University of California

Los Angeles

Support Mobile and Distributed Applications with Named Data Networking

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Zhenkai Zhu

2013

Abstract of the Dissertation

Support Mobile and Distributed Applications with Named Data Networking

by

Zhenkai Zhu

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Lixia Zhang, Chair

The Internet is becoming increasingly mobile. As the the price of smartphones, tablets, and other portable devices becoming more and more affordable, the mobile access to the Internet is becoming the norm. Meanwhile, the current Internet architecture is increasingly challenged by emerging communication patterns. While IP was designed to solve the problem of carrying a point-to-point conversation between two entities, in today's dominant applications, such as video streaming, file sharing, and social networking, users are more interested in obtaining desired content rather than talking to a specific node. Such trends can be expected to continue in the near future and call for a reexamination of the current Internet architecture.

Named Data Networking (NDN) is a proposed future Internet architecture that uses data names instead of host addresses for data delivery. The new architecture incorporates principles that have made the IP protocol suite widely adopted and globally scaled (e.g., the hourglass design and end-to-end principle), but changes the fundamental layer of the architecture to one better suited to modern networks and emerging communication patterns.

In this dissertation, we study how to support mobile and distributed applica-

tions through Named Data Networking. Specifically, we focus on addressing two complementary questions: one is how to provide flexible, secure, and yet simple mobility support that serves the applications' need in the mobile environment, and the other is to how to design distributed applications that can fully exploit the benefits brought by NDN's architectural shift.

The first question is motivated by the fact that mobility support in the current Internet still struggles to satisfy the applications' need even though mobility has become a fundamental characteristic of today's Internet. We address this question by providing a new perspective on mobility support in NDN that addresses the weakness in the existing IP mobility solutions as well as utilizes the lessons learned in IP mobility research. By aligning the mobility support with the data-centric nature of the applications, the name-based data retrieval in NDN design, and the broadcast nature of the wireless media, not only does the new approach address the concerns in today's IP mobility solutions, but it also integrates mobility support using the same approach to cover all types of networks in the mobile environment, including ad hoc and delay-tolerant networks.

The second problem is motivated by the observation that distributed applications, such as group text messaging, file sharing, multimedia conferencing, and joint editing, have penetrated into our daily lives and drastically changed the communication patterns, calling for a new Internet architecture and new application design patterns that are freed from the constraints imposed by IP's point-to-point communication model. In this work, we take an application-driven approach to explore the new design patterns that can fully exploit the new opportunities brought by NDN. We propose ChronoSync, an efficient and completely distributed dataset state synchronization protocol to simplify the designs of distributed applications. ChronoSync leverages the flexible naming in NDN to simplify the task of maintaining up-to-date knowledge of a dataset, and exchanges the knowledge among all parties in a compact crypto digest form. Differences in dataset usually can be

inferred by comparing the digests, and can be efficiently propagated to all parties using NDN's built-in data multicast capability. We also propose a completely distributed and data-centric security design to achieve the goals of providing data provenance and access control to distribute applications in the absence of a central controller. We validate the proposed new design patterns by developing a distributed file sharing application, ChronoShare, that is based on ChronoSync protocol and secured in a data-centric way. Together with the new mobility support approach, our work represents a step towards a new direction of providing useful building blocks in supporting mobile and distributed applications.

The dissertation of Zhenkai Zhu is approved.

Mihaela van der Schaar

Songwu Lu

Mario Gerla

Lixia Zhang, Committee Chair

University of California, Los Angeles

2013

To my parents and my wife...

for their unconditional love

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my academic advisor, Dr. Lixia Zhang, for her constant support and guidance throughout my dissertation. From her I learn a persistent, open, and inquiring aptitude toward both research and life. I also thank Dr. Ryuji Wakikawa, whose ideas and advice helped my work on mobility support research, and Mr. Van Jacobson, whose seminal ideas have continuously inspired me, especially for my work on Named Data Networking. I am also grateful to my committee members, Dr. Songwu Lu, Dr. Mario Gerla, and Dr. Mihaela van der Schaar, for their valuable time, comments, and encouragements on my research. Also, I would like to acknowledge Alexander Afanasyev and others in Internet Research Laboratory at UCLA for the enjoyable collaborations and for their valuable discussions on various research topics. It is my pleasure to acknowledge and thank Chaoyi Bian who visited our lab and worked together with me for 9 months. Finally, I would like to thank my parents who have provided me an immeasurable supply of support; and most of all, my wife Li, for her love, trust, and encouragement during this long and colorful journey.

Vita

2007	Intern at Siemens, Beijing.
2008	B.E. (Automation), Tsinghua University.
2009	Intern at Toyota InfoTechnology Center, Mountain View.
2010	Teaching Assistant, Computer Science Department, UCLA.
2010	Intern at Toyota InfoTechnology Center, Mountain View.
2011	Intern at PARC, Palo Alto.
2011	M.S. (Computer Science), UCLA.
2012	Intern at LinkedIn, Mountain View.
2008–present	Graduate Student Researcher, Computer Science Department, UCLA.

Publications

Lixia Zhang, Ryuji Wakikawa, Zhenkai Zhu, "Supporting Mobility in the Global Internet", *ACM Mobicom Micnet Workshop*, 2009.

Zhenkai Zhu, Ryuji Wakikawa, Lixia Zhang, "Supporting Mobility For Internet Cars", *IEEE Communications Magazine*, May 2011.

Cheshire Stuart, Zhenkai Zhu, Ryuji Wakikawa, Lixia Zhang, "Understanding Apple's Back to My Mac Service", *IETF RFC 6281*, 2011.

Zhenkai Zhu, Ryuji Wakikawa, Lixia Zhang, "A Survey of Mobility Support In the Internet", *IETF RFC 6301*, 2011.

Zhenkai Zhu, Ryuji Wakikawa, Lixia Zhang, "SAIL: A Scalable Approach for Wide-Area IP Mobility", *IEEE INFOCOM Mobiworld Workshop*, 2011.

Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, Lixia Zhang, "ACT: An Audio Conference Tool Over Named Data Networking", *ACM SIGCOMM ICN Workshop*, 2011.

