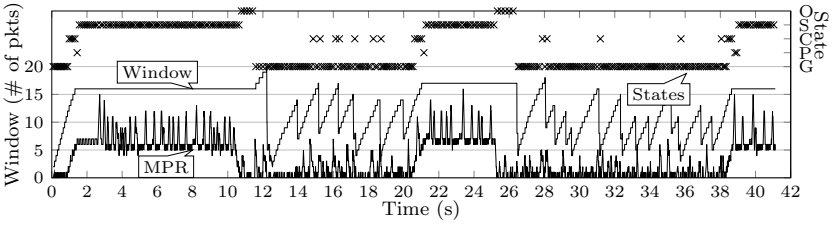


Fig. 7: Result of window control at the receiver using state machine.

fix of the flow over a limited number of hops. *E.g.*, in Fig. 6, after receiving the packets, C_1 propagates prefix `/ICN/SAID.pdf/_v1` over a 2 hop range. When C_2 requests for `/ICN/SAID.pdf/_v1/_s20`, R_7 will forward the request to R_6 and will eventually get a response from C_1. The repair can then bypass the congested link R_5–R_7 and still increase the overall useful throughput to C_2. Note that propagating FIB and the repair requests do not carry the identity of the consumers. Thus, the solution can fully utilize the benefit of NDN without sacrificing privacy and trust (the receivers and senders do not have to reveal their identity to one another). We believe multiple paths will continue to exist, just as we see in current networks [14]. In such a case, repair requests may be forwarded along the path that data was received successfully before, so as to increase the probability of receiving repair data (as proposed in [15]).

The simple mechanism we describe above might still be inefficient. We know that whenever C_1 propagates the prefix, routers will forward repair requests from nearby nodes to C_1 (instead of going upstream). Therefore, for efficiency, C_1 can only propagate the prefix when he receives *all* the packets in the flow, so as to be able to respond to the repair requests. Such a propagation is however useless for applications like VoD that needs repair while packets of the flow are still being delivered. An alternative solution for FIB propagation is for C_1 to propagate the exact ContentName of each packet he receives. Requests from slower receivers can then be redirected to C_1 immediately. This solution benefits applications that need 'in-flow' repair, but it places a large 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, we suggest that the data provider should group n packets into a 'chunk' and replace the `segmentID` in the ContentName with `chunkID/segmentID`. *E.g.*, if n=100, the name of the packet with segmentID=205 should be `/ICN/SAID.pdf/_v1/_c2/_s205`. This will not affect the basic solution since every packet still has a globally unique ContentName. Nonetheless, C_1 can propagate `/ICN/SAID.pdf/_v1/_c2` after receiving packets 200-299 (*i.e.*, FIB entry for the chunk) either via the initial transmission or via repair. FIB entries will be smaller and at the same time will enable other receivers to get repairs from peers much earlier. The chunk size can be specified by the data provider based on the timeliness requirement for the flow.

3.3 Provider Rate Adaptation

ANP delivery ensures the fairness of a flow independent of the transmit rate of the sender. However, different applications have different transmission rate requirements, and the sender needs to adjust the sending rate accordingly. Live streaming and conferencing applications need a certain (minimum) sending rate based on desired video/audio quality. The sender can simply send data at that required rate.

Other applications may have more flexibility, but might have a preference to either satisfy most of the receivers in the first round of delivery or seek to satisfy the fastest receivers (*e.g.*, BitTorrent). For such applications, the transmission rate of the provider is a tradeoff between the (network, provider) load and the content delivery completion time. A higher sending rate could shorten the receiving interval on the faster receivers, but also result in higher retransmission rate in the network to reliably deliver to the other receivers. A lower sending rate may attempt 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. Instead of specifying a constant sending rate, we allow the provider to pick an ACKer from among the receivers to pace the flow similar to pgmcc [1]. The ACKer would send an ACK (*e.g.*, in the form of an Interest) back to the sender for each Data it receives. A feedback loop would be established between the ACKer and the sender like TCP and therefore the rate of the sender is aligned to the available bandwidth of the ACKer. However, unlike pgmcc, the ACKer does not have to be the slowest receiver. Please refer to our technical report [16] for details on ACKer selection.

4. CONGESTION CONTROL FOR ANP

With SAID, since the sending rate is not controlled by the window of each receiver, the difference between the sending and receiving rate can result in the receiver missing some of the packets in the sequence (we use the term 'holes in the sequence'). With SAID, these holes should not be treated as an indication of congestion (as conventional TCP does with loss-based congestion indication), triggering a response from the receivers. Instead, SAID uses REM [17] in a modified manner to indicate congestion appropriately. We also make the important observation that because the sending rate is not controlled by the receivers' congestion windows, receivers with an available bandwidth higher than the transmission rate will not see congestion (or packet loss). Their window will continue to grow (not unlike an uncongested TCP flow's window). This growth in the window essentially causes a large pending request (PR) value at upstream routers. A large PR value will allow a large number of packets to be delivered in a burst towards the receiver (*e.g.*, when the provider decides to increase the transmit rate or when upstream contention on the path disappears). This large burst of Data that will be queued at the bottleneck router (with consequent increase in feedback delay) causes that router to indicate congestion (*e.g.*, REM) on a large burst of packets. This can cause the receiver to over-react and reduce the window too much, thereby ceding more than its fair share to a competing flow. Subsequently, the competing flow would have a large PR, and now in turn experience a similar burst (a 'bang-bang' effect). Although

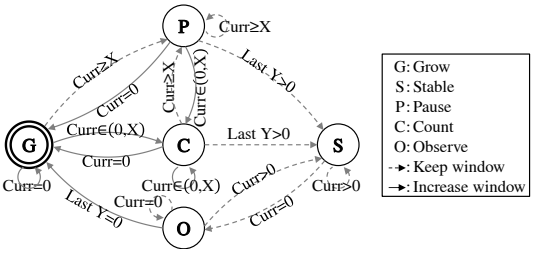


Fig. 8: State machine for receiver window increase.

the AIMD mechanism will eventually converge, the time it takes to converge will depend on the size of burst and feedback-delay. This effect therefore has the potential to under-utilize the network and/or cause unfairness between flows for 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. Because of these considerations, we develop additional mechanisms to achieve congestion control and fairness.

4.1 Identifying Congestion

Since holes in the sequence are no longer an indication of congestion, we depend on help from the network to indicate congestion through Active Queue Management (AQM) mechanisms. In particular, we use Random Early Detection (REM [17]) in our current implementation. The solution can also take advantage of other AQM mechanisms like CoDel [18] that marks the packets on seeing congestion.

With REM, a simple AIMD mechanism on the receiver side (additive increase on receiving data and multiplicative decrease on seeing marks) works acceptably, especially on receivers that have available bandwidth below the sending rate of the provider. The introduction of AQM into NDN can also simplify existing congestion control solutions and help avoid any unnecessary bufferbloat [19].

4.2 Handling Additive Increase

In order to avoid the 'bang-bang' effect and optimize network utilization, the optimal window size should be the bandwidth-delay product of the path between the receiver and its branching point. It can vary depending on the network status, and can vary over time. Therefore, we propose a dynamic mechanism that observes and reacts to the minimum pending request count (MPR) on the path, with a goal of keeping the window size close to the optimal value.

The network piggybacks the instantaneous MPR value with data packets forwarded to receivers. Note that different receivers can see different MPR values even when they receive the same data packet since MPR is set at the branching point. A receiver observes the MPRs over a window so as to: *1*) get the minimum MPR value in that window (MMPR), thus avoiding excessive pending requests in the path, *2*) ensure that MMPR is within a small value but larger than 0 to avoid all pending requests being consumed by a future short burst, and *3*) smooth out variations. To simplify the calculation, our approach decides whether the window should increase based only on MMPR.

We use a state machine (depicted in Fig. 8) to decide the growth of the window. When MMPR==0 (state G), it implies that all the requests are consumed by the Data (incoming Data rate is less than the bandwidth-delay product of the downstream path) and therefore the receiver should increase the window size. With the growth of the window size, the MMPR would grow above 0 (state C in figure). The receiver should keep increasing the window size until he sees MMPR>0 for Y times (state S), which means the window size is big enough and stable. During the increase phase, if the MMPR grows above X (state P), it implies that there is a sudden change in the network and that the window size might be big enough already. The receiver would then stop increasing the window size immediately and wait for the status to either go back to G, C (increase again) or reach S (stop increasing). MMPR might get back to 0 when the receiver is in state S due to the change in the network status, and the receiver should increase the window again. However, to prevent instability in the network, we decide to increase the window only after seeing MMPR=0 for Y times (state O). Note that no matter what state the consumer is in, once it receives a marked packet (indicating a congestion), it will reduce the window size by half, go back to state S and start increasing the window. According to our experiments (in both emulations and simulations), we find it reasonable to have X=Y=5. We see that with these values for X and Y, the mechanism limits the window size properly and avoids bursts when competitors arrive (the bang-bang effect).

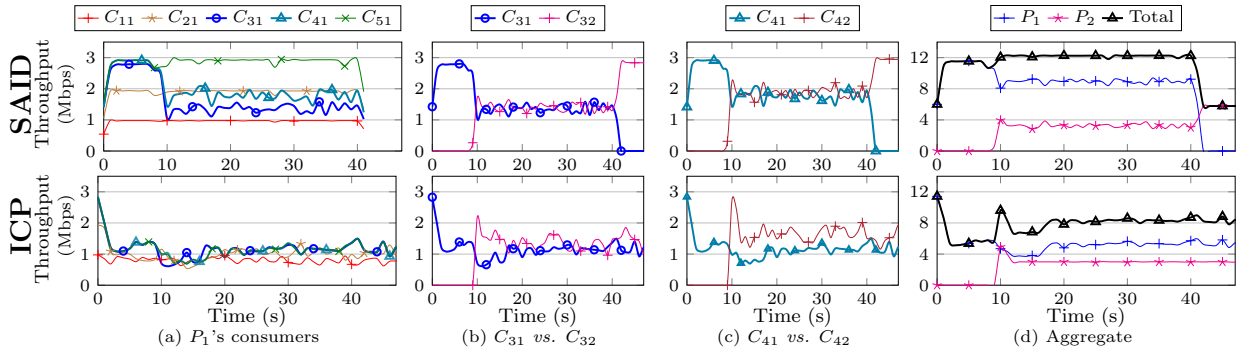
We use an example to trace the MMPR and the window increase decisions to aid in understanding how the scheme works with the values for X and Y mentioned above. The MPR, window size and the state transitions in Fig. 7 are shown for a receiver with $2Mbps$ bandwidth. Every mark represents the state of the state machine (corresponds to Fig. 8) at the end of each window. The MMPR grows in accordance with the window size after $1sec$. After 4 windows (decisions), the MMPR reaches 5 and the receiver stops increasing the window and MMPR stays around 5. Although the MMPR reduces below 5 for several windows at around $4sec$ (due to transient changes) the window size is not increased. When there is new contention at the bottleneck link (an additional competing flow) occurring between $10sec$ and $25sec$, MMPR drops to 0 and the receiver then begins to increase the window. When the receiver receives marked packets (REM) at $12sec$, it uses AIMD to respond to the congestion introduced by the competing flow. We show how this mechanism aids in achieving fairness in §5.1.

5. EVALUATION

We first present evaluations based on our prototype implementation. We then present simulation results for two applications (video streaming and large data file delivery) on a small custom topology (shown in Fig. 10) as well as a large RocketFuel Topology [20] with heterogeneous receiver bandwidths in our custom simulation environment (used in our previous work [21]). We compare SAID with *1*) An ICP [5] solution enhanced to use REM (so as to provide a reasonable comparison with SAID); and *2*) pgmcc [1], a window-based multicast congestion control protocol which aligns the data sender's transmission rate to the slowest receiver.

5.1 Emulation of ANP Delivery

We implemented the core components: *1*) ANP delivery (described in §3.1), and *2*) congestion control (described in §4) in Linux. The modified forwarding engine is implemented in user space and packets are encapsulated in UDP, as with CCNx. Our testbed consists of SAID enabled routers, each running on a single machine (CCNx 0.4). All the consumers and providers are run on a single machine so



(a) P_1's consumers (b) C_{31} *vs.* C_{32} (c) C_{41} *vs.* C_{42} (d) Aggregate

Fig. 9: Emulation result in competition scenario (using topology in Fig. 10, X=10$Mbps$).

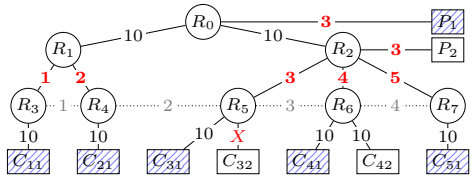


Fig. 10: Dissemination tree topology (bandwidth in $Mbps$).

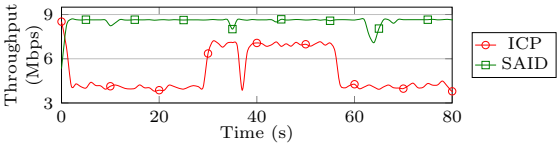


Fig. 11: Aggregate throughput in 4-consumer scenario.

as to have a synchronized clock for accurate result collection. The baseline topology consists of 8 routers (R_0→R_7), 7 consumers (C_{ij}) and 2 providers (P_1, P_2) as illustrated in Fig. 10. Per link latency is 2ms and the numbers on the links represent link bandwidth in Mbps. The bottleneck bandwidth for the consumers are marked in bold red.

• Efficiency of ANP Delivery Model

Fig. 11 illustrates the aggregate throughput achieved by SAID and ICP in the presence of 4 consumers. pgmcc aligns to the slowest subscriber, and its results are easily obtained without needing simulations[5]. The experimental setup was used in §2.2 with consumers C_{11}, C_{21}, C_{31} and C_{41}, having bottleneck bandwidths of 1, 2, 3 and 4 Mbps respectively and the content provider sending at 3Mbps. SAID is able to achieve an aggregate rate close to the maximum achievable throughput (1Mbps (C_{11}) + 2Mbps (C_{21}) + 3Mbps * 2 (C_{31}, C_{41})) of 9Mbps. ICP's maximum throughput is about 4Mbps. With ICP, receivers go out-of-synch and compete with one another on the link between P_1 and R_0. pgmcc also achieves around 4Mbps (1Mbps * 4) throughput.

• Fairness in the Presence of Competition

The experimental setup is P_1 with 5 consumers (named C_{i1} in Fig. 10) and P_2 has 2 consumers (C_{32}, C_{42}). To evaluate SAID's fairness when there is a competing flow from another provider (P_2), we configure P_2 to start sending packets after approximately 10s, with a sending rate of 3Mbps (*i.e.*, as fast as it can) and study its influence in the network under an extreme case. C_{31} and C_{32} are competing for the bottleneck bandwidth between R_2 and R_5 (3Mbps) and C_{41} and C_{42} are competing for the bottleneck bandwidth between R_2 and R_6 (4Mbps). The bandwidth between R_5 and C_{32}, *i.e.* X, is set to 10Mbps so that the link between R_2 and R_5 is the bottleneck link.

Fig. 9 illustrates the throughput achieved at each receiver. As in Fig. 11, we observe in Fig. 9a that in SAID, each of P_1's receiver is able to receive at the bottleneck capacity until 10s. Once the competing flow arrives, the receivers in SAID are able to receive at their *(statistical) fair share* of the bottleneck bandwidth. In ICP, upon the arrival of a competing flow, the throughput achieved by each receiver drops significantly, from an average of 4Mbps to just 1Mbps.

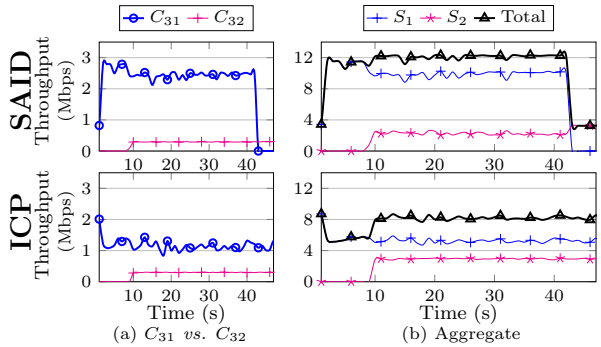
Fig. 9b–9c show that in SAID, both C_{31} and C_{41} are able to receive at their individual fair share (1.5Mbps and 2Mbps) in the presence of competition from P_2's flow. On the other hand, with ICP, the consumer's of both P_1 and P_2 are not able to fully utilize their fair share of the bottleneck bandwidth, resulting in an under-utilization of the bottleneck bandwidths. This is due to the fact that both C_{31} and C_{41} are in fact competing with other consumer's from P_1 in the P_1 to R_0 link and C_{32} and C_{42} are competing with each other in the P_2 to R_0 link, due to the out-of-synch phenomena. The peak at the start of the P_2 flows in ICP, illustrate that for a short period, flows to C_{32} and C_{42} are in-synch.

In Fig. 9d, the aggregate throughput achieved by P_1's and P_2's consumers is close to the maximum achievable throughput with SAID. Till the first 10s, when only P_1 is active and the total achievable throughput of the network is 12Mbps, we observe that SAID is able to get closer to this rate. When P_2 starts at 10s, the ideal achievable aggregate throughput increases to 13Mbps. SAID is able to get closer to this ideal rate. With ICP however, it is approximately 5Mbps before 10s and 8Mbps afterwards. In pgmcc, in the absence of a competitor (*i.e.*, before 10s), P_1 aligns to C_{11} (1mbps) and therefore the total throughput is 5Mbps (all its 5 consumers receive at 1Mbps each). In the presence of a competitor P_2 (after 10s) who is sending at 2Mbps (aligned to C_{32}), the total throughput will increase to 9Mbps.

• Benefit of Receiver-Driven Mechanism

We now modify our experimental setup to a simple dumbbell topology where the link bandwidth between R_5 and C_{32}, *i.e.*, X is 0.3Mbps. Since C_{32} can receive only at 0.3Mbps from P_2, C_{31} is able to make use of the spare capacity of its bottleneck link between R_2 and R_5 and receive at approximately 2.7Mbps (see Fig. 12a), thereby making optimal use of that link (total usage is 2.7Mbps + 0.3Mbps). On the other hand, a flow based fair queuing approach would only achieve 1.8Mbps (1.5Mbps(C_{31})+ 0.3Mbps (C_{32})), since the link between R_2 and R_5 would be shared equally between flows from P_1 and P_2, only for P_2's packets to be dropped at R_5. This result highlights the benefit of using a receiver-driven approach for SAID. In the case of ICP, C_{31} is only

[5]We only implemented pgmcc in the simulation



(a) C_{31} *vs.* C_{32} (b) Aggregate

Fig. 12: Emulation result in competition scenario (using topology in Fig. 10, X=300$kbps$).

Table 1: Stall time (s) in streaming demo (video length=40s).

	C_{11}	C_{21}	C_{31}	C_{41}	Rep.
Baseline	83.384	22.507	2.461	0.886	–%
ICP	90.530	33.965	33.821	33.820	–%
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%

able to receive close to an equal share of the R_2 and R_5 link bandwidth (<1.5Mbps) because the receivers are out-of-synch. ICP gets approx 1.8Mbps aggregate throughput on that link. The total throughout for SAID (12 Mbps) and ICP (8 Mbps) are shown in Fig. 12a. For pgmcc, in the presence of a competitor, the total achievable throughput is 5.6Mbps (1Mbps * 5(C_{11}-C_{51}) + 0.3Mbps * 2(C_{32}, C_{42})).

5.2 Simulation of different applications

• Video Streaming:

We consider a streaming video application with a play-out rate of 3$Mbps$. We evaluate the effectiveness of the approaches using the *Stall time*, which reflects the impact on user experience. No stalls occur when there are no holes in the sequence in the play out buffer for the next 1s of video playback. For the baseline, we run the simulation 4 times with a single receiver requesting the video. SAID-F is when repair is by the provider at the end of the flow (similar to 'download-and-play'). SAID-S is repair by the provider while streaming the video. SAID is the peer-assisted repair solution we propose here. These solutions are compared on the tree topology (Fig. 10, with only C_{11}-C_{41} activated) as well as ICP and pgmcc.

Table 1 shows the stall time for each consumer and the repair ratio (*i.e.*, the percentage of the total number of packets received via repair over the total number of packets received, shown as 'Rep.' in the Table) for a 40s video. In ICP, when all receivers request the video simultaneously, they go *out-of-synch* soon thereafter. The stall time becomes larger than the baseline especially for faster receivers. 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 larger compared to the baseline. The user experience for the faster receivers therefore deteriorates considerably.

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_3 also experiences loss, even though its bandwidth is equal to the playout rate. This happens due to the variability in the network and the overhead of packet headers. The exact receive rate for the application can be a bit lower, despite C_3 having a nominal 3$Mbps$ available bandwidth. 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 (∼22% *vs.* ∼12.5%) since the retransmission has to go through the bottleneck link and affects the primary 'any packet' stream. Nonetheless, 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 (our proposal) 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 twice the bandwidth compared to SAID-F and SAID-S. Despite the repair, the stall time for the slower receivers (C_1−C_3) is much smaller than with the other solutions, even though they are playing the video at the same rate of 3$Mbps$. We varied the video length and observed that the stall time grows proportionally with the video length but the pattern in Table 1 still holds.

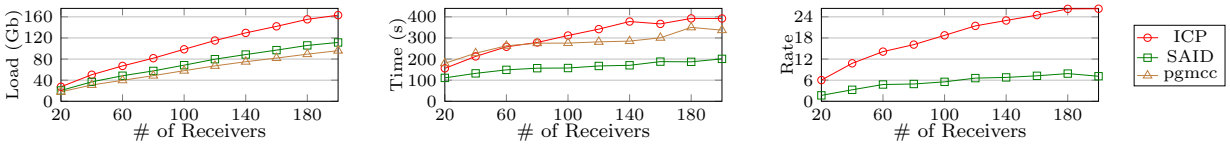
• File Content Delivery:

We evaluated the complete SAID solution with the RocketFuel topology (AS-3967). We randomly place 20-200 receivers on the 79 core routers. Since RocketFuel does not have bandwidth information, we assign available bandwidths in the range of 1-10$Mbps$ for each link. The result of a trace with 100 flows using ICP, SAID and pgmcc is in Fig. 13.

By decoupling reliability from congestion control, SAID has lower network load compared to ICP (Fig. 13a), especially as the number of receivers increases, by up to 46% with 200 receivers. Since ∼60% of the data is delivered at the first attempt, SAID has a much lower average # of transmissions of each packet from the provider and average flow completion time compared to ICP (Fig. 13c). However, SAID consumes more network bandwidth (∼10%) compared to pgmcc since it aligns to a faster receiver and uses repairs subsequently. For the 10% additional network load, SAID achieves lower average completion time (by ∼65%, Fig. 13b).

6. RELATED WORK

Many reliable multicast protocols have been proposed to enable large scale reliable data dissemination. To maximize the utility of multicast, cyclic- and scheduled-multicasts [22, 23] have been proposed to benefit the consumers that are not starting at the same time. These solutions can be broadly classified into two categories based on the chosen repair mechanism: provider repair or peer-assisted repair. The clients in provider-repair mechanisms like [24,25] send 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. Peer-assisted approaches like [26,27] 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 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 [28], in these solutions the receivers have to reveal their identities (IP addresses) to the peers, and thereby trust them as there is no guarantee of data integrity during peer-repair.



(a) Aggregate network load (b) Average completion time (c) Average retransmission rate

Fig. 13: Simulation result for file content delivery application.

Layered multicasts are also proposed to deal with heterogeneous consumers. In [29], 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. Nonetheless, having a reliable multicast capability for a single rate stream is still fundamental for broad-based use across all kinds of applications. Therefore, the layered multicast solutions can be considered as orthogonal to the single-rate reliable multicast.

Pub/sub based large scale data dissemination solutions such as [21, 30] lack an efficient mechanism to ensure reliability and avoid congestion collapse in the network.

7. CONCLUSION

Designing a congestion control mechanism for efficient information delivery to large numbers of receivers is difficult, particularly with heterogeneous receiver bandwidths. Through emulation and an analytical model, we showed that heterogeneous receivers will get out-of-synch with existing receiver-driven, in-sequence request-response approaches. To overcome several of these limitations, we proposed SAID, a control protocol that allows receivers to request 'any-next' packet instead of the 'next in-sequence'. SAID retains network fairness and delivers more packets on the first attempt. To be a general-purpose control protocol that satisfies different application requirements, SAID also provides application-specific modules. The first is a repair module that exploits content at peers and in-network caches to provide varying degrees of reliability to match application need. The second is a provider rate adaptation module that strikes a balance between session completion time and network traffic.

Our evaluations show that SAID achieves efficiency and fairness on each path between the provider and receivers. From a large scale simulation we show SAID reduces the aggregate network load (by ∼46%) and transmission completion times (by more than 50%) compared to ICP. SAID also reduces average completion time by ∼40% while only increasing network load by ∼10% (for repair) compared to pgmcc, which aligns the sending rate to the slowest receiver. Further, based on measurements on a prototype, SAID outperforms ICP, getting 50% higher aggregate throughput and almost twice the throughput of pgmcc. With the efficient repair of SAID, streaming applications have a much smaller stall time compared to other mechanisms.

8. ACKNOWLEDGEMENTS

This work is supported by the EU-Japan ICN2020 Project (EU HORIZON 2020 Grant Agreement No. 723014 and NICT Contract No. 184); and US NSF under Grant No. CNS-1455815. We thank our shepherd, Lixia Zhang, for her insightful feedback, and Xu Chen for his help on modelling.

9. REFERENCES

[1] L. Rizzo, "pgmcc: a tcp-friendly single-rate multicast congestion control scheme," in *SIGCOMM*, 2000.

[2] M. Castro *et al.*, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *JSAC*, pp. 1489–1499, 2002.

[3] V. Jacobson *et al.*, "Networking Named Content," in *CoNEXT*, 2009.

[4] L. Zhang *et al.*, "Named Data Networking (NDN) Project," PARC, Tech. Report NDN-0001, 2010.

[5] G. Carofiglio *et al.*, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *ICN*, 2012.

[6] S. Salsano *et al.*, "Transport-layer issues in information centric networks," in *ICN*, 2012.

[7] L. Saino *et al.*, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *ICC*, 2013.

[8] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM*, 1988.

[9] S. Oueslati *et al.*, "Flow-aware traffic control for a content-centric network," in *INFOCOM*, 2012.

[10] S. Arianfar *et al.*, "On content-centric router design and implications," in *ReArch*, 2010.

[11] V. Gopalakrishnan *et al.*, "Understanding Couch Potatoes: Measurement and Modeling of Interactive Usage of IPTV at Large Scale," in *IMC*, 2011.

[12] S. Floyd *et al.*, "Promoting the use of end-to-end congestion control in the internet," *TON*, pp. 458–472, 1999.

[13] M. Luby *et al.*, "Raptor Forward Error Correction Scheme for Object Delivery," RFC 5053, IETF, 2007.

[14] J. He *et al.*, "Toward Internet-wide Multipath Routing," *Network*, pp. 16–21, 2008.

[15] V. Sourlas *et al.*, "Information resilience through user-assisted caching in disruptive content-centric networks," in *IFIP Networking*, 2015.

[16] J. Chen *et al.*, "SAID: A Scalable and Adaptive Information Dissemination Protocol in ICN," Tech. Rep., 2014. [Online]. Available: https://www.net.informatik.uni-goettingen.de/publications/1912/SAID.pdf

[17] S. Floyd *et al.*, "Random early detection gateways for congestion avoidance," *TON*, pp. 397–413, 1993.

[18] K. Nichols *et al.*, "Controlling Queue Delay," *ACM Communications*, pp. 42–50, 2012.

[19] J. Gettys *et al.*, "Bufferbloat: dark buffers in the internet," *Communications of the ACM*, pp. 57–65, 2012.

[20] R. Mahajan *et al.*, "Inferring Link Weights using End-to-End Measurements," in *IMW*, 2002.

[21] J. Chen *et al.*, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.

[22] K. C. Almeroth *et al.*, "Scalable delivery of web pages using cyclic best-effort multicast," in *INFOCOM*, 1998.

[23] V. Aggarwal *et al.*, "The effectiveness of intelligent scheduling for multicast video-on-demand," in *Multimedia*, 2009.

[24] D. Towsley, "An analysis of a point-to-multipoint channel using a go-back-n error control protocol," *Communications, IEEE Transactions on*, pp. 282–285, 1985.

[25] K. Sabnani *et al.*, "Multidestination protocols for satellite broadcast channels," *Communications, IEEE Transactions on*, pp. 232–240, 1985.

[26] S. Floyd *et al.*, "A reliable multicast framework for light-weight sessions and application level framing," *TON*, pp. 784–803, 1997.

[27] S. Ratnasamy *et al.*, "Revisiting ip multicast," in *SIGCOMM CCR*, vol. 36, no. 4, 2006, pp. 15–26.

[28] N. Duffield *et al.*, "Distrust and privacy: Axioms for multicast congestion control," in *NOSSDAV*, 1999.

[29] S. McCanne *et al.*, "Receiver-driven layered multicast," in *SIGCOMM*, 1996.

[30] N. Fotiou *et al.*, "Developing information networking further: From psirp to pursuit," in *BROADNETS*, 2012.

6 Name Based Disaster Communication Framework

CNS: Content-oriented Notification Service for Managing Disasters

Jiachen Chen*†, Mayutan Arumaithurai†, Xiaoming Fu†, and K. K. Ramakrishnan‡
*WINLAB, Rutgers University, NJ, U.S.A. jiachen@winlab.rutgers.edu
†Institute of Computer Science, University of Göttingen, Germany. {arumaithurai,fu}@cs.uni-goettingen.de
‡University of California, Riverside, CA, U.S.A. kk@cs.ucr.edu

ABSTRACT

Disaster management critically depends on timely and efficient communication. To better deal with an incident, authorities from different services (*e.g.*, fire, police) and jurisdictions need to work together in a new dynamically created team, different from their original organizational/administrative hierarchy. Unfortunately, existing solutions (*e.g.*, IP, or traditional telephony) are not well-suited to deal with such group communication due to the dynamic binding between *roles* and individuals, and mobility. A significant burden is placed on administrators to just establish and maintain necessary channels, distracting them from restoring order. To make things worse, since senders do not know which individual(s) to send to, information cannot reach the right people, delaying rescue efforts.

We propose CNS, leveraging the benefits of ICN to provide the essential communication for efficiently managing disasters. We first design a namespace enabling dynamic creation and evolution of incident related (sub-)namespaces to represent roles of first responders assigned to the disaster. This allows first responders to receive the appropriate information on a timely basis, with senders addressing the recipients based on their roles. Predefined namespace templates for disaster types minimize management overhead for establishing communication. We also find the need for a new enhanced forwarding rule to support such a recipient hierarchy.

We have developed a prototype demonstrating feasibility and efficiency. With the help of large-scale simulations and real-world disaster traces, we compare CNS with an IP-based solution. CNS can significantly reduce network load and latency in addition to the qualitative benefits of simplified operations, appropriate prioritization and security.

CCS Concepts

•Networks → Application layer protocols; Naming and addressing;

ICN'16, September 26 - 28, 2016, Kyoto, Japan

ISBN 978-1-4503-4467-8/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2984356.2984368

Keywords

ICN; Disaster Management; Naming Structure; Notification

1. INTRODUCTION

We have witnessed an increasing number of disaster events, both natural events and ones caused by terrorists in the recent past. Most of these events, such as the 2005 London bombing (adversarial), the 2014 Kaohsiung gas explosion (accidental), and the 2015 Nepal earthquake (natural) point to a common trend in that these disasters often comprise multiple incidents occurring in different places around the same time. These events result in a massive need for first-response teams to manage the aftermath of the disaster, including rescue operations, dealing with emergencies and ensuring people are safe and appropriately informed to prevent panic. A detailed the report of London Bombing [1] emphasized the need for enhanced communication capabilities and repeatedly recommended putting in place effective communications within *and between* the emergency services for such incidents. There is a clear need for cross-functional cooperation and collaboration across administrative and management boundaries to manage the disaster. Special teams of first-responders with different complementary expertise have to be dynamically formed, with a management and organizational structure that is different from their normal management and administrative hierarchy and responsibility. In fact such dynamically formed teams responding to the incident might also include members who are not part of the traditional first response service team (*e.g.*, the role of the underground control center in the London Bombing incident).

However, we observe that it is difficult to achieve the effective communication in the aftermath of a disaster with existing communication frameworks. The first difficulty lies in the fact that in order to send a message (either calling for help or providing information about disaster), the sender needs to know the specific individual(s) (and their phone numbers or eventually their IP addresses) who are dealing with the incident. As a result, it often causes messages to not go to the right people and results in confusion rather than deeper understanding of the disaster among the first responders. The consequence of such confusion is that the management of the event is inadequate, with commanders erroneously dispatching or not allocating resources appropriately. What make matters worse is that, once the first-responders are mobilized, the lack of communication among those dynamically formed groups results in difficulty in correcting for errors (*e.g.*, redeployment) or sending them information in a timely, efficient manner as the

managers/commanders have to keep track of the position of each unit and send messages to the individuals separately. All these issues resulted in delayed response and poor outcomes for disaster management.

Timely information dissemination to the right recipient is important for disaster management. A communication framework based on IP – a location-dependent protocol – has inherent difficulties since the communication has to be based on the individual's address. On the other hand, Information-Centric Networking (ICN) paradigms such as NDN [2], MobilityFirst [3] and XIA [4], among others, treat contents and names as first class entities. These solutions enable the access of information based on its name or identity, without regard to location. The ability to access and disseminate information with network support enables ICNs to deliver the desired information in a timely manner. Some ICN approaches, such as [5], provide enhancements to ICN to get efficiencies in delivering information using a "push" semantic (publish/subscribme and multicast). With such an approach, establishment of a group is convenient without having to first distribute group IP addresses before information is exchanged. These capabilities of ICN are highly desirable for communication in disaster management situations.

However, these approaches need to be further enhanced to satisfy the communication needs of disaster management, where context- and recipient-driven communication is needed. Senders need to address recipients based on their roles/persona (or context) rather than the individual(s) or their addresses. *E.g.*, a commander might want to send a command to all firemen dealing with the London Bombing at Aldgate site without the need to know each of them. Further, members should have a way to dynamically form specific teams (instantiate their roles) in order to send and receive messages efficiently. Of course, they need to retain their original administrative hierarchy for other purposes.

In this paper, we propose the introduction of two capabilities in ICN to achieve this functionality. The first is a flexible namespace that can represent the context (or organization) of the normal administrative hierarchy as well as the special recipient hierarchy needed for incident response. The namespace should be able to evolve (create, modify and revoke elements) dynamically during the lifetime of disasters to minimize the management and messaging required to just manage the team. The second is a forwarding logic required for supporting recipient hierarchy, in contrast to how ICN routers forward traffic based on the typical name hierarchy for content. For example, with the typical content hierarchy in (the COPSS enhancement to) NDN, when a recipient subscribes to a name (*e.g.*, `/sports`, indicating an interest in messages related to all sports topics) he would receive messages that are sent to the names under it in the hierarchy as well (*e.g.*, `/sports/football`, indicating the message is related to football under the sports category). Forwarding based on a longest prefix match enables this quite simply. However, with recipient hierarchies, a recipient subscribing to a name (*e.g.*, `/police/Aldgate`, for the policemen in Aldgate) should receive messages that are sent to the names above it in the hierarchy (*e.g.*, `/police`, for all policemen). This new forwarding logic should be added to the network to enable the efficient forwarding for disaster management.