Zhenkai Zhu, Ryuji Wakikawa, Stuart Cheshire, Lixia Zhang, "Home As You Go: An Engineering Approach to Mobility-Capable Extended Home Networks", *AINTECT*, 2011.

Zhenkai Zhu, Paolo Gasti, Yanbin Lu, Jeffery Berke, Van Jacobson, Lixia Zhang, "A New Approach to Securing Audio Conference Tools", *AWFIT*, 2011.

Zhenkai Zhu, Chaoyi Bian, Alexander Afanasyev, Van Jacobson, Lixia Zhang, "Chronos: Serverless Multi-User Chat Over NDN", *NDN TR0008*, October 2012.

Zhenkai Zhu, Alexander Afanasyev, "Lets ChronoSync: Decentralized Dataset State Synchronization in Named Data Networking", *under submission*, 2013.

Zhenkai Zhu, Alexander Afanasyev, Lixia Zhang, "A New Perspective on Mobility Support", *under submission*, 2013.

CHAPTER 1

Introduction

The Internet is increasingly mobile. More than 1 billion [Mil13] smartphones are already in use and about 100 billion mobile applications have been downloaded by May, 2013 [app, goob]. As the the price of smartphones, tablets, and other portable devices becoming more and more affordable, one can confidently state that the mobile access to the Internet is becoming the norm, rather than the exception as we regarded it just a few years ago. Meanwhile, the current Internet architecture is increasingly challenged by emerging communication patterns. IP was designed to solve the problem of carrying a point-to-point conversation between two entities. However, today the dominant applications that contribute the most of Internet traffic are video streaming, file sharing, and social networking [san13], in which users are more interested in obtaining desired content rather than talking to a specific node. Such trends can be expected to continue in the near future and call for a reexamination of the current Internet architecture in order to address the challenges of supporting applications with new communication patterns in an era of mobile.

Named Data Networking (NDN) [ZEB10] is a proposed future Internet architecture that uses data names instead of host addresses for data delivery. The new architecture incorporates principles that have made the IP protocol suite widely adopted and globally scaled (e.g., the hourglass design and end-to-end principle), but changes the fundamental layer of the architecture to one better suited to modern networks and emerging communication patterns.

In this dissertation, we study how to support mobile and distributed applications through Named Data Networking. Specifically, we focus on addressing two complementary questions: one is how to provide flexible, secure, and yet simple mobility support that serves the applications' need in the mobile environment, and the other is how to design distributed applications that can fully exploit the benefits brought by NDN's architectural shift.

The first question is motivated by the fact that mobility support in current Internet still struggles to satisfy the need of applications even though mobility has become a fundamental characteristic of today's Internet. Over the last two decades, many efforts have been devoted to developing solutions for IP mobility support, which addresses two basic questions: how to deliver packets to the mobile node and how to maintain the transport and higher layer connections despite of the mobile's location changes. Yet, although a wide variety of protocols have been proposed, some of which have become the Internet standards, there is still no wide deployment of one or a set of IP mobility solutions. Concerns were raised about the inflexible communication model, the weak security measures, the potential suboptimal data path, etc.. Furthermore, these solutions are based on the assumption that a mobile node is always connected to the infrastructure, and its movement only results in different connecting points. This is a rather limiting assumption, as in reality mobility often leads to intermittent connectivity (e.g. vehicles on the road) or opportunistic ad hoc connectivity among a set of mobile nodes, which are currently handled by two separate branches of networking research, delay-tolerant networking (DTN) and mobile ad hoc network (MANET), respectively. However, such separation creates a great hurdle in developing applications that can work under any type of networks, as each branch has produced their own set of solutions that are largely orthogonal and incompatible with each other. As a result, an application running on the mobile node may need to switch from a TCP-based application protocol over 4G to probably a UDP-based, DTN protocol

when the connectivity condition changes. Such frustrating limitations highlight the need of a new mobility support architecture that is able to support mobile communications under varying connectivity conditions, which a mobile usually experiences when roaming.

Can one provide flexible, secure, and yet simple mobility support that serves the applications' need in the mobile environment? Our answer is a resounding yes. In this work, we first study the existing IP mobility support solutions in order to understand the design space of mobility support and identify the limitations of the current solutions. We then take a further step forward and assess the overall picture of mobility support in NDN. By aligning the mobility support with the data-centric nature of the applications, the name-based data retrieval in NDN design, and the broadcast nature of the wireless media, we provide a new perspective on mobility support that addresses the weakness in the existing IP mobility solutions as well as utilizes the lessons learned in IP mobility research. Not only can the NDN architecture address the concerns in today's IP mobility solutions, but it also can integrate mobility support using the same approach to cover all types of networks in the mobile environment, including ad hoc and delay-tolerant networks.

The second question is motivated by the observation that distributed applications, such as group text messaging, file sharing, multimedia conferencing, and joint editing, have penetrated into our daily lives and drastically changed the communication patterns, calling for a new Internet architecture and new application design patterns that are freed from the constraints imposed by IP's point-to-point communication model. While such applications operate in terms of content with the need to disseminate data among multiple parties, the current Internet architecture forces them to communicate by discovering locations and establishing point-to-point communication channels, resulting in unnecessary tussles and inefficient data distribution. To cope with the constraints imposed by IP, the existing

distributed applications either resort to a centralized paradigm or peer-to-peer solutions. However, the former approach, while being straightforward, results in single point of failure and centralized control of data; and the latter approach requires building a sophisticated application overlay structure and implementing application level data multicast, which is often undermined by the mis-matching between the overlay network and the underlying network topology.

In this work, we take an application-driven approach to explore the new design patterns that can fully exploit the new opportunities brought by NDN. When developing expeditionary distributed applications over NDN, we observed the ubiquitous need of efficient and robust dataset synchronization support and also the need to secure the applications in a distributed manner. To that end, we propose ChronoSync, an efficient and completely distributed dataset state synchronization protocol to simplify the designs of distributed applications. ChronoSync leverages the flexible naming in NDN to simplify the task of maintaining the up-to-date knowledge of a dataset, and exchanges the knowledge among all parties in a compact crypto digest form. Differences in dataset usually can be inferred by comparing the digests, and can be efficiently propagated to all parties using NDN's built-in data multicast capability. We also propose a completely distributed and data-centric security design to achieve the goals of providing data provenance and access control to distribute applications in the absence of a central controller. We validate the proposed new design patterns by developing a distributed file sharing application, ChronoShare, that is based on ChronoSync and secured in a data-centric way. Together with the new mobility support approach, our work represents a step towards a new direction of providing useful building blocks in supporting mobile and distributed applications development.