Based on a careful study of different kinds of disasters (using sometimes limited publicly available information), we make the following specific contributions:

- We identify the requirements to an efficient communication platform in managing disasters;
- The design of a naming schema that can support both communication within the normal administrative hierarchy as well as supporting cross-jurisdiction/cross-functional communication in disaster situations;
- A plan-and-instantiate mechanism that allows the government to plan the "roles" (namespace structure) beforehand and dynamically instantiate the recovery plan for a disaster, thus minimizing the management and messaging required in the first stage of the disaster recovery;
- A new forwarding logic (recipient-hierarchy) to meet the need for communicating to dynamically formed groups sets of recipients for efficient multicast/anycast; and
- Qualitative and quantitative studies that show the benefit of CNS in terms of flexibility and efficiency in managing the disaster.

2. RELATED WORK

Here, we discuss the state of the art techniques that are, and can be used by emergency services.

2.1 Legacy and IP-based Emergency Services

Existing, legacy emergency service infrastructures that rely on circuit switched telephony (including mobile infrastructure such as MobileIP for voice calls) for emergency calls is not suited for data communication and cannot enable enhanced (interactive) services including video, written messages and contextual information. Recently, an attempt is being made to design and implement the *next generation emergency services* (NG112, NG9-1-1) [6–9] by adapting the IP infrastructure to meet the requirements of emergency services. A goal of this work is to allow citizens/authorities to contact emergency services with technologies they use to communicate every day. *E.g.*, work such as location-identification [10] deals with identifying location of a SIP caller, services such as LoST [8] map location to services based on service boundary [11] (*i.e.*, contact the closest or administratively correct police station). However, most of these work only focus on emergency calls from civilians (*e.g.*, [9]). Moreover, these design choices are affected by the limitations of the legacy as well as the IP infrastructure and cannot work out of the box in a fragmented scenario since they depend heavily on end-to-end communication.

2.2 Location Independent Architectures

ICN, as we stated before, shifts the focus from location dependent routing to forwarding messages based on the content identities. NDN [2, 12] is a popular variant of ICN. The current design of NDN adopts a URL-like scheme for content names, *e.g.*, this paper could have a name `/ICN16/CNS.pdf`. Content providers register the availability of content by its prefix. These prefixes are announced for global reachability in the Forwarding Information Base (FIB) of routers. Two kinds of packets are used: *Interest* and *Data*. An Interest is sent by a consumer to query for data. When forwarding an Interest, routers perform *longest-prefix matching* in FIB and find proper outgoing (inter)face towards the provider. Any data provider who receives the Interest can respond with a Data packet. Data packets follow the reverse path established by the Interest.

COPSS [5] extends NDN with push (or multicast) functionality. Instead of using ContentNames to identify con-

tent, COPSS uses hierarchical Content Descriptors (CDs) to describe content, *i.e.*, a content can have multiple CDs and a CD can identify multiple content items. CDs are also in the form of URLs, *e.g.*, the paper can have CDs `/Networking/ICN`, `/UniGöttingen/papers/CNS`, *etc.* A consumer interested in a CD can subscribe to the CD and will receive all the contents with the CD and its descendants. COPSS maintains a rendezvous point (RP) based subscription tree in a new data structure in the routers called Subscription Table (ST). As described in [13], it also allows the presence of multiple RPs to avoid traffic concentration.

The commonality across these solutions is that they propose the use of *name based forwarding* which avoids the early binding of forwarding messages to a specific location. CNS leverages the benefits provided by these solutions, while paying particular attention to the requirements of name-based communication in disaster situations to enhance their capability. Additionally, CNS can leverage LoST-like services to map roles to location dependent authorities.

2.3 ICN-DTN

Some early work has proposed the use of ICN in Delay Tolerate Networks (DTN)/fragmented environments. In [14], the authors present scenarios for using ICN in natural disasters. Work such as [15–18] deal with improving data delivery in DTN environments by leveraging benefits of ICN. [16] proposes a priority-based information dissemination/flooding mechanism in DTN, where the priority is based on the names; [17] proposes the use of ICN for vehicle to vehicle communication; [15,18] propose to use ICN-based mules to spread information in DTN; [19] proposes an energy efficient message delivery mechanism that leverages collaborative communication in disasters.

CNS can make use of these solutions to perform data delivery in fragmented/DTN scenarios (disasters). Additionally, our work complements these efforts by providing a more comprehensive solution to handle the requirements of all kinds of disasters (adversarial, accidental and natural), both in fragmented and non-fragmented cases. Moreover, CNS details a naming mechanism that allows authorities to focus on the role instead of the individual that is performing that role at a particular point of time.

3. STUDY OF DISASTER SCENARIOS

We explore several example disaster situations to help us understand the requirements for the communication platform. According to the US National Protection Framework [20], disasters can be generally divided into 3 categories – adversarial, natural and accidental. We first look at adversarial disasters (with an example) which have the highest requirements on the communication platform and then extend our view to accidental and natural disasters to build a generic platform for all kinds of disasters.

3.1 Disaster Example – London Bombing 2005

At 8:50 am, July 7, 2005[1], a suicide bomb was detonated on the eastbound Circle Line train #204 traveling from Liverpool Street to Aldgate Station. Within 1 min, a second explosion took place on a Circle line train #206, going westbound from Edgware Road to Paddington. Approximately 2 min later, a third bomb was detonated on a southbound Piccadilly Line train #311. At 9:47 am, a fourth bomb was detonated on the top deck of a #30 bus at Tavistock Square. The explosions resulted in 52 deaths, 700 people being physically injured and hundreds of people being directly affected.

The overall picture from 8:50 until about 9:15 was chaotic. Multiple, often conflicting, reports were being made, some to London Underground's Network Control Centre, some to the emergency services, and some to the media. It was not clear what had happened, or indeed where. One major cause for the chaos was that the messages could not reach the right people (*e.g.*, first responders, resource managers).

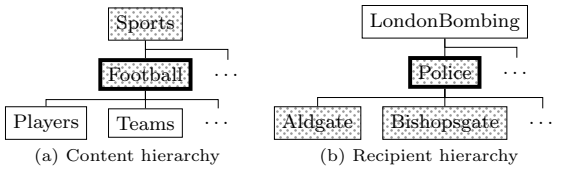
Under the circumstances where the situation was not clear, first responders were unable to dispatch resources effectively. The first fire engines were dispatched to Praed Street instead of Edgware Road at 9:00 and it was not re-deployed until 9:37, nearly 40 minutes after the initial event. The survivors spoke repeatedly of the apparent lack of ambulances, equipment and supplies at the scenes, *even an hour or more after the explosions*, to the event review committee. Dispersal of patients to hospitals was also uneven because of a breakdown of communications within the Ambulance Service.

The lack of proper communication among different services hampered coordination. The London Emergency Services Procedure Manual clearly states that a "major incident" can be declared by any of the emergency services, the implication being that this will be done on behalf of all the services. However, different first responding services declared a major-incident separately, and at different time periods (*e.g.*, in Aldgate, the ambulance service realized it 19 minutes after the fire brigade did; in the case of King's Cross, fire brigades were dispatched to the wrong sites and it is unclear when they realized that it was a site of a major incident). This late realization also affects the involvement, either directly or indirectly, of large numbers of people including other agencies such as the Local Authorities, National Health Service, Environment Agency, Military and Voluntary agencies. Certain hospitals in the vicinity were not aware of such an incident and did not participate in the rescue efforts for a long time.

Other departments/individuals can also play an important role in managing the disaster, which also needs proper communication with other disaster managers. *E.g.*, the London Underground Network Control Center put in an emergency services call to three sites (all correct places) at 8:59 am (only 9 min after the explosions). The records revealed that these calls did not result in the immediate dispatch of the emergency services to the scenes. For some reason, the message did not seem to get through to the right people. London Underground Emergency Response Unit – a small and little-known team which does not even have the right to use the blue light on the roads – played a crucial role in emergency response in the absence of the Fire Brigade at Russel Square. At Tavistock Square, there were no other ambulances at the scene at that time, but the bus was located outside the headquarters of the British Medical Association and doctors and other trained first-aiders came out of the building to care for the injured. If there was proper communication to the hospitals and nearby services, the victims could have be evacuated earlier.

All these issues demonstrate that the key to an effective response to a major or catastrophic incident is communication. This includes effective communication *within and between* the emergency, health, transport and *other services*,

[1]This description of events is mainly based on [1].



(a) Content hierarchy (b) Recipient hierarchy

Fig. 1: Content hierarchy *vs.* recipient hierarchy (hashed nodes: receivers, bold: name prefix in packet).

and also with the individuals caught up in the incident, and the public at large.

What happened in London on July 7, 2005 could happen in any country, city, at any time, as we have witnessed over the last decade. Anecdotal reports in newspapers indicate the outcomes in many of these cases were impacted by lack of timely and appropriate communication. Other than the adversarial disasters, governments also have to deal with disasters caused by accidents and nature, such as earthquakes, tsunamis, nuclear reactor incidents, *etc.* Although the cause and the scale of a disaster might differ, all major incidents can be expected to share some typical characteristics: *1*) the involvement of numerous, different, agencies in the response, *2*) the importance of effective communications within and between those agencies, and *3*) the crucial importance of approaching each incident from the point of view of those directly caught up in it, either as members of public or as individuals involved in the response. The requirements on the communication platform in terms of dynamic group formation, and convenient role-based communication is similar in many of these cases.

3.2 Communication Platform Requirements

Based on studying several of these events, we arrive at the following common requirements for the communication in managing disasters:

• **Predefined roles & dynamically formed groups:**

Governments usually have plans to manage different disasters. However, these plans only consider the *roles* rather than the individuals/identities that "instantiate" these roles. This poses a challenge for current communication frameworks since they mainly focus on identities of individuals and reach them based on the communication devices they have. It is preferable to have a mapping from the role to the identity, with the mapping known to everyone authorized to manage and help the disaster. Therefore, the platform should be able to support communication based on such predefined roles and support a dynamic mapping from roles to identities without burdening individuals to manually maintain the mapping. Thus, when people are trying to provide information, they can reach the proper receivers more easily.

• **Efficient group communication:**

To deal with disasters at different scales, governments usually need to mobilize varying numbers of first responders. These responders should collectively be able to share information and receive commands. They might follow a separate control hierarchy, *e.g.*, a commander might want to send commands to the police team that is responding to the specific event, or even to all the first responders.

• **Content hierarchy *vs.* recipient hierarchy:**

The semantics for the recipient hierarchy, which is often used in the command chain, is different from what is used with content hierarchy in name oriented network architectures, such as NDN/COPSS. With content hierarchy, if a consumer sends an Interest to `/Sports/Football` (the bold node in Fig. 1a), the interest will be sent to providers serving (by propagated FIB entries) `/Sports` and `/Sports/Football` (dotted nodes in Fig. 1a) according to the longest prefix match rule. Providers serving `/Sports/Football/Players` (or `Teams`) will not receive this request. Longest prefix match is sufficient to handle content hierarchy.

However, in a command chain (we refer to as recipient hierarchy), when a commander wants to send an interest to all policemen dealing with London Bombing (with name `/LondonBombing/Police`, the bold node in Fig. 1b), this message should reach policemen serving the FIB entries `/LondonBombing/Police` and its descendants like `Aldgate`, `Bishopsgate` (dotted nodes in Fig. 1b). This is not achievable using longest prefix match based forwarding. A similar challenge arises in pub/sub as well, when forwarding based on the subscription table entries. Therefore, we see a need to have a recipient hierarchy in the network in order to send packets to people serving names that are descendants of the prefix contained in the packet.

• **Priority-based communication:**

Prioritization is important in disaster communication. Extreme solutions like ACCess OverLoad Control (ACCOLC) do allow authorities to have reasonable communication, but at the cost of blocking all civilian traffic and at the risk of causing public panic. Therefore, when civilian communication is still desired in such situations, the system should not place a blanket block of all civilian communication (as is often the case with current telephony-based solutions). Instead, it should prioritize the communication among authorities, enabling an efficient command chain for managing the situation.

4. ARCHITECTURAL DESIGN

This section first provides an overview of CNS and then focusses on design details such as naming, use of templates and forwarding.

4.1 Architecture Overview

CNS holistically considers communication both for normal situations and for disaster scenarios, with a goal of using the same common infrastructure at all times. To find a suitable naming schema for CNS, we studied the use of flat names [3,21], hierarchical names [5,12] or even more complicated namespaces [22, 23]. We observe that organizational hierarchy is common, well understood and is often efficient in managing human interactions. The hierarchical structure is widely adopted in many situations, including managing first responder services, the military, *etc.* In our architecture, we try to represent what is already used in real world communications, so that the users do not need to change the organization or behavior they are already used to. We also want to have the network exploit the namespace to replicate and distribute the information efficiently to the group of recipients defined by the names. Therefore, CNS adopts hierarchical names for communication in disasters.

Fig. 2 shows an example namespace at a country level (UK). For communication among authorities under normal circumstances, CNS uses a naming structure to represent the administrative/organizational hierarchy (left side in the figure). For dynamically-created and possibly transient teams dealing with different incidents, a place holder `/UK/Incidents` is created (right side). Each incident (from a small incident

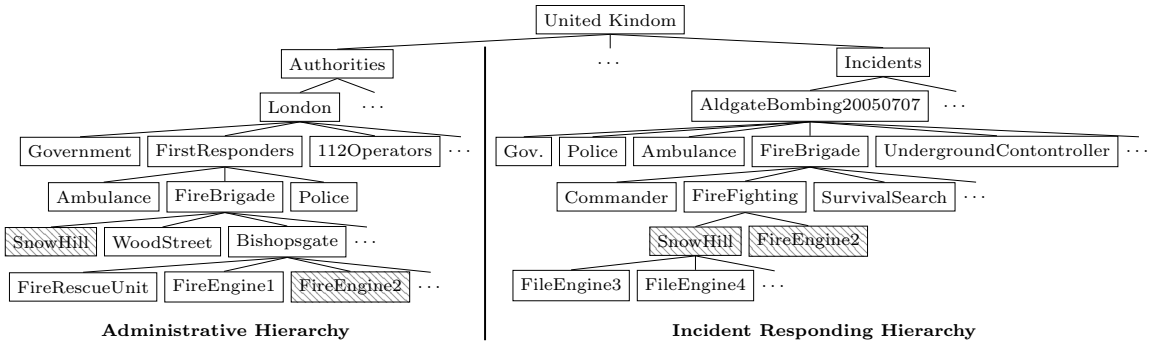


Fig. 2: Name Hierarchy in CNS.

such as a gas leak to a disaster such as a tsunami) will have a sub-namespace created under this place holder. Templates for different types of disasters can be planned beforehand and instantiated on seeing the disaster in order to minimize management and messaging overheads.

CNS is an *application-layer* design that helps the disaster managers to decide what names they need to use and how they are going to communicate when disaster strikes. It can run on any Information-Centric Network like [3, 12, 21–23] as long as there is a proper mapping from the hierarchical application-layer names to the identities used in the network. *E.g.*, each node in the hierarchy can have a GUID in MobilityFirst [3]. To send a message, the sender needs to carry all the related GUIDs. However, the GUIDs in MobilityFirst do not have relationships. To send a message to `FireBrigade` in the administrative hierarchy, the sender has to carry the GUIDs of all the descendants of `FireBrigade`, with associated traffic and computation overhead. CNS can also use assertion-based networks (*e.g.*, INS [22]) since it is easy to map the hierarchical structure to (XML-based) assertions. Nonetheless, each router in INS has to parse complicated XML queries before forwarding the information. While it is true that INS can provide similar functionality, solutions such as NDN provides sufficient functionality at a much lower cost. Therefore, we prefer to run CNS over NDN/COPSS like networks since they provide native support for hierarchies at a lower cost, so that CNS can exploit the network for efficient multicast/anycast, something that is critical for efficient disaster management.

4.2 Administrative Hierarchy

To provide convenient and efficient communication among people associated with various authorities as well as people outside these groups, CNS uses a name-based solution – a hierarchically structured name architecture to represent the organizational command chain. This is convenient as they only need to communicate with a "role" at the appropriate position rather than considering the individual who "instantiates" the role. As for mobility, since communication is based on names, individuals, including first responders will not have a new identity when they physically move from one network "location" to another.

Fig. 2 shows a possible namespace for communication for authorities in London, especially those responsible for safety, law and order, *etc.* All authority-related roles are under the prefix `.../Authorities`. We assume that there is central control (called the GOLD coordinating group in London) for all first responder services, but the division of responsibility is different across departments. *I.e.*, the distribution of police stations is different from the distribution of ambulance pools, which is again different from the fire brigade stations. Therefore, the namespace assignment should follow the same organizational command chain as in the real world. The figure shows a possible division of responsibility for police services in London. Each sub-node can be further divided to correspond to the real-world command chain.

The communication model for normal circumstances is similar to what was proposed in [2] and [5]. The communicating parties either send unicast (similar to VoCCN [24]), multicast (similar to pub/sub in COPSS) or anycast (similar to query/response in NDN). Following Fig. 2, when a commander wants to send a message (multicast) to all firemen in London, he can simply send a message with name `.../London/FirstResponders/FireBrigade`. All the fire fighters listening (subscribed) to FireBrigade at WoodStreet, Bishopsgate, SnowHill, *etc.* would receive the message. With this functionality, each message will only be sent once and is replicated in the network. If a commander wants to talk to any firemen in WoodStreet, he can simply initiate VoCCN with callee=`.../FireBrigade/WoodStreet`. All the firemen listening on the channel are serving this FIB (or its prefix) and they will receive the Interest. Bi-directional communication can start after the basic handshake.