The rest of the dissertation is organized as follows. In Chapter 2, we briefly describe the mobility support problem in IP and also the concepts of NDN that are essential to describe our work. We review the existing IP mobility support

solutions in Chapter 3 to understand the solution space and to shed light on future efforts. In Chapter 4, we describe our new perspective on mobility support with NDN. In Chapter 5, we explore the new design patterns for distributed applications over NDN, presenting ChronoSync protocol and the data-centric security approach. We conclude the dissertation in Chatper 6.

CHAPTER 2

Background

In this chapter we provide the background for IP mobility problem and its solution space. We also provide background on the Named Data Networking architecture, which encompasses a conceptually simple yet transformational architectural shift from today's focus on where – addresses and hosts – to what – the content that users really care about, and provides a foundation on which our work is based.

2.1 IP Mobility

The problem of IP mobility dated back to the birth of IP networks, when the majority, if not all, of the hosts were stationary. Taking this fact for granted, early protocol designs often rest on the assumption that a host can be uniquely identified by its IP address, which is not subject to change over a relatively long time. For instance, a TCP connection is identified by a well-known 5-tuple: source IP address, source port, destination IP address, destination port, and the protocol type. However, this assumption no longer holds once the hosts start to roam around, prompting researchers and practitioners to proposes various designs to remedy the problem.

2.1.1 IP mobility support problem

The basic question in IP mobility support is how to send data to a moving receiver. For simplicity, henceforth we call the moving receiver *"mobile node"* (or a *"mobile"*

for short), and the host that sends data to a mobile the *"correspondent node"* (CN)[1].

Due to IP's host-based, point-to-point communication paradigm, the IP mobility support solutions generally aim to solve essentially two problems: find the new location of a mobile and keep the data communications uninterrupted despite of the move. To achieve these goals, the mobility support solutions have to have three essential components:

1. A stable IP address as an identifier for a mobile. This is to uphold the assumption that a host can always be identified by its stable IP addresses, based on which various protocols, including TCP, have been designed.

2. A locator, which is usually an IP address representing the mobile's current location. This is required by IP's host-to-host delivery model.

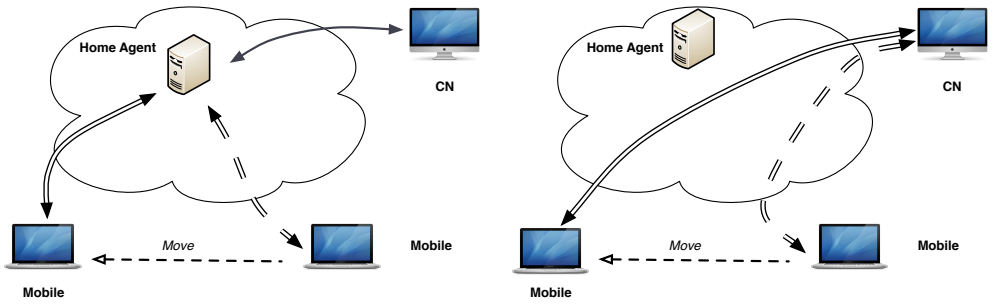
3. A mapping between the locator and the identifier.

2.1.2 Mobile IP: an example of IP mobility solution

Mobile IP (MIP) [Per96] is probably the best known mobility support protocol. The Internet Engineering Task Force (IETF) developed the first MIP standard in 1996, and subsequently defined different versions of the MIP standard for IPv4 and IPv6 networks. Although these standards differ in details, the high-level design is the same. Here we illustrate the basic design using Mobile IPv6 [JPA04].

Each mobile is assigned a home agent, from which it acquires its home address. The home address (HoA) remains the same regardless of the mobile's movement and will be used in the place where stable IP address is expected. As the mobile moves, it also obtains care-of-addresses (CoAs) from the access routers. The home agent is notified whenever the CoA of the mobile changes, such that it always

[1]Note that CN could be a mobile too.



(a) Base mode. (b) Route optimization mode.

Figure 2.1: Mobile IPv6. The double-stroked arrows represent the IPv6-in-IPv6 tunnel, and the dashed double-stroke arrow represent the tunnel before the mobile moves.

maintains the current HoA to CoA mapping. A CN sends data to a mobile using its HoA as the destination address. As a result, the packets are routed to the mobile's home agent, which in turn forwards the packets to the mobile by encapsulating them with the mobile's CoA as the destination. Vice versa, when the mobile sends data to the CN, it tunnels the data to the home agent, which decapsulate the packets and forward them to the CN. Figure 2.1a depicts the Mobile IPv6 operations described above. For this part, the CN operates on the normal IPv6, and does not need to understand Mobile IPv6.

Depending on the locations of the mobile, its home agent, and the CN, the packets may be forwarded along a triangle path. Hence, Mobile IPv6 includes measures, called *"route optimization"* (see Figure 2.1b), to allow the mobile to directly communicate with a CN, if the CN also understands Mobile IPv6, after the initial contact has been made with with the assistance of the home agent. Whenever the mobile acquires a new CoA, in addition to notifying the home agent, it also notify the CN, enabling CN to also maintain an up-to-date mapping between the mobile's HoA and CoA. As a result, the mobile and the CN can directly communication by establishing an IPv6-in-IPv6 tunnel between the two,

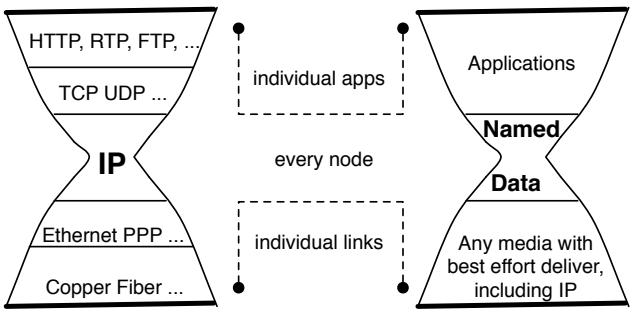


Figure 2.2: The thin waist of the network stack is changed from IP to named data in NDN

which is updated whenever the mobile changes its CoA, bypassing the home agent.