To deal with the duality between recipient hierarchies and content hierarchies, CNS modifies the forwarding engine to support the new semantics, and adds an extra bit in the packet header indicating which forwarding strategy routers should use (see §4.4). Namespace `112Operators` is used to receive emergency calls from civilians. It can be further divided based on the requirement of the emergency call bureau. The network does not place any limit on civilian emergency calls, but in NDN, each data would have a signature from the data provider to enable identification of callers.

Namespace `.../London/Government` could be established for civilians to receive news from the government. The government can use this channel to broadcast alerts on disasters, report progress on rescue efforts, *etc.*

With name-based communication, CNS allows emergency management authorities to build up a hierarchy according to their real-world command chain. Officers in different departments can listen to (serve a prefix or subscribe) these channels to "instantiate" these roles. When communicating, authorities only need to communicate to the role rather than to the the individual who is currently in that role. CNS can provide convenience, flexibility and efficiency for communication among authorities even in non-disaster situations.

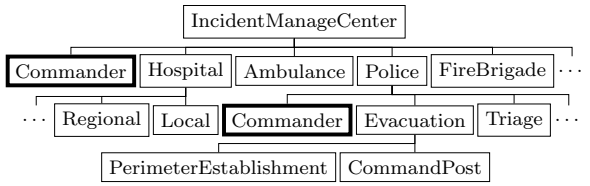


Fig. 3: Template for bomb incident management.

4.3 Incident Response Hierarchy

In this sub-section, we will walk through how a government should prepare a disaster template beforehand, how an authority instantiates a new namespace in the hierarchy when it knows of a disaster, and how first responders listen to the new namespace based on their duty assignment and communicate with each other. Disaster relief of the London bombing will be used as an example.

4.3.1 Disaster Templates

While it is true that first responders can still use the channels established for normal operations, temporary command chains (different from the original organizational structure) may need to be set up based on the magnitude of the disaster. As a standard operation procedure (SOP), government emergency management agencies prepare different plans for different kinds of disasters [25]. These plans usually focus more on the roles/functions (*e.g.*, communication/power restoration, civilian relocation [26]) in the aftermath of a disaster rather than the exact scale or location of the response team, so that each plan can deal with a certain kind of disaster. The assignment of the responders is performed during the disaster based on the actual situation.

CNS allows emergency management agencies to prepare a namespace template similar to their existing disaster management plans. Fig. 3 shows an example template for bomb incident management based on the plan written by authorities in Marietta, Georgia, USA [27]. The root of the template represents the management center. It will be renamed with the event identity when the template is installed into the existing namespace. Under the root, there are usually the services that are involved in incident management, *e.g.*, Ambulance, Police, Fire, *etc.* The sub-namespace under the departments can be set up based on the responsibility of each department. *E.g.*, in [27], the responsibilities of the Marietta police department after "actual bomb or explosive device detonated" include: triage the people on site, evacuate civilians, search for explosives, *etc.* The planner can setup sub-namespaces like Evacuation, Triage, Explosive Search accordingly. These functions can be further divided and sub-namespaces created as needed (*e.g.*, `PerimeterEstablishment` and `CommandPost` in the figure).

The disaster plan can also include Hospital in the namespace, although hospitals might not have a node in the original administrative and organizational hierarchy. However, when there is a disaster, the government would need hospitals to stand by, wait for notifications and report their status. The planner for disaster management can therefore provide a namespace to the hospitals. The namespace can also be sub-divided based on the functions (*e.g.*, by distance or speciality) in case the disaster management officers want to send different messages to different kinds of hospitals.

Sometimes, disaster management officials may want to send messages only to the manager/commander of a certain team rather than the whole group (*e.g.*, a fire fighter might need to report the situation to the on-site police supervisor). To deal with this requirement, the planner can set a Commander under each level of management (see the bold nodes in Fig. 3). The fire fighter can now send messages to namespace `.../Police/Commander` and only the police supervisor(s) who listen to that name will receive the message.

4.3.2 Dynamic Group Formation

Once there is a template, the primary disaster management commander can easily "instantiate" the template in the namespace when a disaster occurs. In Fig. 2, there is an "Incidents" sub-namespace. According to a disaster response hierarchy, such a namespace can also be placed at a departmental, state or national level to deal with disasters of different scales. What the commander needs to do is to provide a name for the incident, and then "copy" the whole tree of the template to the Incidents namespace. *E.g.*, `Aldgate-Bombing20050707` can be seen as an example of instantiating the bomb incident management plan (Fig. 3). Note that instantiating a template does not change anything in the network. The routers do not store the namespace (no extra state). Nor will new routes have to be created until the first responders listen to the roles. However, the namespace exists in the application layer, and the first responders would receive information about the name(s) they should listen to and the name(s) they should communicate with. The form of communication could range from query/response to anycast to multicast. This design helps to reduce the substantial amount of control messages exchanged immediately after instantiating a template. The network would build up the FIB gradually when first responders listen to the names (by propagating FIB/ST). Also, the roles that do not have people responding to it (it can happen in many cases since the template might consider a more complicated situation and involve more people) will not have extra FIB entries. Therefore, planners can feel free to instantiate a template, and/or design a more detailed and complex template without worrying about more state (control overhead) being used.

4.3.3 Role Instantiation & Authorization

The right to send/receive messages in a certain namespace has to be authorized, similar to the capabilities in the real world. Let us look at how an event may play out. When a disaster occurs, the administrator/commander contacts the departments in the administrative/organizational hierarchy. The departments would dispatch units to deal with the incident. Similarly, in CNS, the incident commander has a key to instantiate the incident namespace. He can use the key to certify the departments that will be involved in the disaster management. The keys can be provided to the departments, by just using the department namespace. The departments can use the key to further certify first responders. The keys to the first responders may be given following the administrative hierarchy.

On receiving the new key to the disaster namespace, first responders can serve prefixes or subscribe to CDs accordingly. The network will build appropriate dissemination paths based on the routing strategy. In Fig. 2, FireEngine2 (under Bishopsgate in the administrative hierarchy) is mobilized for AldgateBombing. The commander can even add a whole department into the disaster management namespace (*e.g.*, SnowHill and all its units are added to deal with Aldgate bombing).

The benefit of the new namespace is effective and convenient communication. *E.g.*, when an incident commander

wants to send instructions to the policemen who are establishing a perimeter, some of the policemen might come from WoodStreet while others might come from Bishopsgate. If the original administrative command chain were used, the commander would have to send the instructions twice, and the policemen from WoodStreet that are dealing with other duties will also receive them. This is an unnecessary burden and overhead on the commander, the network and the first responders. Using the disaster namespace, the commander only needs to send the instructions once and they will only be disseminated to the appropriate officers.

The requirements for disaster management may change according to the nature of the disaster and the judgement of the management officers. CNS also allows the officers to dynamically add sub namespaces as needed and new first responders can join the new namespace to participate in the disaster management. The procedure is similar to adding a new disaster namespace and we omit the details here.

4.4 Supporting Recipient Hierarchies

With CNS, we observe that communication along the chain of command for both the administrative and incident response is based on the recipient hierarchy, which is quite different from the content hierarchy.

Although it is possible to support recipient hierarchies without modifying the forwarding logic in an NDN/COPSS router, such a solution can result in inefficiency in the network. *E.g.*, consider the case where the police commander in Fig. 1b needs to get multicast calls/messages meant to reach *all* of the members dealing with London Bombing (group identified by the name `/LB`), or to the subset who are the members of the police department (identified by `/LB/Police`), but he does not need to get messages sent to a specific individual (*e.g.*, `/LB/Police/Aldgate/PoliceA`). To avoid that with a content hierarchy, he has to create and listen to a new name `/LB/Police/Commander`. This is equivalent to using specific multicast "channels" without taking advantage of the hierarchy. Then, upper-layer commanders have to send copies of each message to multiple, separate channels. This not only results in more traffic in the network, it is also undesirable in the real world since it places a considerable burden on the commanders to have to keep track and send to each of the names in the namespace.

Because of these concerns, we propose an additional forwarding logic for routers. A flag in the packet header can be used to indicate if the packet should be forwarded based on this recipient hierarchy or the usual content hierarchy. On receiving such packets, based on whether it is unicast/anycast (an Interest packet) or multicast (a Publication packet), the router would choose to look into the FIB or ST accordingly (following the COPSS design). The new logic requires the router to find a match in FIB/ST and forward packets to *any/all entries* under the name that is a match. *E.g.*, [28] provides a mechanism to perform efficient lookups in a data structure like the FIB and ST. On detecting a match, the router would only have to get the outgoing faces of any or all the descendants in the tree, and forward the packet accordingly. *E.g.*, on receiving an Interest with name `.../FireBrigade` and the special flag (an initial packet for a call to any responders in the fire brigade), the routers will forward it to a responder who serves `.../FireBrigade/*` (could be an engine group in SnowHill). On receiving a Publication with CD `.../FireBrigade` and the special flag (a multicast message sent to all the officers in the fire brigade service), the routers will get all the outgoing faces of subscribers who subscribe to `.../FireBrigade/*`, and forward the packet accordingly. With this subtle change, we can support the new recipient hierarchy. The feasibility and efficiency of the modification is evaluated in §5.

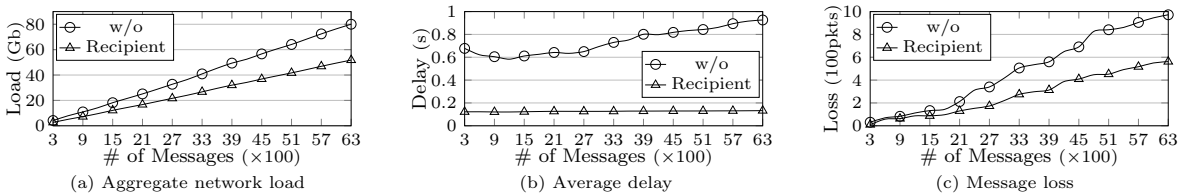
4.5 Attribute-based Prioritization

During the course of a disaster event, citizens seek to communicate with one another to convey and enquire about their well-being and location. After the London bombing incidents reported by BBC, Vodafone experienced a 250% increase in the volume of calls and a doubling of the volume of text messages. Cable and Wireless handled 10 times the normal call volume of the Vodafone and O_2 networks. This sudden burst of traffic severely affected information exchange related to the incident, impacting: *1*) communication between different services, which usually have to go through the civilian network; *2*) communication between civilians and first responders for information updates; and *3*) civilian calling for help, even among each other. Although the ACCess OverLoad Control (ACCOLC) feature adopted in telephony allows authorities with special devices to communicate on the civilian channel (by suppressing all other civilian traffic), using it would have cause even more severe issues: *1*) authorities without special devices would not be able to communicate; and *2*) public panic since civilians would lose communication at all. Therefore, a system based on logical prioritization (rather than blocking) is desirable, to allow the disaster-related communication to be guaranteed while the traffic among civilians is supported in a best-effort manner.

In NDN, a straightforward way is to use name-based prioritization. *E.g.*, in Fig. 2, we can create a rule to prioritize all the traffic with name `.../Authorities` and `.../Incidents`. However, this solution has a significant drawback – it complicates the namespace. Authorities might end up having a whole name hierarchy for prioritized traffic and the same (or very similar one) for non-prioritized traffic since for the same set of receivers (destinations), the priority can vary based on content. With multi-level prioritization, it would result in having a hierarchy for each priority level, with more states in the network in both FIB and ST. Even worse, first responders have to listen (subscribe/propagate FIB) to multiple names, which will place additional burden on both the network and the responders.

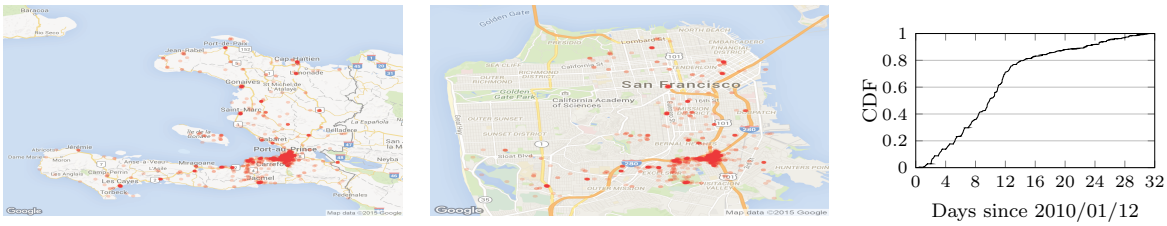
To decouple prioritization from the destination (receiver) of a message, we use an attribute field in the packet to indicate its priority level. Routers prioritize packets based on this attribute field. This attribute can be added/suggested by applications automatically. To prevent people from abusing the prioritized attributes, we can use a signature to validate if the sender can prioritize the message (and decide the priority level)[2]. For communication among authorities, we can verify if the (key of the) sender is signed (directly or indirectly) by the key of authorities in the chain of trust. Of course, authorities can also send non-prioritized messages even with the same public key and the same destination. When a civilian wants to provide important information to authorities, he can also add the prioritized attribute. While it is true that the validation of keys and signatures would

[2]Signature can be added to both Data and Interest packets [29] for authentication.



(a) Aggregate network load (b) Average delay (c) Message loss

Fig. 4: Evaluation result of CNS (w/ and w/o recipient hierarchy) in lab testbed.



(a) Messages in Haiti (original) (b) Messages in San Francisco (modified) (c) Messages over time (original)

Fig. 5: Messages transformation Haiti ↦ San Francisco, on space and time.

cause overhead, we argue that it is an inevitable overhead either in the network layer or in the application layer. We believe that in-network validation is a more appropriate choice since it can prevent malicious content from even entering the network (compared to the forward and eventually-discard mechanism used by application-layer solutions). Efficiency-wise, we can make access points and gateways (or some network functions) perform the validation rather than having every router perform it, so that forwarding in the network does not suffer from the overhead.

The attribute field can also be used to alter the forwarding rules for the packets. *E.g.*, we can use an attribute to represent a civilian calling for help from nearby users. Routers can broadcast this message within a limited scope of a few hops, but with priority.

5. EVALUATION

With the help of a prototype of CNS deployed in our lab testbed and a synthetic data trace, we demonstrate the feasibility of implementing and deploying CNS and show the efficiency of the proposed recipient hierarchy. A real-world topology with a real-world trace is studied using our simulator (widely used in previous work [5, 13]) for a comparison between CNS and MobileIP [30], the current state of the art for communication based on location of individuals rather than dynamically created groups. We omitted some detailed data regarding the setting of parameters here (please see the extended material [31] for a detailed description).

5.1 Lab Testbed Evaluation

We first evaluate the feasibility and efficiency of our proposed recipient hierarchy (§ 4.4) in our testbed.

5.1.1 Data Set

Our lab testbed comprises 6 physical machines that are used as routers and are connected by links with a $100Mbps$ bandwidth and $10ms$ delay. We use a single server to emulate 63 users and these users can dynamically link (with $50Mbps$ bandwidth and $5ms$ delay) to any of the six routers, based on their movement pattern. The time interval between movements of a user is probabilistic, uniformly distributed between $2s$ to $120s$. The users form a 3-level quad tree recipient hierarchy, and each node in the tree has 3 users.

The total simulation duration is 70 minutes including 4,390 reconnections in total. The total # of messages exchanged is 6,300 and the size of each message ranges from 1-99 packets (1,500 bytes per packet). Each message is assigned to a user (sender) and each sender can choose to multicast the message either to his own department (a node he is listening to) or to a subordinate department (a child node). Messages are sent based on a uniform distribution over a total period of 60 minutes (first message is sent at minute 5).

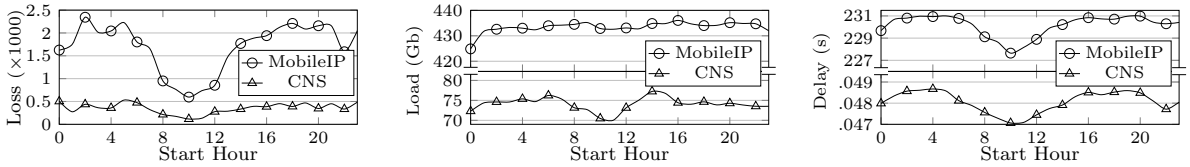
5.1.2 Evaluation Results

In Fig. 4, we compare the two variants of CNS (w/ and w/o recipient hierarchy) in terms of the total traffic, delay and # of packet lost, for varying # of messages. Message queueing at a router's busy outgoing face, can adversely impact delivery delay, especially at the RPs. Message loss is mainly caused by mobility. Nodes cannot receive messages till the network state (in FIB/ST) is properly established.

Our results show that CNS with recipient hierarchy has: *1*) lower total traffic since only one copy of the message is sent from the sender while solution without recipient hierarchy has to send the same data to each leaf node in the hierarchy separately (see Fig. 4a); *2*) lower delay since fewer packets go through the RP, thereby facing a smaller queuing delay (see Fig. 4b); and *3*) lower packet loss rate since the solution is able to better aggregate the subscriptions (see Fig. 4c). Based on packets captured by Wireshark, we see that the computation overhead for forwarding each packet is <2%. The results show that the recipient hierarchy is a cost-effective enhancement to ICN for information dissemination. However, note that this result does not mean that content hierarchy is not efficient for the purpose it was designed. The main take-away is that with a proper namespace structure, and the new forwarding rule (recipient hierarchy), CNS is better suited for the application scenario of disaster management. Essentially, both approaches are needed for a complete communication framework.