2.2 Named Data Networking

With the challenge from the changing communication patterns, where users and applications operate in term of content, today's Internet's requirements to communicate by discovering and specifying locations often create tussles. To better accommodate the emerging patterns of communications, Jacobson, Zhang, et al. have proposed Named Data Networking (NDN) architecture in recent work [JST09, ZEB10] to evolve the Internet. The thin waist, as in today's IP architecture, is the centerpiece of the NDN architecture, as show in Figure 2.2. This change, though seeming simple in first glance, leads to significant difference between IP and NDN in their operations of data delivery. In this section, we briefly cover the basic concepts in NDN upon which our work rests.

2.2.1 NDN packet types

There are two types of packets in NDN, namely Interest and Data, as shown in Figure 2.3.

The Interest packet is sent when a consumer requests data. It carries a name,

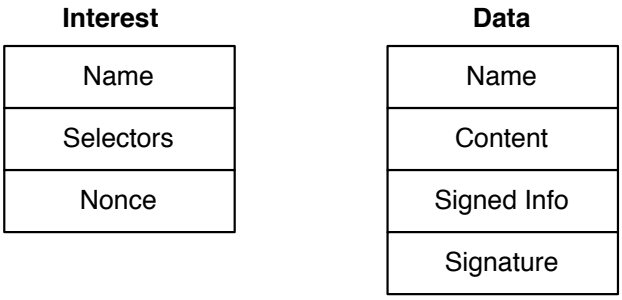


Figure 2.3: NDN packet types

which indicates the desired data, a selector field, which can specify preferences and other restrictions if more than one Data packets can satisfy the Interest, and a nonce.

A Data packet can be used to satisfy an Interest as long as the name carried in the Interest is a prefix of that of the Data. Besides the name and the actual content, a Data packet also includes *"signed info"*, which contains additional information such as publisher's key name, the timestamp, etc., and the signature signed by the publisher.

2.2.2 Hierarchical names

Each piece of data in NDN is named, using a hierarchical structured name that is unique in a specific scope. For example, this thesis may have the name `/ucla.edu/cs/zhenkai/thesis.pdf`, where `/` indicates a boundary between name components.

The hierarchy is useful in allowing applications to represent relations between data pieces. Often, an application instance would be assigned a name prefix and publishes its data under the assigned prefix. It also enables name-based routing to scale, in that the routing announcements can be aggregated. Furthermore, hierarchical names also facilitate the *"scoping"*, that is, some names may only

be meaningful in a specific network domain and are not meant to leak to other domains. For instance, `/ndn/broadcast` prefix indicates that the Interest for names under it should be broadcasted in the NDN testbed [Tes], but should not be forwarded beyond the boundary of the testbed.

2.2.3 Data-centric security

Instead of securing the communication channels as of today's Internet, NDN secure the data itself. Each piece of data is cryptographically bound to its name, signed by the publisher's private key. The signature, along with a trust management system that can verify the publisher's public key, enables applications to determine the provenance and integrity of the data, regardless of how (and from where) the data is obtained.

Retaining the end-to-end approach, NDN offers publishers, consumers, and applications great flexibility in choosing or customizing their trust models. As a result, the data packets in NDN are meaningful independent of where they come from, and can be cached inside the network to satisfy future requests.

2.2.4 Receiver-driven communications

Communications in NDN are driven by the receivers. To receive data, a consumer has to explicitly solicit it by sending out an Interest packet with the desired data name. A router remembers the interface from which the Interest comes from, and forwards it towards the data producer according to *"Forwarding Information Base"* (FIB), which is populated by some name-based routing protocol. Once a Data packet with the desired name has been reached, it is propagated back to the consumer following the reverse path taken by the Interest. Note that one Interest can only retrieve one Data, and thus it maintains a strict flow balance.

The Interests that are yet to be satisfied are stored in the *"Pending Interest*

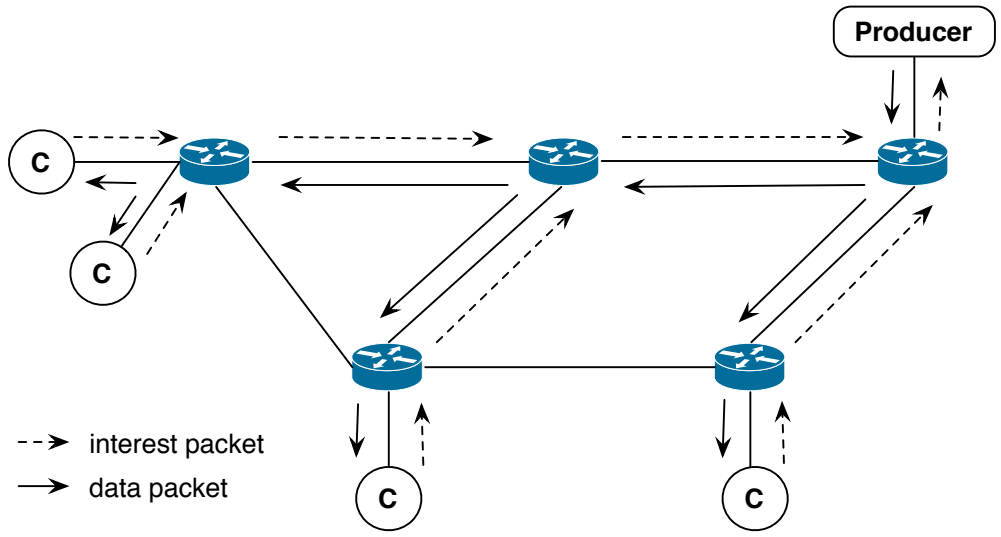


Figure 2.4: NDN naturally supports multicast data delivery. Dashed arrows represent Interests for the same data and solid arrows represent the multicasted data following the reverse paths of the Interests.