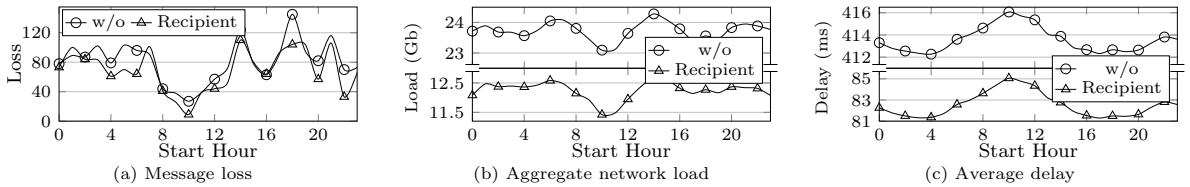
5.2 Trace Driven Simulation

In order to perform a realistic evaluation of a disaster scenario, we needed a data set that consists of: *1*) workload (messages sent and received), and *2*) communication infrastructure and mobility pattern in the disaster. Due to the



(a) Message loss (b) Aggregate network load (c) Average delay

Fig. 6: Simulation result of content hierarchy (Note the difference in scale for Load and Delay).



(a) Message loss (b) Aggregate network load (c) Average delay

Fig. 7: Simulation result of recipient hierarchy (Note the difference in scale for Load and Delay).

lack of such a complete data set, we combined 2 real-world data sets to emulate a city-level disaster, if that were to happen, say *e.g.*, in San-Francisco. We first make use of the emergency messages that were sent in the aftermath of the Haiti earthquake [32, 33] as the workload during a disaster. We then used a San-Francisco topology [34] with cab movement [35] to represent the communication environment first responders could face during a disaster. We realize that this is limiting, as mobility patterns are very likely to be different, but it does demonstrate the effectiveness of CNS.

The topology (see [31] for detail) consists of 232 routers distributed across 5 overlapping ISPs, and with each ISP having 5 non-overlapping domains. With MobileIP, we set up one home agent for each ISP at its root node. In the case of CNS, we have just one rendezvous point (RP) in the whole network. We used relatively low bandwidth links ($100Mbps$ per inter-ISP link) to reflect the use primarily for emergency data. Router processing matches the link rate, while Home-Agent processing for redirecting packets is $2ms$ and location modification is $5ms$. Although the data set is not large, our solution can scale according to other real world needs.

Haiti's message data set consisted of only a subset of the actual messages sent, *i.e.* it had 3,131 messages of the total that were sent in the first month after the earthquake. Therefore, in order to scale it up, we compressed the time for sending the messages to 1 hour, to emulate the situation of a large number of messages being sent in the immediate aftermath of a disaster. The CDF of messages *vs.* time is shown in Fig. 5c. The CDF of messages used in our evaluation look the same (but with different time scale of 1 hour) and therefore omitted in the figure. The # of packets (1500 bytes each) per message were obtained by dividing the message size by 50. The messages in the data set have the latitude and longitude of their origin (see Fig. 5a, each dot represents a message and the darker color means many messages are sent at almost the same place). We mapped these messages into the San Francisco map by performing a linear transformation and scaling it (see Fig. 5b). Since the dataset is small, we can imagine this to be a small-scale disaster (*e.g.*, highway bomb). Each emergency message is sent by an end-host linked to a router nearest to the messages' origin (*i.e.* based on the message's latitude and longitude).

The movement of receivers is also based on the San Francisco cab data trace [35] on 2008/5/28 (Wednesday), with the long trace showing a similar pattern on a daily basis. There were 494 cabs in action on this day. Since our message dataset is compacted to 1 hour, we divided the 1 day cab movement into 24 sub-traces to study the effects of an emergency (*e.g.*, a bomb detonation) at different time periods to correspond to different movement patterns and load.

5.2.1 Communication using Content Hierarchy

The Haiti message dataset is already separated into a hierarchy consisting of 8 major categories (*e.g.*, urgencies, urgency logistics, public health, *etc.*) with each category consisting of several sub-categories. The total # of categories is 36, and each message can belong to multiple categories. We configure each receiver to listen to 1 category, *i.e.* the cabs are seen as first responders moving around to help people and they could receive requests to answer emergency calls.

We compare CNS to MobileIP (Fig. 6) in terms of the # of message losses, aggregate network load and latency. Our results illustrate that CNS has a significantly lower loss rate (Fig. 6a), lower network load (Fig. 6b) and lower delay (Fig. 6c) as compared to MobileIP, regardless of which hour the disaster occurs. This is due to the fact that MobileIP relies on unicast, which results in more traffic, higher latency (path stretch), and congestion at the 5 home-agents. Moreover, in the case of MobileIP, the end-host cannot fetch the content from a neighbour who also received that data. Note that MobileIP consumes 6 times the network load compared to CNS and causes unacceptable delay ($>200s$).

On the other hand, CNS can aggregate subscriptions, lower the probability of message loss (when you are near an authority who also subscribed to the channel or a higher-level channel, a subscription can succeed within 1 hop) and ensure that there is less congestion on the RP even though there is only one RP as compared to MobileIP's 5 home-agents.

5.2.2 Communication with Recipient Hierarchies

Finally, we use the Haiti message dataset as a basis to emulate the information exchange among authorities to study the benefits of the proposed recipient hierarchy as compared to using the content hierarchy approach. We build a recipient hierarchy containing 6 levels and assign the messages and cabs into the recipient hierarchy randomly (see [31] for detailed subscription relationship).

In Fig. 7, we compare CNS *vs.* the case without recipient hierarchy using the same metrics that we used to analyze content hierarchy. We observe that CNS using the recipient hierarchy has a slightly lower loss rate due to improved ag-

gregation. More importantly, CNS with recipient hierarchy outperforms the other variant in terms of aggregate network load (almost by half) and latency (by up to 80%) due to its improved design. While we used just a single RP to highlight the benefit of the proposed CNS with recipient hierarchy, solutions such as those proposed in [13] can be used to increase the number of RPs and load balance among them in order to handle higher load. When the traditional NDN-content query approach is used, we can achieve comparable performance as CNS using the recipient hierarchy when users (first responders) know exactly when and from whom a message is being sent so that they can send an interest using NDN. Otherwise, without the benefit of aggregation, such an approach only performs as well as unicast. These results confirm the lab testbed based results of §5.1.

6. CONCLUSION

We have proposed an approach using ICN to provide flexible and timely communication during and after a disaster. We identified the requirements for such an architecture by performing an extensive study of official reports and anecdotal reports in the aftermath of several actual disasters (including terrorist attacks). Our proposed architecture includes enhancements to the current ICN approaches for communication among authorities, especially for dynamically formed teams of first responders. A key contribution is the dynamic creation and evolution of incident related (sub) namespaces, recipient hierarchies, to represent the context and roles of first responders assigned to the disaster. A new enhanced forwarding strategy to support such recipient hierarchies is very useful to minimize the amount of message transmissions. With the help of a prototype and large scale trace-driven simulations, we highlight the quantitative benefits of CNS in terms of network load and latency as compared to an IP based solution.

We believe it is important to shift the focus on disaster communication from being an afterthought to being a first class citizen, exploiting emerging network architectures. Effective, convenient and timely communication could result in better outcomes, including fewer causalities.

7. ACKNOWLEDGEMENTS

This work is supported by the ICN2020 Project (Advancing ICN towards real-world deployment through research, innovative applications, and global scale experimentation), a research project supported jointly by the European Commission under its HORIZON 2020 (Grant Agreement No. 723014) and the National Institute of Information and Communications Technology (NICT) in Japan (Contract No. 184); and US NSF under Grant No. CNS-1455815. We thank our shepherd, Lixia Zhang, for her support and insightful feedback.

8. REFERENCES

[1] L. Assembly *et al.*, *Report of the 7 July review committee*. Greater London Authority, 2006.

[2] V. Jacobson *et al.*, "Networking Named Content," in *CoNext*, 2009.

[3] A. Venkataramani *et al.*, "MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture," *SIGCOMM CCR*, pp. 74–80, 2014.

[4] A. Anand *et al.*, "XIA: An Architecture for an Evolvable and Trustworthy Internet," in *HotNets*, 2011.

[5] J. Chen *et al.*, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.

[6] E. E. N. A. (EENA), "Ng112 project," http://www.eena.org/pages/ng-112.

[7] N. E. N. A. (NENA), "NG9-1-1 Project," http://www.nena.org/?NG911_Project.

[8] T. Hardie *et al.*, "LoST: A Location-to-Service Translation Protocol," RFC 5222, Aug. 2008.

[9] B. Rosen *et al.*, "Framework for Emergency Calling Using Internet Multimedia," RFC 6443, Dec. 2011.

[10] R. Barnes and M. Lepinski, "Using Imprecise Location for Emergency Context Resolution," IETF Draft draft-ietf-ecrit-rough-loc-05.txt, Jul. 2012.

[11] K. Wolf, "Location-to-Service Translation (LoST) Service List Boundary Extension," RFC 6197, Apr. 2011.

[12] L. Zhang *et al.*, "Named Data Networking (NDN) Project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[13] J. Chen *et al.*, "G-COPSS: A Content Centric Communication Infrastructure for Gaming," in *ICDCS*, 2012.

[14] J. Seedorf *et al.*, "Using ICN in disaster scenarios," IETF Draft draft-seedorf-icn-disaster-04, Oct. 2015.

[15] E. Monticelli *et al.*, "An Information Centric Approach for Communications in Disaster Situations," in *LANMAN*, 2014.

[16] I. Psaras *et al.*, "Name-Based Replication Priorities in Disaster Cases," in *NOM*, 2014.

[17] L. Wang *et al.*, "Data Naming in Vehicle-to-Vehicle Communications," in *NOMEN*, 2012.

[18] A. Tagami *et al.*, "Name-based Push/Pull Message Dissemination for Disaster Message Board," in *LANMAN*, 2016.

[19] S. Kim *et al.*, "Power-Saving NDN-Based Message Delivery Based on Collaborative Communication in Disasters," in *LANMAN*, 2015.

[20] "National Protection Framework," http://www.fema.gov/media-library/assets/documents/97350, 2014.

[21] B. Ahlgren *et al.*, "Design Considerations for a Network of Information," in *ReArch*, 2008.

[22] W. Adjie-Winoto *et al.*, "The Design and Implementation of An Intentional Naming System," *SIGOPS*, pp. 186–201, 1999.

[23] W. Fenner *et al.*, "XTreeNet: Scalable Overlay Networks for XML Content Dissemination and Querying," in *WCW*, 2005.

[24] V. Jacobson *et al.*, "VoCCN: Voice-over Content-Centric Networks," in *ReArch*, 2009.

[25] U. of Florida, *The Disaster Handbook – National Edition*, ch. 3.7 The Role of Government in a Disaster.

[26] F. E. M. A. (FEMA), "Continuity of Operations Planning Template for Federal Departments/Agencies," Sep. 2013.

[27] Marietta, Georgia, USA, "Marietta Police Department Bomb Incident Management Plan."

[28] Y. Wang *et al.*, "Wire Speed Name Lookup: A GPU-based Approach," in *NSDI*, 2013.

[29] J. Burke *et al.*, "Securing Instrumented Environments over Content-Centric Networking: The Case of Lighting Control and NDN," in *NOM*, 2013.

[30] C. E. Perkins, "Mobile IP," *Communications Magazine*, vol. 35, no. 5, pp. 84–99, 1997.

[31] J. Chen *et al.*, "CNS: Content-oriented Notification Service for Managing Disasters (Extended Material)," University of Göttingen, Germany. https://www.net.informatik.uni-goettingen.de/publications/1947/PDF, Technical Report IFI-TB-2015-04, Nov. 2015.

[32] Wikipedia, "Timeline of relief efforts after the 2010 Haiti earthquake," http://en.wikipedia.org/wiki/Timeline_of_relief_efforts_after_the_2010_Haiti_earthquake.

[33] datahub, "Haiti Crisis Map," http://datahub.io/dataset/ushahidi/resource/81d058a8-173a-49d9-8ce9-4edf5e7cafc9.

[34] F. Zhang *et al.*, "EdgeBuffer: Caching and Prefetching Content at the Edge in the MobilityFirst Future Internet Architecture," in *WoWMoM*, 2015.

[35] M. Piorkowski *et al.*, "Dataset of Mobility Traces of Taxi Cabs in San Francisco, USA," http://crawdad.org/epfl/mobility/20090224/, Feb. 2009.

7 Name Based Enhancement For Network Management

Exploiting ICN for Flexible Management of Software-Defined Networks

Mayutan Arumaithurai*, Jiachen Chen*, Edo Monticelli*, Xiaoming Fu* and K. K. Ramakrishnan‡
*Institute of Computer Science, University of Göttingen, Germany.
Email: {arumaithurai,jiachen,monticelli,fu}@cs.uni-goettingen.de
‡University of California, Riverside, CA, U.S.A. Email: kk@cs.ucr.edu

ABSTRACT

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. We observe that current SDN solutions pre-translate policy (what) into forwarding rules at specific switches (where). We argue that this choice limits the dynamicity, flexibility and reliability that a software based network could provide. Information Centric Networking (ICN) shifts the focus of networks away from being predominantly location oriented communication environments. We believe ICN can significantly improve the flexibility for network management. In this paper, we focus 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. We propose Function-Centric Service Chaining (FCSC), a solution that exploits ICN to provide flexibility in managing networks that utilize virtualization to dynamically place functions in the network as required. We use a real-world topology to compare the performance of FCSC and a more "traditional" SDN solution. We show that FCSC reacts to failures with fewer packet drops, adapts to new middleboxes more quickly, and maintains less state in the network.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Management

General Terms

Design; Management

Keywords

ICN; Service Chaining; Network Management; SDN; Network Function Virtualizaion; Middlebox

ICN'14, September 24–26, 2014, Paris, France.

ACM 978-1-4503-3206-4/14/09 ...$15.00.
http://dx.doi.org/10.1145/2660129.2660147 .

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. We envision 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. In this work, we make 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* [9, 19, 40] 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. We use 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 Content Delivery Network (CDN), Dynamic Site Accelerator (DSA [1]), TCP optimization over tunnel, *etc.*, in order to improve the perceived user experience [2].

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 [15, 16] 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) [14, 20] 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. We envisage 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 Software Defined Networking (SDN) [22, 33, 45] 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 [18, 27] 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 and BGP. But the controller has to keep track of the status of the middleboxes and the network.

We argue 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 §3.3 for detailed descriptions).

Information-Centric Network (ICN [4,21,44]) is a new networking paradigm that introduces ContentNames to decouple the user interests from data location. Following this line of thinking, we present *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. 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.

The key contributions of this work are:

- We exploit the combination of ICN with SDN to meet the dynamic requirements of service chaining. We propose 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, we show how FCSC compliments the current SDN solution in terms of network state amount, packet drop rate on node failure and overall latency.

The rest of the paper is organized as follows: §2 discusses previous work on service chaining and information-centric network; §3 provides detailed description on the service chaining scenario and the problems with the existing solutions. §4 describes the design rational of FCSC and §5 details upon the solution. §6 illustrates simulation results and conclusion is inferred in §7.

2. RELATED WORK

In this section, we briefly present existing work on service chaining, then present an overview of ICN and finally present existing work that involves both SDN and ICN.

2.1 Existing Solutions for Service Chaining

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

2.1.1 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 [6, 40] 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. [6] 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 [10, 17, 30, 31, 37, 39]. 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.

2.1.2 Policy-Based Routing (PBR):

Cisco's policy-based approach [13] 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.

2.1.3 SDN-based Service Chaining:

Several solutions have been presented that leverage SDN [22, 33, 45]. 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.

2.2 Information Centric Networking

Information Centric Networking (ICN) has been actively studied in recent years [4, 5, 11, 12, 21, 24]. ICN shifts the focus of the network from node location (IP, MAC, *etc.*) to data names. Such design enables name-based routing which forwards the requests of a specific name towards a best source of the data in terms of latency, available bandwidth, source load and *etc.* Named-Data Networking NDN [4, 21] is one of the popular ICN solutions. NDN uses human-readable, hierarchical names such as `/thisroom/projector` or `/icn/papers/FCSC. pdf`. The forwarding engines perform the longest-prefix matching in the FIB to find the next-hop router closer to the data provider. FCSC adopts the idea of ICN since the naming layer focuses more on the name of the functions rather than the location of the function instances. It is implemented on a model similar to NDN (naming and routing) but with some changes (no Pending Interest Table or reverse-path forwarding).

2.3 Works that Combine ICN and SDN

There are several works that try to explore the potential for combining ICN and SDN [29, 34, 38]. But most of these works use SDN technology to enable incremental deployment of ICN. To the best of our knowledge, this is the first work that tries to improve the performance of SDN via information-centric concept (ICN).

3. SCENARIO DESCRIPTION AND PROBLEM STATEMENT

In this section, we describe 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. We will use these as the basis to demonstrate the benefits of our proposed approach.

3.1 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 Deep Packet Inspection (DPI), policy, QoS, Network Address Translation (NAT), Dynamic Site Accelator (DSA), proxying, transparent caching, accounting and 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 [35].

3.2 Detailed Requirements

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

3.2.1 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*.

3.2.2 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.

3.2.3 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.

3.2.4 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 [32], the recovery time of a failure should be within 10s of milliseconds.

3.3 Limitations of Existing SDN Solutions

Current SDN solutions [22,33,45] perform better than policy based routing (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 [43].

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.

4. FCSC OVERVIEW

In this section, we start by reasoning the design choices we have made and then describe the whole architecture based on the design choices. Design details will be provided in §5.

4.1 Design Rationale

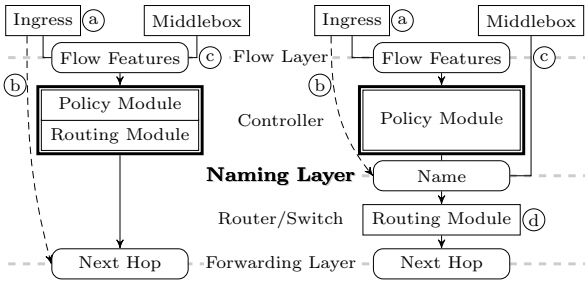
To achieve the requirements of flexibility, dynamicity and reliability as described above, we propose to add a naming layer (similar to ICN) to the current SDN architecture. Moreover, to improve the scalability of the system, we choose to put flow-state in the packet header rather than in the switches.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

4.1.1 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.

We find 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), we primarily incorporate 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 [36, 42], line-speed (hashed) name forwarding is achievable with existing hardware.



(a) SDN (b) FCSC

Figure 1: Architectural design of FCSC *vs.* SDN.

4.1.2 Flow State in the Packet Header

We note that 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, we choose 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 our solution, 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.

4.1.3 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. Our solution 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.

4.2 Architecture Description

Fig. 1 illustrates the logical separation of the architecture of FCSC compared to existing SDN solutions. As described

above, we add a *Naming Layer* 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, we describe 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↦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↦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 [41], IS-IS [8]) or logically centralized (*e.g.*, SDN).

b. 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.

c. 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 what we would 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.

d. 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 [32] precomputes 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*).

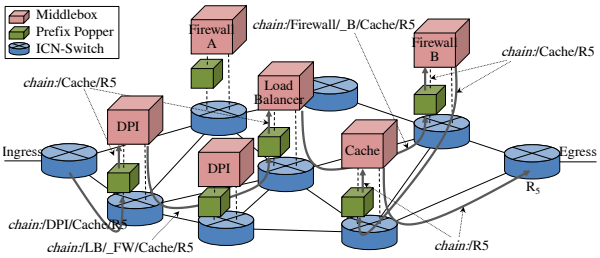


Figure 2: **Example of the name changing of a packet in FCSC.**

5. DESIGN DETAILS

In this section, we describe in detail how we design the architecture to ensure a highly efficient and scalable service delivery network.

5.1 Naming Strategy

We extend 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, we use the scheme identifier (as per URI Generic Syntax [7]) "*chain*" to represent the packets for service chaining. We can use other identifies like "*monitor*", "*ctrl*", *etc.*, 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, we 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. 2 illustrates the lifetime of a packet in our 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 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.

5.2 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, we provide a simple standard interface for the routing module to control the forwarding decision. We add a "cost" field 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. We discuss the benefits and issues of some possible routing solutions – generally categorized into 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 we only maintain function state.

Centralized/SDN solutions (*e.g.*, [33]) 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.*, [41]) 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.

We realize that both the centralized and the distributed routing methods face the difficulty of achieving real-time load balancing similar to [3]. 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. Our architecture can fully support such load balancers (see the example in §5.1).

5.3 Stateful Middleboxes

There might be some functions in the network that need to maintain state. 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. 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. 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 name from the packet header, we 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. 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.

5.4 Security

Security is another big 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. We do 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.

6. EVALUATION

In this section, we evaluate on a custom simulator (widely used in previous works [12] [11]) the dynamicity, reliability and scalability of FCSC with a distributed routing module and compare it to a relatively simple (physically) centralized SDN solution that is conceptually similar to [33, 45]. These approaches use the basic OpenFlow protocol constructs that is a controller interacting with network forwarding elements [28].

We recognize that there are approaches for decentralized SDN solutions like [23, 43], but the results from [25] 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 we use here can also be viewed as such a portion.

We first demonstrate the benefits of FCSC on a small synthetic topology (shown in Fig. 3a). Subsequently, we run a large-scale simulation on a real world topology to evaluate the scalability and the efficiency of our solution in a more realistic environment.

6.1 Study of FCSC Behavior

Fig. 3a shows a simple topology with multiple middleboxes. R_1-R_6 are FCSC capable switches. N_1-N_4 and DPI provide functions A and B and DPI as noted in the figure. Src and Dst are the source 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 $1000pps$.

We use several scenarios to compare the behavior of FCSC with the simple SDN solution:

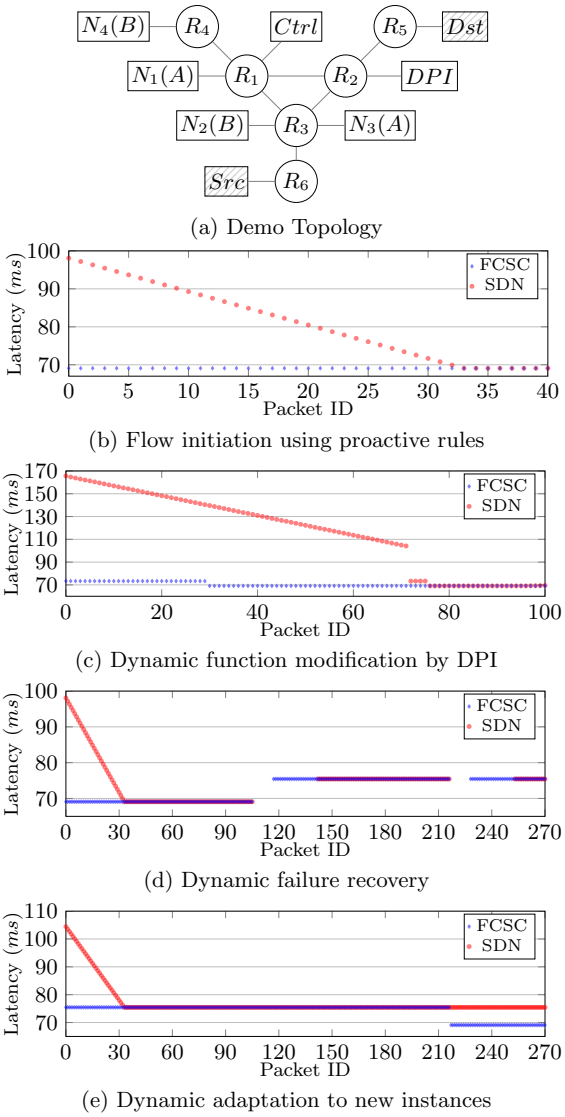
6.1.1 *Proactive rules for flow initiation:*

We first compare the initiation phase for both solutions. We compare FCSC with an SDN solution without proactive rules set up in the switches.

Fig. 3b 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. Because 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 of middlebox functions (*e.g.*, games).

We recognize that with proactive rules set up in the switches, SDN solutions can achieve lower latencies, as good as FCSC. However, we believe it still requires a lot more effort to precalculate the paths for different permutations as compared to our solution.

6.1.2 *Dynamic Policy change on Middleboxes:*



(a) Demo Topology

(b) Flow initiation using proactive rules

(c) Dynamic function modification by DPI

(d) Dynamic failure recovery

(e) Dynamic adaptation to new instances

Figure 3: Behavior of FCSC: Topology and Results.

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. 3c shows the latency of the first 90 packets in the flow. In FCSC, only the first 29 packets go through DPI with less than $75ms$ 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 $165ms$. Another 4 packets experienced a loop since the rules are not setup atomically.

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

6.1.3 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. We dispose N_3 at $150s$ and N_2 at $240s$ and see the packet loss and recovery time in FCSC and SDN.

Fig. 3d 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.

6.1.4 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. 3e 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 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.

6.2 Large Scale Evaluation

We adopt a slightly modified Rocketfuel topology [26] (Exodus, AS-3967, see Fig. 5) 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. We eventually get 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, we assign $100Mbps$ bandwidth to all the links.

We assume 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.

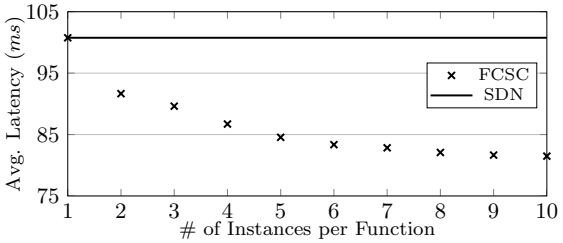


Figure 4: Benefit of increasing # of instances per function.

6.2.1 Varying number of function instances

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

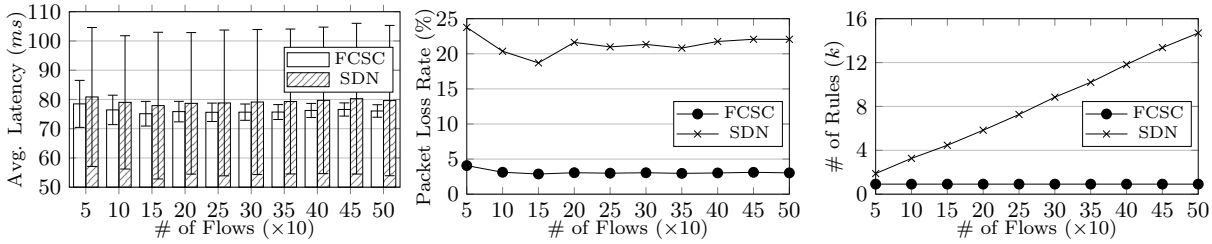
At the beginning of the simulation, we have 1 instance for each function initialized. Then, we start a new randomly located instance for each function every half a minute until the maximum number of instances is reached (the maximum number of instances for each run is shown in the x-axis in Fig. 4). 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 $30s$ and lasts until the end of the simulation. In the third run, a third instance is put into the network at time $60s$ and so on.

Fig. 4 shows the average latency vs. the number of instances eventually initiated for that simulation run. We see that FCSC can automatically adapt by making use of new instances that are closer, even for the ongoing flows and the average latency drops from $100.75ms$ to $91.66ms$ when adding a second instance. The latency is further reduced to around $85ms$ 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. We can also observe that the higher the number of instances, the lower the incremental benefit. In our scenario consisting of 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 our architectural proposal, and is part of our future work.

6.2.2 Varying number of flows & function instances

We now load the simulation with a varying number of flows (50, 100, 150, ..., 500). Each flow has its own arrival time (within the first $5min$), a sending rate in the range



(a) Latency (and 95% CI) vs. # of flows (b) % of packets lost vs. # of flows (c) # of rules vs. # of flows

Figure 6: Results for varying-flow simulation.

Figure 5: RocketFuel topology (Exodus, AS-3967).

($1.2Mbps$ - $11.9Mbps$) and duration ($0.05s$ to $91.24s$). We also randomly generated $1,151$ middlebox creation/failure events during the simulation period. We run the trace on both the FCSC and SDN solutions and compare the average latency for the flows, packet loss caused by middlebox failure, and the number of rules stored on the switches.

Fig. 6a 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. 6b 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.

FCSC capable 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 a n-tuple, potentially with wild-cards). Therefore, the number of rules do not change when we vary the number of flows (as shown in Fig. 6c). However the number of rules in SDN grows with the number of flows. We argue that our solution can be more scalable especially in a large network with millions of concurrent flows.

7. 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. By proposing FCSC, we explore the potential of using information-centric concepts within an SDN-based network management environment, especially focusing on service chaining. Using both synthetic and realistic topologies, we show 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). For our future work, we will explore better routing mechanisms that can fully exploit the benefits provided by FCSC.

8. ACKNOWLEDGMENTS

The research leading to these results has received funding from the EU-JAPAN initiative by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement No. 608518 (GreenICN), NICT under Contract No. 167 and the Volkswagen Foundation Project "Simulation Science Center". The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the Simulation Science Center project, the European Commission, or NICT.

9. REFERENCES

[1] http://blog.streamingmedia.com/2010/10/how-dynamic-site-acceleration-works-what-akamai-and-cotendo-offer.html.

[2] http://blog.streamingmedia.com/2011/12/its-official-akamai-to-acquire-content-good-for-akamai-bad-for-customers.html.

[3] J. Abley and K. E. Lindqvist. Operation of anycast services. In *IETF, RFC*, number 4786, 2006.

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

[5] S. Arianfar, P. Nikander, and J. Ott. On Content-centric Router Design and Implications. In *Re-Arch*, 2010.

[6] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *SIGCOMM*, 2004.

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

[8] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. In *IETF, RFC*, number 1195, 1990.

[9] B. E. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. In *IETF, RFC*, number 3234, 2002.

[10] I. Castineyra and M. Steenstrup. The Nimrod routing architecture. In *IETF, RFC*, 1992.

[11] J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan. G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications. In *ICDCS*, 2012.

[12] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan. COPSS: An Efficient Content Oriented Publish/Subscribe System. In *ANCS*, 2011.

[13] CISCO. Policy-based routing. Technical report.

[14] 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.

[15] J. Erman, A. Gerber, K. K. Ramakrishnan, S. Sen, and O. Spatscheck. Over the Top Video: The Gorilla in Cellular Networks. In *IMC*, 2011.

[16] J. Erman and K. Ramakrishnan. Understanding the Super-sized Traffic of the Super Bowl. In *IMC*, 2013.

[17] B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. *arXiv preprint*, 2006.

[18] B. Heller, R. Sherwood, and N. McKeown. The Controller Placement Problem. *SIGCOMM CCR*, pages 473–478, 2012.

[19] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *IMC*, 2011.

[20] J. Hwang, K. Ramakrishnan, and T. Wood. NetVM: high performance and flexible networking using virtualization on commodity platforms. In *NSDI*, 2014.

[21] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *CoNEXT*, 2009.

[22] 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 *SIGCOMM*, 2013.

[23] 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.

[24] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM CCR*, pages 181–192, 2007.

[25] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann. Logically centralized?: state distribution trade-offs in software defined networks. In *HotSDN*, 2012.

[26] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *IMW*, 2002.

[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM CCR*, pages 69–74, 2008.

[28] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM CCR*, pages 69–74, 2008.

[29] N. B. Melazzi, A. Detti, G. Mazza, G. Morabito, S. Salsano, and L. Veltri. An openflow-based testbed for information centric networking. In *FutureNetw*, 2012.

[30] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. In *IETF, RFC*, number 4423, 2006.

[31] A. Myles, D. B. Johnson, and C. Perkins. Mobile host protocol supporting route optimization and authentication. *JSAC*, pages 839–849, 1995.

[32] P. Pan, G. Swallow, A. Atlas, et al. Fast reroute extensions to RSVP-TE for LSP tunnels. In *IETF, RFC*, number 4090, 2005.

[33] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *SIGCOMM*, 2013.

[34] R. Ravindran, X. Liu, A. Chakraborti, X. Zhang, and G. Wang. Towards software defined ICN based edge-cloud services. In *CloudNet*, 2013.

[35] 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 *SIGCOMM*, 2012.

[36] W. So, A. Narayanan, and D. Oran. Named data networking on a router: fast and dos-resistant forwarding with hash tables. In *ANCS*, 2013.

[37] Stoica, Adkins, Ratnasamy, Shenker, Surana, and Zhuang. Internet Indirection Infrastructure. In *SIGCOMM*, 2002.

[38] M. Vahlenkamp, F. Schneider, D. Kutscher, and J. Seedorf. Enabling Information Centric Networking in IP Networks Using SDN. In *SDN4FNS*, 2013.

[39] M. Walfish and H. Balakrishnan. Untangling the Web from DNS. In *NSDI*, 2004.

[40] M. Walfish, J. Stribling, M. N. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. In *OSDI*, 2004.

[41] L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang. Ospfn: An ospf based routing protocol for named data networking. *Tech. Rep*, 2012.

[42] 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.

[43] M. Yu, J. Rexford, M. J. Freedman, and J. Wang. Scalable Flow-based Networking with DIFANE. In *SIGCOMM*, 2010.

[44] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[45] 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.

8 Conclusion and Outlook

In this last chapter, I provide a summary of all the work done in this thesis and also list some possible research items that could extend this work and/or are currently at a very early stage.

8.1 Summary

8.1.1 Content Oriented Publish Subscribe System

In this work, we presented and evaluated COPSS, an efficient pub/sub-based content delivery system exploiting the fundamental capability of CCN for efficient information dissemination. COPSS builds on existing proposals for CCN/NDN to provide pub/sub functionality. COPSS is designed to be scalable to a large number of publishers, subscribers and CDs. We recognize that a pub/sub platform needs to be able to accommodate users going offline, and allow them to retrieve content that has been published during the time the subscriber was offline. We also explicitly address the need for incremental deployment of CCN capability in traditional IP networks. We use trace-driven simulations to evaluate the COPSS architecture. COPSS reduces the aggregate network load and the publisher load significantly. The additional CCN 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. COPSS is substantially more efficient than existing pull-based CCN proposals (such as NDN) because of its inherent pub/sub capability.

8.1.2 Hybrid Content Oriented Publish Subscribe System

In this work, we present *Hybrid-Content Oriented Publish Subscribe System* (Hybrid-COPSS), an architecture to integrate CCN functionality with the current IP architecture. We present a detailed solution to integrate both the pub/sub and the query/response based content-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. In this work, we have addressed the 3-way tradeoff that arises when considering the incremental deployment of CCN, 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 we go more towards a pure IP environment; there is increased latency in a pure CCN environment because of the additional per-hop processing; finally there is an increase in processing cost for each CCN hop because of

the additional complexity. Our evaluations suggest that Hybrid-COPSS strives to achieve a proper balance in this trade-off by putting CCN functionality at key points and hash-based forwarding at the other routers. Moreover, we optimize the query/response dimension of Hybrid-COPSS and show that an RP based top down approach provides the best means for service providers to incrementally deploy caching-capable CCN nodes. The Hybrid-COPSS inter-domain solution recognizes the current challenges in having inter-domain IP multicast and overcomes it with the use of CCN overlay nodes at the domain edges. The inter-domain Hybrid-COPSS cuts inter-domain traffic almost by half even compared to our earlier CCN proposal, COPSS.