Table" (PIT) of the router. When multiple Interests for the same data are received from downstreams, the router only forwards the first one towards the upstream. Nevertheless, the set of interfaces from which Interests for the same name are received is recorded in the PIT entry. When the Data arrives, the router looks up the PIT, finds the matching entry, and forwards the Data to all interfaces listed. Therefore, multicast data delivery is naturally supported by NDN, as illustrated in Figure 2.4.

2.2.5 Intelligent data plane

In NDN, one fundamental change from IP, whose data plane is stateless, is the PIT records maintained at each router. Each PIT entry indicates the expectation of a Data packet, and is removed after a matching Data comes or timeout occurs. This per-packet state information makes NDN's data plane adaptive in handling network failures, and effective in utilizing network resources.

Aside from the multicast data delivery, the per-packet state allows routers to monitor packet delivery performance of different interfaces and detect duplicate Interest or Data packets, which ensures no loop would happen. Together, the data plane feedback and loop-free forwarding enable the NDN routers to make decisions on how to forward Interests through multiple interfaces, effectively support service selection, load balancing, and fast failure detection as well as alternative paths exploring. This capability, which we call *"Forwarding Strategy"*, is of immense significance especially in providing better services for mobile devices with multiple interfaces (WiFi, 4G, Ethernet, etc.).

CHAPTER 3

A Study of IP Mobility Support

The Internet community has been working on IP mobility support research and standardization since the early '90s. Yet, new issues continue to arise and new solutions continue to be developed to address them, making one wonder how much more we have yet to discover about the problem space as well as the solution space.

As a result, we believe that a thorough study of the proposed solutions to IP mobility support on the table can help us not only identify their commonalities and differences but also clarify issues yet to be addressed and shed insight on the efforts to support mobility in the evolving Internet architecture.

3.1 A Review of IP Mobility Solutions

In this section, we review existing IP mobility support protocols as listed in Table 3.1. During the course we show that the myriad of different designs of IP mobility support are merely different approaches to provide mapping between the mobile's identifier and its locator, which, as stated in Section 2.1.1, is the central piece of all designs.

3.1.1 Forerunners in early 90s

At the beginning of 1990s, researchers started to realize the need to provide support for mobile devices to continuously access networks. Consequently, several protocols that casted significant influence on later designs were proposed, among

Protocol	Year	Protocol	Year
Columbia	1991	TIMIP	2001
Virtual IP	1991	M-SCTP	2002
LSR	1993	HIP	2003
Mobile IP	1996	Connexion	2004
MSM-IP	1997	ILNPv6	2005
Cellular IP	1998	Global HAHA	2006
HMIP	1998	PMIP	2006
FMIP	1998	BTMM	2007
HAWAII	1999	WINMO	2008
NEMO	2000	LISP-Mobility	2009
E2E	2000		

Table 3.1: IP mobility solutions studied, listed roughly in chronicle order

which the Columbia protocol [IDM91], Virtual IP [TYT91], and a protocol based on IP Loose source routing (LSR) [BP93] are representative.

3.1.1.1 Columbia protocol

This protocol was designed to provide mobility support on a campus. A router named Mobile Support Station (MSS) is set up in each wireless cell and serves as the default access router for all mobile nodes in that cell. Each MSS also knows how to reach other MSSs (e.g., all MSSs could be in one multicast group, or a list of IP addresses of all MSSs could be statically configured).

A mobile node obtains an IP address from a special IP prefix, and the mobile node uses this IP address regardless of the cell to which it belongs. Each MSS keeps a tracking list of mobile nodes that are currently in its cell by periodically broadcasting beacons. When receiving beacons, the mobile replies to the MSS

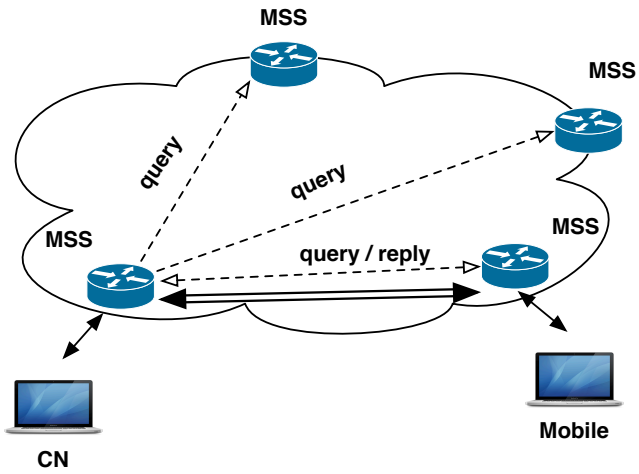


Figure 3.1: An example communication scenario in Columbia protocol

with a message containing its IP address, and also the IP address of the previous MSS if the beacon came from a new MSS. In the latter case, the new MSS is responsible to notify the old MSS that a mobile has left its cell.

When a corresponding node (CN)[1] sends a packet to a mobile node, the packet goes to the MSS nearest to the CN, which proceeds to perform one of the following two actions:

- deliver the packet directly if it is currently serving the mobile node;
- otherwise, broadcast a query to all other MSSs if there is no information about the mobile node in the cache; the reply to the query is cached; it then tunnels the packet to the MSS where the reply comes from, which then decapsulates and delivers it to the mobile node.

An example communication scenario is depicted in Figure 3.1.

3.1.1.2 Virtual IP

This design has two basic ideas:

[1]See Section 2.1.1 for the definition of a correspondent node

1. a packet carries both identifier and locator for a node;

2. the identifier is an IP address assigned from an IP prefix announced by the home network of a mobile node, where the mapping between the identifier and locator is kept.

The IP header is modified to allow packets sent by a mobile to carry two IP addresses: a virtual IP address (identifier) and a regular IP address (locator). Every time the mobile node changes its location, it notifies the home network with its new IP address. A mobile's virtual IP never changes, and is used in protocols such as TCP to keep the sessions undisrupted.