8.1.3 Gaming Over Content Oriented Publish Subscribe System

This work presents G-COPSS that functions as an efficient decentralized communication infrastructure for a gaming environment by leveraging the advantages of a content-centric network. We have implemented G-COPSS both on a testbed and a simulator. We obtained the micro-benchmark results and compared the performance of G-COPSS (over 60 players on the testbed) to that of an IP-server and NDN based approaches. We also show, using trace driven simulations for a system (over 400 players) that G-COPSS is able to outperform a pure IP-server based gaming environment in regards to aggregated network traffic and update latency. We also outperform the NDN approach, which depends on a query-response model for such a gaming application. We are currently working on algorithms for improving RP selection and creating a gaming environment that uses G-COPSS features to acquire real user behavior data and gaming traces.

8.1.4 Name Based Multicast Congestion Control Framework

Designing a congestion control mechanism for information delivery to large numbers of receivers is difficult, particularly with heterogeneous receiver bandwidths. It continues to be a challenge also with Information Centric Networks. Through emulation and an analytical model, we showed that heterogeneous receivers will get out-of-sync with existing receiver-driven, in-sequence congestion control mechanisms. To overcome the resulting inefficiencies, we proposed *Scalable and Adaptive Information Dissemination* (SAID). SAID allows receivers to request "any-next" packet, instead of the "next in-sequence" and thus delivers more packets on the first attempt. For missing packets, other receivers can provide "repair" packets even if the in-network caches no longer hold the content. Privacy and trust is maintained during the repair phase.

The evaluations show that SAID achieves efficiency and fairness on each path between the information provider and receivers. From a large scale simulation, SAID can reduce aggregate network load (by up to $\sim$46%) and transmission completion times (by more than 50%) compared to existing congestion control mechanisms. SAID also reduces completion time by $\sim$40% while only increasing network load by $\sim$10% compared to pgmcc, which

aligns the sending rate to the slowest receiver. Based on measurements on a prototype, SAID outperforms ICP, getting 50% higher aggregate throughput and almost twice the throughput of pgmcc. With the efficient repair of SAID, streaming applications have a much smaller stall time compared to the other mechanisms.

8.1.5 Name Based Disaster Communication Framework

We have proposed an approach using ICN to provide flexible and timely communication during and after a disaster. We identified the requirements for such an architecture by performing an extensive study of official reports and anecdotal reports in the aftermath of several actual disasters (including terrorist attacks). The proposed architecture includes enhancements to the current ICN approaches for communication among authorities, especially for dynamically formed teams of first responders. A key contribution is the dynamic creation and evolution of incident related (sub-)namespaces, recipient hierarchies, to represent the context and roles of first responders assigned to the disaster. A new enhanced forwarding strategy to support such recipient hierarchies is very useful to minimize the amount of message transmissions. With the help of a prototype and large scale trace-driven simulations, we highlight the quantitative benefits of *Content Notification System* (CNS) in terms of network load and latency as compared to an IP based solution.

We believe it is important to shift the focus on disaster communication from being an afterthought to being a first class citizen, exploiting emerging network architectures. Effective, convenient and timely communication could result in better outcomes, including fewer causalities.

Early stage work on how ICN could support DTN, i.e. fragmented environments, can be found in [32, 33, 34, 35, 36, 37, 38, 39, 40].

8.1.6 Name Based Enhancement For Network Management

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. By proposing FCSC, we explore the potential of using information-centric concepts within an SDN-based network management environment, especially focusing on service chaining. Using both synthetic and realistic topologies, we show 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).

8.2 Outlook

This dissertation work started at a time when ICN was at a nascent state with many open issues. The work undertaken in this thesis contributed solutions to many of these open issues. However, a lot more work is required to make ICN a reality and my team is continuing to contribute by participating in research, prototyping, large scale testing, real testbed based testing [44, 45, 46, 47]. Below, I list some of the existing/ongoing work and highlight how it could be extended.

8.2.1 Application and Service advances

8.2.1.1 Video Delivery

Video distribution will be one of the most important application in the future Internet. Due to its in-network caching and stateful forwarding, ICN technology is deemed to cope with the distribution of video content at scale, like and better than HTTP-based CDN. Accordingly, Bhat et al. [48] show preliminary results on the load balancing in ICN realized by utilizing multiple producers and in-network caching. Among the video streaming technologies, Dynamic Adaptive Streaming over HTTP (DASH) will surely be used in the future, while considering that Video Distribution majors like YouTube are already using it. This technique has also been introduced in the context of ICN [49, 50, 51]. Several proofs of concept have been examined, e.g., NetInf-enabled live video streaming [52] demonstrated at FIS Nordic Ski World Championship in 2015; Peer to Peer Live video streaming for cellular network [53]. These proofs of concepts clarified ICN's outstanding performance on the streaming latency and its great scalability in the real field test. Further work is required to study how ICN could support issues such as new forms of multimedia interactive features (e.g. multiple camera angles, augmented reality, interaction with sensors [54]), region access control, congestion-avoidance route/source selection, support of future 5G network and realistic Subscription Video OnDemand (S-VoD) platform with support for multi-region, multi-domain and multi-player video delivery.

8.2.1.2 Social Network

Social Network Applications like Facebook and WhatsApp are increasingly used for exchanging multimedia contents, like photos or videos. To support at scale the group-based multimedia delivery that these applications require, related companies are using CDNs. For instance Facebook is an Akamai customer, and it is also placing its own servers inside points of presence owned by ISPs to speed up delivery of its content to users in places around the world. In this framework, since an ICN could provide services similar to CDN one, it is worth to explore the possibility of using ICN also to support social network applications. For instance, [55] shows how an Internet Service Provider (ISP) can capitalize on ICN by deploying a social network messaging system at a fraction of the complexity of today's

systems. [56] presents a social network platform based on ICN. [57] uses ICN to deliver social contents and design a content caching policy related with the social behaviors. Further work is required to study how ICN could support social networks in areas such as context-awareness, geo-referencing, high quality multimedia contents, backend services, cloud and 5G/NFV network.

8.2.1.3 IoT

Big data analysis services based on data collected from IoT devices are promising applications and many IoT infrastructures for collecting sensed data from a huge number of IoT devices are proposed in [58]. However, most IoT infrastructures are cloud-based and IoT devices upload sensed data to pre-determined servers in clouds. These cloud-based infrastructures lack flexibility in choosing suitable IoT devices in dynamic environments. Since adopting ICN as a basis of the IoT infrastructure is promising in terms of flexibility of naming, traffic reduction by caching and content based security, many studies address research issues on designing ICN based IoT infrastructures. One issue is light-weight implementation on tiny devices and Bacelli et al. implemented tiny CCN code and evaluate the code in testbeds [59]. Another important issue is the scheme for collecting sensed data from a set of IoT devices which have interested attributes. Amadeo et al. [60] develop a mechanism for sending an Interest packet to multiple sources, but the mechanism is only applicable in single hop environments. Further work is required to study how ICN could support IoT in areas such as context/attribute based naming/communication, multi-source data retrieval coupled with content-based security, data-retrieval without specifying the exact names of IoT devices and how to manage a large number of IoT devices.

8.2.1.4 Resolution and Mapping

Although name resolution and routing in ICN is performed on the network layer, [61] successfully showed that there is a need for object resolution that maps the user inputs into the name of the object that network can use for retrieving the data. [61] proposed an architecture for object resolution system and also showed the feasibility of the solution with a prototype [62]. Similarly, as shown in the MobilityFirst [20] work on a Global Name Resolution Service (GNRS) [63], name resolution services could be very useful for ICN like architectures since they reduce the amount of routing updates required at routers and instead support as late a binding as feasible and are able to support mobility very well.

8.2.1.5 ICN and Virtualization

Nowadays, information centric routers, e.g. NDN and CCNx, are mainly software based and there are some initial literature studies on the use of ICN over virtualization infrastructures. Syrivelis et al. [64] propose a software defined information-centric network based in the

PURSUIT framework, whereas in [65], the authors make attempts to combine CCN/NDN with OPENFLOW to improve cache and content routing efficiency. In [66], a flexible transition solution between legacy IP network and ICN network is proposed with the help of SDN. Finally, in [67], an efficient integration of SDN and ICN is proposed based on a recently proposed future Internet architecture, named CoLoR [68]. Such infrastructures (ICN over virtualization) could make possible the deployment of ICN functions in a cloud data center, at the network edge (a.k.a. fog computing) [69], in a 5G network supporting resource slicing [70], etc. The dominant standard for deploying virtual network functions is the ETSI NFV [71], whose main implementation is OpenNFV, aka OPNFV [72]. FCSC [41] propose the use of ICN to complement SDN by providing a naming layer to support function name based forwarding and optimization. Currently, literature on the topic of ICN “over" virtualization infrastructure is not very extended and mostly concerns possible benefits, use cases, and applications (e.g. [69, 73, 74]). Further work to explore the issue from a point of view much more closer to the current NFV standardization activities is required.

8.2.2 Infrastructural advances

8.2.2.1 5G

5G (5th generation mobile networks) is the next major phase of mobile networks, which aims to further advance the performance of mobile communication considering the market conditions and the requirements in 2020 or later [75, 76]. 5G networks have a number of special requirements, e.g., the guarantee of user throughput of 1 Gbps everywhere, a user-plane latency of less than 1ms and a 1000-fold system capacity per km^2 compared to LTE. In order to meet these requirements, 5G systems have a number of new features, e.g., the millimetre wave utilization, the software-defined mobile core network and the device-to-device communication (D2D). ICN has the potential to utilize these features more effectively. ITU-T FG IMT-2020 [77] analysed ICN as one of the networking innovations required to support the development of 5G systems. Carofiglio at el. [78] quantified the opportunities of ICN for mobile backhaul network evolutions by analysing how much bandwidth can be saved. Tagami et al. [79] proposed a novel content delivery scheme that aggregates macro networks and small-area networks provided by base stations of the millimetre wave communication.

8.2.2.2 Congestion-aware Routing and Traffic Management

While literature provides several papers about ICN multipath forwarding and congestion control (e.g. [80, 81, 82, 83]), caching adds a new dimension to the network optimization problem. By accounting for path-level congestion in terms of bandwidth share at the bottleneck link, in [84] combine caching and congestion control. Similarly, in [85] authors use congestion feedback for the purpose of caching and demonstrate the benefit of such congestion caching to reduce network congestion and to increase user-centric performance. In [86], authors look at in-network caching from a different angle and question whether

caches can take on alternative roles and assist with the control of network congestion and in-network resource management. The VIP framework [87] jointly optimizes caching and multipath forwarding with a back-pressure algorithm. Finally, the impact of caching to fairness in resource allocation has been studied in [88] where it shows that LRU caching and AIMD congestion control discriminates against flows of unpopular content.

Traffic management was traditionally carried out as a distributed optimization algorithm to solve a network utility maximization problem [89, 90, 91] for elastic traffic. Video streaming traffic management is receiving an increased attention; with recent work focusing on HTTP adaptive streaming (HAS) or, in particular, dynamic adaptive streaming over HTTP (DASH) [92]. The challenge here is to adapt streaming representation at the application layer with limited feedback from lower layers and to overcome streaming buffer underrun problems [93, 94, 95]. While most of the recent works on streaming adaptation have focused on techniques to improve each streaming flow separately, [96] makes an effort to optimize HAS in a scenario where there are multiple streaming flows competing for bandwidth. Preliminary results on streaming over ICN are presented in [97, 49], where HAS is adapted to the context of ICN. Finally, the effect of caching to HAS streaming is considered in [50, 98, 99]. Given caching is a vital inherent component of ICN, it is important to fully integrate it with the other existing network functions.

8.2.2.3 Mobility

ICN is a promising architecture for the forthcoming 5G networks and naturally resolves various scalability problems of the centralized mobile architectures, such as the EPC (Evolved Packet Core) and the MobileIP in terms of mobility management. A centralized mobility management scheme, such as the MobileIP indirection, incurs large signalling overheads to send devices' location updates to an anchor. On the contrary, the request-response style of NDN communication naturally supports consumer mobility by allowing routers record the state of such a pair of request and response. However, naive ICN does not natively support producer mobility at-scale, which is needed for real-time communication among users. One possible approach to producer mobility is to introduce indirection into ICN/NDN [100, 101, 102] like EPC and MobileIP, but this inherits all the drawbacks of indirection. Another promising approach is an anchor-less approach such that a device's location is recorded at edge routers without using any anchor [103, 104, 105]. The anchor-less mobility management mechanism is designed by [104] by averaging state-full forwarding [106], where a device‘s location is recorded at all edge routers which it is connected.

However, the existing anchor-less mobility studies fail to address important issues raised by mobility management of cellular networks, despite being among the main parts of 5G networks. To name a few issues: i) how do we harmonize anchor-less mobility management and indirection of cellular networks; ii) signalling overheads caused by a paging mechanism of cellular networks are not negligible.

8.2.2.4 Caching

Caching of content has been a research topic for a long time, including for instance the area of web caching [107, 108] and P2P file sharing [109]. Content caching is one of the key features in ICN, which enables not only content-based caching but also chunk (or packet) level caching [110]. Chunk level caching achieves more flexibility [111], efficiency, and dynamism [112] in the network. The effectiveness of various in-network caching mechanisms in ICN has been thoroughly investigated in [113, 114, 115, 116]. A cooperative caching architecture that takes the lifetime of content into consideration is proposed in [117, 118, 119]. A multicast based packet caching architecture is designed in [120]. The impact of in-network caching to the content mobility is addressed in [121]. A performance analysis of partial caching is discussed in [122], whereas the effectiveness of content caching in multimedia applications is investigated in [123]. [124] exploits in-network caching to support collaborative downloading of content and offload traffic from a cellular to a cellular-assisted device-to-device network scenario. This is done by extending the ICN functionality with the usage of fountain coding. Finally, [125] exploits in-network caching in a way to preserve content over time in the case of disrupted networks in order to enable the usage of functional parts of the infrastructure, even when these are disconnected from the rest of the network.

Outside the caching mechanisms, the caching functionality of ICN itself has been compared with the content delivery networks (CDNs) as a competitor since both of them were designed to realize efficient content distribution over the network. On the other hand, the interconnection of ICNs and CDNs have been considered [44, 126] since the CDNs have already been vital network services.

There are still open issues in caching strategies in any-to-any group communications, especially in social content sharing, supporting producer mobility, and efficient video streaming. Also, caching scheme specialized for disaster and accident management is challenging to improve the reliability and dependability. On the other hand, CDNs can be viewed as multi-source content delivery mechanisms since they locate multiple points of presences (PoPs) as content repositories.

8.2.3 Prototying and experiments of real-world and large-scale tesbeds

In order to facilitate the deployment of ICN and to design and develop the key elements (framework, protocol, tools, APIs), a research-design-prototype-experiment driven approach is required. A lot of focus should therefore be put on prototyping and performing large scale experiments. Global testbeds such as the NDN testbed [127] and CUTEi [128] could be made use of to perform realistic tests. However, these testbeds are not suitable by itself to widely evaluate performance or to deploy new/early-stage experimental functionality (e.g., a new routing protocol). Therefore, local testbeds could be used to perform complementary experiments such as to test ICN functionality under heavy load,

having a large number of connections, supporting different qualities of video, etc. and to analyze ICN performance in varying network conditions, in a controlled environment. These testbeds need to be complimented with realistic experiments in order to obtain an understanding of ICN's performance in large scale and/or real-world scenarios. As part of our EU-Japan ICN2020 project [46] that I am currently leading, we are contributing to the development and experimentation process.

Bibliography

[1] Cisco Systems, Inc., "Cisco Visual Networking Index: Forecast and Methodology, 2014 – 2019 ," May 2015.

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

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *CoNext*, 2009.

[4] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM CCR*, pp. 181–192, 2007.

[5] S. Arianfar, P. Nikander, and J. Ott, "On Content-centric Router Design and Implications," in *Re-Arch*, 2010.

[6] J. Seedorf and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," RFC 5693 (Informational), Internet Engineering Task Force, Oct. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5693.txt

[7] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) Project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[8] "Named-data networking (ndn) project," http://named-data.org/.

[9] C. Esteve, F. Verdi, and M. Magalhaes, "Towards a new generation of information-oriented Internetworking architectures," in *ReArch*, 2008.

[10] "Pursuit project," www.fp7-pursuit.eu.

[11] K. V. D. Lagutin and S. Tarkoma, "Valencia FIA book 2010 Publish/Subscribe for Internet: PSIRP Perspective," 2010.

[12] C. E. R. S. A. P. Jokela, A. Zahemszky and P. Nikander, "LIPSIN: Line Speed Publish/Subscribe Inter-Networking," in *ACM SIGCOMM'09. Barcelona, Spain.* ACM, August 2009.

[13] V. Koptchev and V. Dimitrov., "Traffic and Congestion Control in a Publish/Subscribe Network," in *CompSysTech, Sofia, Bulgaria.*, June 2010.

[14] G. C. P. X. Vasilakos, V. A. Siris and M. Pomonis, "Proactive Selective Neighbor Caching for Enhancing Mobility Support in Information-Centric Networks," in *Proceedings of SIGCOMM, ICN Workshop, Helsinki, Finland.* ACM, 2012.

[15] "expressive internet architecture project," http://www.cs.cmu.edu/ xia/.

[16] "Content mediator architecture for content-aware networks project," http://www.comet-project.org/.

[17] "4ward project," http://www.4ward-project.eu/.

[18] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design Considerations for a Network of Information," in *ReArch*, 2008.

[19] "Sail project," http://www.sail-project.eu/.