Without prior knowledge of a mobile, the CN first uses the mobile's virtual IP address as the destination IP address, i.e. the locator is set to be the same as the identifier. As a result, the packet goes to the mobile's home network and the home agent redirects the packet to mobile by replacing the locator field with the mobile's current address.

To alleviate the problem of triangle routing, the design lets CNs and routers cache the identifier-locator mapping carried in the packets. The cache is purged if timeout happens or the destination gateway router sends a control message indicating that the cached locator is no longer valid for the mobile.

3.1.1.3 LSR-based protocol

In this protocol, each mobile has a designated router, called Mobile Router, that manages its mobility. A Mobile Router assigns an IP address (used as an identifier) for each mobile it manages and announces reachability to those IP addresses.

Another network entity in the LSR design is Mobile Access Station (MAS), through which a mobile gets its connectivity to the Internet. The mobile node reports the IP address of its current serving MAS (locator) to its Mobile Router.

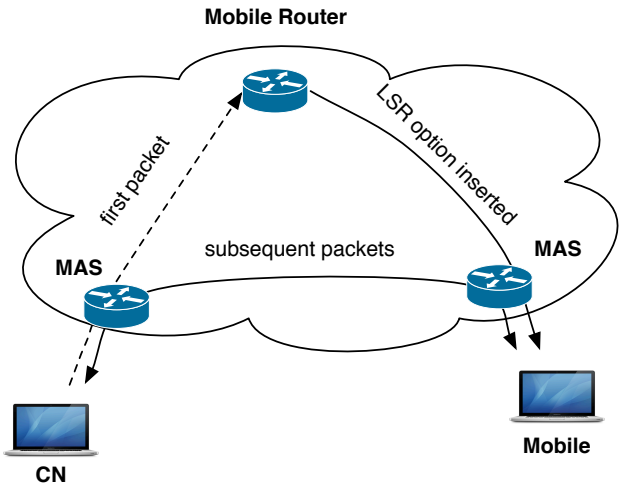


Figure 3.2: Communication in LSR-based protocol

The CN uses the identifier to reach the mobile node in the first place. If the CN and the mobile node are attached to the same MAS, the MAS simply forwards packets between the two; otherwise, the packet from CN is routed to the Mobile Router of the mobile. The Mobile Router looks up the mappings to find the serving MAS of the mobile node, and inserts the LSR option into the IP header of the packet with the IP address of the MAS on it. In this way, the packet is redirected to the MAS which then delivers the packet to the mobile. To this point, the locator of the mobile node is already included in the LSR option, and the two parties can communicate directly by reversing the LSR option in the incoming packet. Figure 3.2 illustrates the above process.

3.1.2 Mobile IP and its extensions

Since its debut as the IETF standard in 1996, Mobile IP, which is briefly introduced in Section 2.1.2, has attracted a lot of attentions from both researchers and industrial practitioners, and has been deployed in a number of commercial systems. Due to its popularity and the status as the "de facto" IP mobility solution, various extensions and enhancements have been proposed, some of them

standardized, to improve the performance under different circumstances.

3.1.2.1 Extensions for better handoff performance

For instance, Hierarchical Mobile IP [SCE08] aims to scale Mobile IP and improve the handoff performance by handling mobility in a local region locally. That is, the mobile, if roams within a region served by a local Mobility Anchor Point (MAP), only updates its location to MAP; only when moving to a new region served by another MAP will the mobile node notify its home agent the IP address of the new MAP. In this way, the handoff delay is reduced due to the physically close distance to local MAP, and the bottleneck problem of the home agent is alleviated to some extent by reduced frequency of updating from the mobile nodes.

Another attempt to reduce handoff delay and disruption was made by FMIPv6 [Koo09], which mainly includes two measures.

1. It enables a mobile to detect a new network and formulate a prospective CoA when it is still connected to the current network.

2. The mobile node, while trying to register the new CoA to home agent and optionally to the CN, can request the previous router to redirect the packets destined to it to the new CoA through IP tunneling, so that the disruption during the handoff process is mitigated.

3.1.2.2 Extensions for network mobility

One family of extensions that draws much attention is Network Mobility (NEMO) [DWP05], which aims to support mobility scenarios where a group of hosts move together (e.g., passenger devices in cars, trains, or airplanes). Under such circumstance, it would be rather inefficient to handle the mobility for each individual host separately. Furthermore, the burst of location update messages by the group could

further deteriorate the situation.

Thus, NEMO propose that each moving network employs a mobile router that is responsible for handling the mobility for all hosts on board. Conceptually the mobile router works in a way similar to a normal mobile node in Mobile IP. However, instead of having a single HoA, it obtains an IP block from the home agent, and assigns IP addresses in the block to the hosts on board. While traveling, all traffic to and from the mobile network flows through a bidirectional tunnel between the mobile router and the home agent. As a result, only the mobile router needs to update the home agent when the access network changes, and the mobility is transparent to the group of hosts on board.

3.1.2.3 Extensions for better data path

The motivation for such extensions, represented by Global Home Agent to Home Agent Protocol (G-HAHA) [WVM06], is to eliminate the triangle routing problem of Mobile IP without forcing the CNs to support mobility. The route optimization procedure included in Mobile IP standard requires the CNs to understand the Mobile IP protocol and always maintain the mapping between the CoA and HoA of a mobile, which is not always possible or desirable.

G-HAHA utilizes a group of home agents, which, distributed over a geographically and topologically large region, announce the same home prefix to the routing system, effectively creating a large scale anycast group. Each mobile is assigned an HoA from the anycast prefix, and can register with any of the home agents. During the roaming, a mobile node M always register to the closest home agent H from its access network, which accepts and request and informs all other home agents about the mapping [M, H]. When a CN sends packets to M, the packets are routed towards the closest home agent to CN as a result of anycast. This home agent in turn lookup the serving home agent H for the mobile M in the

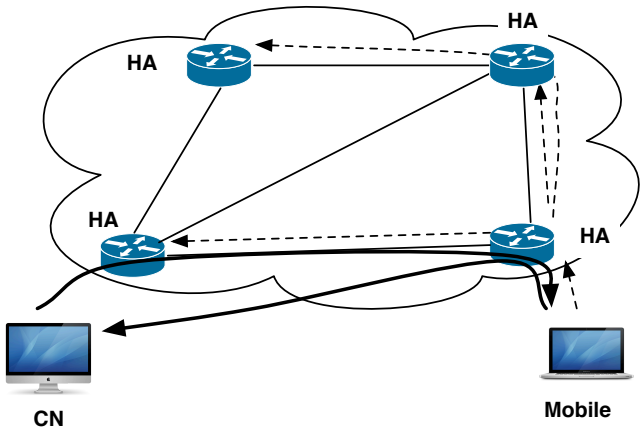


Figure 3.3: G-HAHA operations. Dashed arrows represent the location update messages, and solid arrows represent the data flows.

mapping database, which is always kept synchronized in all home agents. It then tunnels the packets to H, which finally decapsultes and delivers them to M. One the reverse direction, the home agent near M forwards packets directly to CN after receiving them through IP tunnel from M. Figure 3.3 demonstrates the above described operations.

With a reasonably large number of home agents deployed and distributed widely, it is with high probability that there is a home agent sufficiently close to the mobile or the CN, effectively eliminating the triangle routing.

3.1.2.4 Extensions for backward compatibility

Some mobile operator went further to take direct control over mobility support by network and requires no mobility awareness in the mobile devices (and thus the legacy devices can be supported). Proxy Mobile IP (PMIP) [GLD08] is one of such proposals. PMIP introduces two new types of network entities, the local mobility anchor (LMA) and the mobile access gateway (MAG). Each mobile node is assigned an LMA within an operator's network, which assigns the mobile a home prefix and plays a role similar to that of the home agent in MIP. The MAG, as a

result of operators desire, first acquires a mobile's identity and verifies whether it is authorized for access. In addition, it also monitors the mobile's attaching and detaching events, and updates the LMA accordingly. To hide a mobile's roaming from itself, a MAG also advertise the home prefix of each currently attached mobile, so that a mobile consequently treats the MAG as its default router. This enables PMIP to imitate an entire operator's network as a single link for each mobile, so that a mobile does not detect any change with respect to its layer 3 attachment as long as it roams within the operator's network.

3.1.3 Host-route based protocols

Another approach adopted by a group of protocols [Val99, RPT02, GEN01] is to set up a host route for each mobile within a network. This approach is network-based, in that the mobility is handled by network entities, rather than the mobile devices themselves.

3.1.3.1 Cellular IP

Cellular IP [Val99] is designed as a local mobility solution to work in conjunction with Mobile IP. When entering a Cellular IP based network, a mobile reports to its Home Agent the IP address of the network's border router as its CoA, and uses a locally assigned IP address when it roams in this network. To track the locations of the mobiles, routers in the network monitor the packets originated from each mobile and maintain a hop-by-hop reverse path from each router to each mobile. Idle mobiles send dummy packets to the border router with a low frequency to help routers maintain the reverse paths to reach them. To keep the overhead low, only a subset of the routers maintain reverse paths for idle mobile nodes. The routers use timeout to remove obsolete path states.

When the border router receives a packet destined to an idle mobile, whose

precise location is not tracked, it sends out a query using scoped flooding. If a receiving router knows how to reach the destination mobile, it forwards the query to the corresponding interface; otherwise it forwards the query to all its interfaces except the one the query came from. Once the mobile receives the query, it sends a route-update message to the border router, setting up a precise reverse path with short timeout value through all the routers along the data path, via which the border router forwards packets to the mobile.

3.1.3.2 HAWAII

Similarly, Handoff-Aware Wireless Access Internet Infrastructure (HAWAII) [RPT02] also aims to provide network-based local mobility support.

Different from Cellular IP, where the precise route to a mobile is set up on-demand, HAWAII always maintains a precise route between the gateway router and a mobile. When the mobile moves, HAWAII dynamically modifies the route to the mobile by installing a host-based forwarding entry on the routers located along the shortest path between the old and new base stations of the mobile. Alternatively, a new sub-path between the mobile and the cross-over router can be established to avoid the stretched route. Here, the cross-over router is the router at the intersection of two paths, one between the gateway and the old base station and the second between the old base station and the new base station.

In HAWAII, the mobile only periodically sends refresh messages to the base station, and the base station along with other routers take care of the path maintenance.

3.1.3.3 TIMIP

Terminal Independent Mobile IP (TIMIP) [GEN01] integrates the design of Cellular IP and HAWAII.

On one hand, it refreshes the routing paths with dummy packets if the mobile node is idle. On the other hand, handoff within a domain results in the changes of routing tables in the routers. Besides, the IP layer is coupled with layer 2 handoff mechanisms, and special nodes can work as Mobile IP proxies for legacy mobiles that do not support Mobile IP. Thus, as long as the mobile roams within the domain, the legacy node has the same degree of mobility support as a Mobile-IP-capable node.

3.1.4 Global routing based protocols

Supporting mobility through dynamic routing is conceptually simple: one can announce the location of a mobile throughout the entire Internet so that all routers know how to reach the mobile. However, announcing routes for individual mobiles is practically impossible, and thus the work in this category focus on network mobility.

3.1.4.1 Connexion

Boeing deployed the Connexion service [And06], based on BGP routing, during 2004–2006 that enabled travelers on board of a plane to access the Internet. The design assigns a permanent /24 IP address prefix to each mobile network (in this case an airplane) and uses BGP to propagate the reachability to these prefixes. When an airplane with a prefix P moves from an access router R_a on the ground to another access router R_b, R_a withdraws the prefix P and R_b announces P, thus the attachment point changes of each airplane are propagated to the rest of the Internet. Consequently this design raises routing scalability concerns: if the number of mobile networks becomes large, the amount of rapid BGP updates will also increase proportionally which could lead to severe router overload.