[20] A. Venkataramani, J. F. Kurose, D. Raychaudhuri, K. Nagaraja, M. Mao, and S. Banerjee, "MobilityFirst: A Mobility-Centric and Trustworthy Internet Architecture," *SIGCOMM CCR*, pp. 74–80, 2014.

[21] Wikipedia, "7 July 2005 London bombings," https://en.wikipedia.org/wiki/7_July_2005_London_bombings.

[22] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan, "COPSS: An Efficient Content Oriented Pub/Sub System," in *ANCS*, 2011.

[23] J. Chen, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "Coexist: Integrating Content Oriented Publish/Subscribe Systems with IP," in *Proceedings of the ANCS Symposium*. ACM/IEEE, 2012.

[24] ——, "Coexist: A Hybrid Approach for Content Oriented Publish/Subscribe Systems," in *ICN*, 2012.

[25] ——, "G-COPSS: A Content Centric Communication Infrastructure for Gaming," in *ICDCS*, 2012.

[26] ——, "G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications," in *LANMAN*, 2011.

[27] ——, "SAID: A Scalable and Adaptive Information Dissemination Protocol in ICN," in *ACM ICN*, 2016.

[28] ——, "SAID: A Control Protocol for Scalable and Adaptive Information Dissemination in ICN," *CoRR*, vol. abs/1510.08530, 2015. [Online]. Available: http://arxiv.org/abs/1510.08530

[29] ——, "SAID: A Scalable and Adaptive Information Dissemination Protocol in ICN," University of Göttingen, Germany. https://www.net.informatik.uni-goettingen.de/publications/1912/SAID.pdf, Tech. Rep., 2014.

[30] ——, "Reliable Publish/Subscribe in Content-Centric Networks," in *Proceedings of SIGCOMM, ICN Workshop*. ACM, 2013.

[31] ——, "CNS: Content-oriented Notification Service for Managing Disasters," in *ACM ICN*, 2016.

[32] ——, "CNS: Content-oriented Notification Service for Managing Disasters (Extended Material)," University of Göttingen, Germany. https://www.net.informatik.uni-goettingen.de/publications/1947/PDF, Technical Report IFI-TB-2015-04, Nov. 2015.

[33] J. Seedorf, M. Arumaithurai, A. Tagami, K. K. Ramakrishnan, and N. B. Melazzi, "Using ICN in disaster scenarios," IETF Draft draft-seedorf-icn-disaster-04, Oct. 2015.

[34] E. Monticelli, M. Arumaithurai, I. Psaras, X. Fu, and K. K. Ramakrishnan, "Combining Opportunistic and Information Centric Networks in Real World Applications," *IEEE/KuVS NetSys PhD Forum*, Mar. 2015.

[35] T. Yagyu, K. Nakamura, T. Asami, K. Sugiyama, A. Tagami, T. Hasegawa, and M. Arumaithurai, "Demo: Content-Based Push/Pull Message Dissemination for Disaster Message Board," in *Proceedings of the 2nd International Conference on Information-Centric Networking (ICN) (Demonstration)*. ACM, 2015, pp. 205–206.

[36] K. Sugiyama, A. Tagami, T. Yagyu, T. Hasegawa, M. Arumaithurai, and K. K. Ramakrishnan, "Multipath Support for Name-based Information Dissemination in Fragmented Networks," in *Proceedings of the 2nd International Conference on Information-Centric Networking (ICN) (Poster)*. ACM, 2015, pp. 201–202.

[37] ——, "Name-Based Information Dissemination for Fragmented Networks in Disasters," in *COMSNETS*, 2015.

[38] E. Monticelli, B. M. Schubert, M. Arumaithurai, X. Fu, and K. K. Ramakrishnan, "An Information Centric Approach for Communications in Disaster Situations," in *LANMAN*, 2014.

[39] I. Psaras, L. Saino, M. Arumaithurai, K. K. Ramakrishnan, and G. Pavlou, "Name-Based Replication Priorities in Disaster Cases," in *NOM*, 2014.

[40] J. Chen, M. Arumaithurai, X. Fu, and K. Ramakrishnan, "CNS: A Content-Centric Notification Service," in *Proceedings of the 21st International Conference on Network Protocols (ICNP) (Demonstration)*. IEEE, 2013, pp. 1–2.

[41] M. Arumaithurai, J. Chen, E. Monticelli, X. Fu, and K. K. Ramakrishnan, "Exploiting ICN for Flexible Management of Software-Defined Networks," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014.

[42] A. T. Y. P. K. R. X. F. G. P. Sameer G Kulkarni, Mayutan Arumaithurai, "Name Enhanced SDN Framework for Service Function Chaining of Elastic Network Functions," in *Poster Session, IEEE INFOCOM 2016, San Franscisco, U.S.A*, 2016.

[43] E. M. X. F. Mayutan Arumaithurai, Jiachen Chen and K. Ramakrishnan, "Prototype of an ICN Based Approach For Flexible Service Chaining in SDN," in *Demonstration Session, IEEE INFOCOM 2015, Hongkong, China*, 2015.

[44] "Greenicn," http://www.greenicn.org/.

[45] "List of GreenICN Publications," http://www.greenicn.org/deliverables/publications/.

[46] "Icn2020," http://www.icn2020.org/.

[47] "PDF of all Mayutan's Publications," http://www.greenicn.org/deliverables/publications/.

[48] D. Bhat, C. Wang, A. Rizk, and M. Zink, "A load balancing approach for adaptive bitrate streaming in Information Centric networks," in *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[49] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, "An experimental analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks," in *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, July 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6607500

[50] Y. Liu, J. Geurts, J.-C. Point, S. Lederer, B. Rainer, C. Muller, C. Timmerer, and H. Hellwagner, "Dynamic adaptive streaming over CCN: A caching and overhead analysis," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 3629–3633. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6655116

[51] C. W. S. Lederer, "Adaptive Video Streaming over ICN," 2015. [Online]. Available: https://tools.ietf.org/html/draft-irtf-icnrg-videostreaming-05

[52] A. M. Malik, B. Ahlgren, and B. Ohlman, "NetInf Live Video Streaming for Events with Large Crowds," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 209–210.

[53] A. Detti, B. Ricci, and N. Blefari-Melazzi, "Mobile peer-to-peer video streaming over information-centric networks," *Computer Networks*, vol. 81, pp. 272–288, 2015.

[54] J. Burke, "ICN as an Enabler for New Forms of Multimedia Experience," Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on, 2015.

[55] B. Mathieu, P. Truong, W. You, and J.-F. Peltier, "Information-centric networking: a natural design for social network applications," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 44–51, 2012.

[56] J. Kim, M.-w. Jang, B.-J. Lee, and K. Kim, "Content centric network-based virtual private community," in *Consumer Electronics (ICCE), 2011 IEEE International Conference on*. IEEE, 2011, pp. 843–844.

[57] C. Bernardini, T. Silverston, and O. Festor, "Using social network information into icn."

[58] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE COMMUNICATION SURVEYS & TUTORIALS*, vol. 17, no. 4, pp. 2347–2376, Sep 2015.

[59] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Waehlisch, "Information centric networking in the iot: Experiments with ndn in the wild," in *ACM ICN 2014*, Sep 2014, pp. 316–321.

[60] M. Amadeo, C. Campolo, and Molinaro, "Multi-source data retrieval in iot via named data," in *ACM ICN 2014*, Sep 2014, pp. 67–76.

[61] 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 *The 21st IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, 2015.

[62] ——, "Prototype of an Architecture for Object Resolution Services in Information-Centric Environment," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 203–204.

[63] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 247–258.

[64] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis, and L. Tassiulas, "Pursuing a software defined information-centric network," in *Software Defined Networking (EWSDN), 2012 European Workshop on*. IEEE, 2012, pp. 103–108.

[65] J. V. T. et. al., "Controller-based routing scheme for named data network," Technical Report, 2012. [Online]. Available: http://www.gta.ufrj.br/ftp/gta/TechReports

[66] A. Chanda, C. Westphal, and D. Raychaudhuri, "Content Based Traffic Engineering in Software Defined Information Centric Networks," *ArXiv e-prints*, 2013.

[67] H. Luo, J. Cui, Z. Chen, M. Jin, and H. Zhang, "Efficient integration of software defined networking and information-centric networking with color," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, 2014, pp. 1962–1967.

[68] H. Luo, Z. Chen, J. Cui, H. Zhang, M. Zukerman, and C. Qiao, "Color: An information-centric internet architecture for innovations," *IEEE Network*, vol. 28, no. 3, pp. 4–10, 2014.

[69] Ravindran, R. and Xuan Liu and Chakraborti, A. and Xinwen Zhang and Guoqiang Wang, "Towards software defined icn based edge-cloud services," in *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*, Nov 2013, pp. 227–235.

[70] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network store: Exploring slicing in future 5g networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2015, pp. 8–13.

[71] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 90–97, 2015.

[72] "OpenNFV project page," http://www.opnfv.org.

[73] N. Niebert, I. El Khayat, S. Baucke, R. Keller, R. Rembarz, and J. Sachs, "Network virtualization: A viable path towards the future internet," *Wireless Personal Communications*, vol. 45, no. 4, pp. 511–520, 2008.

[74] B. Ahlgren, P. Aranda, P. Chemouil, S. Oueslati, L. M. Correia, H. Karl, M. Sollner, A. Welin *et al.*, "Content, connectivity, and cloud: ingredients for the network of the future," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 62–70, 2011.

[75] The 5G Infrastructure Public Private Partnership, "White Papers," https://5g-ppp.eu/white-papers/.

[76] Association of Radio Industries and Businesses, "ARIB 2020 and Beyond Ad Hoc Group White Paper : Mobile Communications Systems for 2020 and beyond," http://www.arib.or.jp/english/20bah-wp-100.pdf, October 2014.

[77] ITU-T, "Focus Group on IMT-2020," http://www.itu.int/en/ITU-T/focusgroups/imt-2020/Pages/default.aspx.

[78] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Scalable mobile backhauling via information-centric networking," in *Proceedings of IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, April 2015, pp. 1–6.

[79] A. Tagami, K. Yokota, C. Sasaki, and K. Yamaoka, "Splitting Control-User Plane on Communication Protocol for Spotty Network," in *Proceedings of ACM*

International Workshop on Mobility in the Evolving Internet Architecture (MobiArch), Paris, France, September 2015, pp. 26–31.

[80] A. Detti, C. Pisa, and N. Blefari Melazzi, "Modeling multipath forwarding strategies in information centric networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, April 2015, pp. 324–329.

[81] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, "Multipath Congestion Control in Content-Centric Networks," in *The 2nd IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2013)*, 2013.

[82] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal Multipath Congestion Control and Request Forwarding in Information-Centric Networks," in *IEEE International Conference on Network Protocols*, Oct. 2013.

[83] A. Detti, C. Pisa, and N. Blefari-Melazzi, "Multipath forwarding strategies in Information Centric Networks with AIMD congestion control," *CoRR*, vol. abs/1412.2204, 2014. [Online]. Available: http://arxiv.org/abs/1412.2204

[84] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware Caching and Search in Information-centric Networks," in *Proceedings of the 1st International Conference on Information-centric Networking*, ser. ICN '14. New York, NY, USA: ACM, 2014, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/2660129.2660145

[85] D. Nguyen, K. Sugiyama, and A. Tagami, "Congestion price for Cache Management in Information-Centric Networking," in *Proceedings of INFOCOM WRKSHPS*, 2015.

[86] I. Psaras, L. Saino, and G. Pavlou, "Revisiting Resource Pooling: The Case for In-Network Resource Sharing," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. New York, NY, USA: ACM, 2014, pp. 24:1–24:7. [Online]. Available: http://doi.acm.org/10.1145/2670518.2673875

[87] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks," in *Proceedings of the 1st International Conference on Information-centric Networking*, ser. ICN '14. New York, NY, USA: ACM, 2014, pp. 117–126. [Online]. Available: http://doi.acm.org/10.1145/2660129.2660151

[88] D. Saucez, I. Cianci, L. Grieco, and C. Barakat, "When AIMD meets ICN: A bandwidth sharing perspective," in *Networking Conference, 2014 IFIP*, June 2014, pp. 1–9.

[89] S. H. Low and D. E. Lapsley, "Optimization Flow control – I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 861–874, December 1999.

[90] F. P. Kelly, "Mathematical modelling of the Internet," *Mathematics unlimited-2001 and beyond*, pp. 685–702, 2001.

[91] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan 2007. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4118456

[92] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *MultiMedia, IEEE*, vol. 18, no. 4, pp. 62–67, April 2011. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6077864

[93] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626296

[94] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 325–338, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787486

[95] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 1, pp. 326–340, Feb 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6704839

[96] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 4, pp. 719–733, April 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6774592

[97] C. Sieber, T. Hossfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, "Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 1318–1323. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6573184

[98] P. Juluri and D. Medhi, "Cache'N DASH: Efficient Caching for DASH," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 599–600. [Online]. Available: http://doi.acm.org/10.1145/2785956.2790015

[99] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP Adaptive Streaming: Friend or Foe?" in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA:

ACM, 2014, pp. 31:31–31:36. [Online]. Available: http://doi.acm.org/10.1145/2578260.2578270

[100] F. Hermans, E. Ngai, and P. Gunningberg, "Global source mobility in the content-centric networking architecture," in *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, Jun. 2012, pp. 13–18.

[101] D. Kim, J. Kim, Y. Kim, H. Yoon, and I. Yeom, "Mobility Support in Content Centric Networks," in *Proceedings of ACM ICN workshop 2012*, Aug. 2012.

[102] L. Wang, O. Waltari, and J. Kangasharju, "Mobiccn: Mobility Support with Greedy Routing in Content-Centric Networks," in *Proceedings of IEEE Globecom 2013*, Dec. 2013, pp. 2069–2075.

[103] Y. Zhang, H. Zhang, and L. Zhang, "Kite: A Mobility Support Scheme for NDN," in *Proceedings of ACM ICN 2015*, Sep. 2015, pp. 179–180.

[104] J. Augé, G. Carofiglio, G. Grassi, G. P. Luca Muscariello, and X. Zeng, "Anchor-less Producer Mobility in ICN," in *Proceedings of ACM ICN 2015*, Sep. 2015, pp. 189–190.

[105] M. Ishino, Y. Koizumi, and T. Hasegawa, "Leveraging Proximity Services for Relay Device Discovery in User-provided IoT Networks," in *Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT 2015)*, Dec. 2015.

[106] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A Case for Stateful Forwarding Plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, July 2013.

[107] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, Sep 2002.

[108] S. Podlipnig and L. Böszörmenyi, "A Survey of Web Cache Replacement Strategies," *ACM Computer Surveys*, vol. 35, no. 4, pp. 374–398, Dec. 2003. [Online]. Available: http://doi.acm.org/10.1145/954339.954341

[109] F. Lehrieder, G. Dan, T. Hossfeld, S. Oechsner, and V. Singeorzan, "Caching for BitTorrent-Like P2P Systems: A Simple Fluid Model and Its Implications," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, pp. 1176–1189, Aug 2012.

[110] S. Arianfar, P. Nikander, and J. Ott, "Packet-level caching for information-centric networking," in *ACM SIGCOMM, ReArch Workshop*, 2010.

[111] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 13–18.

[112] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, "Offloading cellular networks with Information-Centric Networking: The case of video streaming," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2012, pp. 1–3.

[113] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 55–60.

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

[115] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *IEEE 21st International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2012, pp. 1–7.

[116] Y. Kim and I. Yeom, "Performance analysis of in-network caching for content-centric networking," *Computer Networks*, vol. 57, no. 13, pp. 2465–2482, 2013.

[117] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, March 2012, pp. 316–321.

[118] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2012, pp. 268–273.

[119] S. Guo, H. Xie, and G. Shi, "Collaborative forwarding and caching in content centric networks," in *NETWORKING 2012*. Springer, 2012, pp. 41–55.

[120] K. Katsaros, G. Xylomenos, and G. C. Polyzos, "MultiCache: An overlay architecture for information-centric networking," *Computer Networks*, vol. 55, no. 4, pp. 936–947, 2011.

[121] X. Vasilakos, V. A. Siris, G. C. Polyzos, and M. Pomonis, "Proactive selective neighbor caching for enhancing mobility support in information-centric networks," in *Proceedings of the second edition of the ICN workshop on Information-centric networking*. ACM, 2012, pp. 61–66.

[122] L. Wang, S. Bayhan, and J. Kangasharju, "Optimal chunking and partial caching in information-centric networks," *Computer Communications*, vol. 61, no. 1, pp. 48–57, May 2015.

[123] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig, "Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand

Workloads," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. New York, NY, USA: ACM, 2014, pp. 363–376. [Online]. Available: http://doi.acm.org/10.1145/2674005.2675003

[124] G. Parisis, V. Sourlas, K. Katsaros, W. Chai, and G. Pavlou, "Enhancing Multi-source Content Delivery in Content-Centric Networks with Fountain Coding," in *CoNEXT CCDWN Workshop*, 2015.

[125] V. Sourlas, L. Tassiulas, I. Psaras, and G. Pavlou, "Information resilience through user-assisted caching in disruptive Content-nbnCentric Networks," in *IFIP Networking Conference (IFIP Networking), 2015*, May 2015, pp. 1–9.

[126] J. Seedorf, J. Peterson, S. Previdi, R. van Brandenburg, and K. Ma, "Cdni request routing: Footprint and capabilities semantics," https://tools.ietf.org/html/draft-ietf-cdni-footprint-capabilities-semantics-09, Nov. 2015.

[127] "NDN Testbed project page," http://named-data.net/ndn-testbed.

[128] A. Hitoshi, L. Ruidong, and C. Nakjung, "Container-Based Unified Testbed for Information-Centric Networking," *IEEE Network Magazine*, no. 6, 2014.