3.1.4.2 WINMO

Wide-Area IP Network Mobility (WINMO) [HMY08] aims to address the routing scalability concern of Connexion. Like Connexion, it also assigns each mobile network a stable prefix. However it reduces the total number of updates originated from mobile networks by orders of magnitude through two new approaches. First, WINMO uses various heuristics to limit the propagation scope of routing updates caused by mobile movements. As a result, not all routers may know all mobiles' precise locations. Second, WINMO adopts the basic idea in MIP by assigning each mobile network a "home" in the following way. WINMO assigns mobile network prefixes out of a small set of well-known Mobile Prefixes. These Mobile Prefixes are announced by a small set of Aggregation Routers which also keep track of all mobile networks current locations. Thus these Aggregation Routers play a role similar to Home Agents in MIP and can be used as the last resort to reach mobile networks.

3.1.5 End-to-end solutions

The protocols described so far share a common assumption that the CN is not required to be aware of the mobile's movement[2]. A group of solutions have also been developed that expose the mobile's movement to the CN, thus providing mobility support without requiring either intermediates (e.g., home agents in Mobile IP) to forward packets or network routing to track the mobile's locations. We call this class of solutions end-to-end solutions.

Typically, they use the existing DNS infrastructure to store the mapping between a mobile's identifier and its current location. A CN can query the DNS to obtain the location of a mobile at the beginning of the data exchanges, and the mobile is responsible for updating both the DNS and the CN afterwards during

[2]Mobile IP's route optimization, for example, allows CN to track the mobile, but it is not mandatory

the communications. If the CN and the mobile move simultaneously (and hence the former misses the latter's update message), it can always resort back to query DNS again to find the mobile's latest location.

3.1.5.1 E2E and M-SCTP

E2E (End-to-End) communication [SB00] gets its name from its end-to-end architecture and is the first proposal that utilizes existing DNS service to track a mobile node's current location. The stable identifier for a mobile is the domain name of the mobile. The mobile uses Dynamic DNS [RTB97] to update its current IP address in DNS servers. To keep the ongoing TCP connection unaffected by mobility, a TCP Migrate option is introduced to allow both ends to replace the IP addresses and ports in TCP 5-tuple on the fly. Thus, the CN can query DNS to obtain the current Locator of the mobile, and after the TCP connection is established, the mobile will be responsible for updating its Locator for this session.

Inspired by E2E, Mobile Stream Control Transmission Protocol (M-SCTP) [XKW02] was proposed in 2002. Similarly, it uses Dynamic DNS to track the mobile nodes and allows both ends to add/delete IP addresses used in Stream Control Transmission Protocol (SCTP) [SXM00] associations during the move.

3.1.5.2 Host Identity protocol

The Host Identity protocol (HIP) [MNJ08] assigns each host an identifier made of cryptographic keys and adds a new Host Identity layer between the transport and network layers. Host Identities, which are essentially public keys, are used to identify the mobile nodes, and IP addresses are used only for routing purposes. In order to reuse the existing code, a Host Identity Tag (HIT), which is a 128-bit hash value of the Host Identity, is used in transport and other upper-layer protocols.

HIP can use DNS as the rendezvous point that holds the mappings between HITs and IP addresses. However, HIP by default uses its own static infrastructure Rendezvous Servers (RVS) in expectation of better rendezvous service. Each mobile node has a designated RVS, which tracks the current location of the mobile node. When a CN wants to communicate with mobile node, it queries DNS with a mobile node's HIT to obtain the IP address of the mobile node's RVS and sends out the first packet. After receiving this first packet, RVS relays it to the mobile node. The mobile node and correspondent node can henceforth start communication on the direct path. If the mobile node moves to a new address, it notifies the CN by sending HIP UPDATE with LOCATOR parameter indicating its new IP address. Meanwhile, it also updates the mapping in RVS.

3.1.5.3 ILNPv6

ILNPv6 stands for *"Identifier-Locator Network Protocol for IPv6"* [ABH07]. Similar to HIP, it also attempts to clearly separate identifier from locator.

The ILNPv6 packet header was deliberately made similar to the IPv6 header. Essentially, it breaks an IPv6 address into two components: high-order 64 bits as a locator and low-order 64 bits as an identifier. The identifier identifies a host, instead of an interface, and is used in upper-layer protocols (e.g., TCP, FTP); on the other hand, the locator changes with the movement of the mobile node, and a set of locators can be associated with a single identifier. Several new DNS resource records (RRs) are required, among which I (identifier record) and L (locator record) are most important. As in current Internet, the CN will query the DNS about the mobile's domain name to determine where to send the packet. During the movement, the mobile node uses secure Dynamic DNS update to ensure that the locator values stored in DNS are up to date. It also sends locator update messages to the CNs that are currently communicating with it. As an optimization, ILNPv6 supports soft-handoff, which allows the use of multiple lo-

cators simultaneously to achieve smooth transition. ILNPv6 also supports mobile networks.

3.1.5.4 LISP-Mobility

LISP-Mobility [FLM11] is a relatively new design. Its designers hope to utilize functions and services provided by Locator/ID Separation Protocol (LISP) [FFM12], which is designed for Internet routing scalability, to support mobility as well.

Conceptually, LISP-Mobility may seem similar to some protocols we have mentioned so far, such as ILNPv6 and Mobile IP. Lightweight Ingress Tunnel Router and Egress Tunnel Router functions are implemented on each mobile node, and all the packets to and from the mobile node are processed by the two router functions (so the mobile node looks like a LISP site). Each mobile node is assigned a static Endpoint ID, as well as a preconfigured Map-Server. When a mobile node roams into a network and obtains a new Routing Locator, it updates its Routing Locator set in the Map-Server, and it also clears the cached Routing Locator in the Ingress Tunnel Routers or Proxy Tunnel Routers of the CNs. Thus, the CN can always learn the up-to-date location of the mobile node by the resolution of the mobile node's Endpoint ID, either issued by itself or issued after receiving the notification from the mobile node about the staled cache. The data would always travel through the shortest path.

3.1.5.5 BTMM

Back to My Mac (BTMM) [CZW11] is an engineering approach to mobility support and has been deployed since 2007 with Mac OS Leopard release. Each user gets a MobileMe account (which includes BTMM service), and Apple, Inc. provides DNS service for all BTMM users. The reachability information of the user's machine is published in DNS.

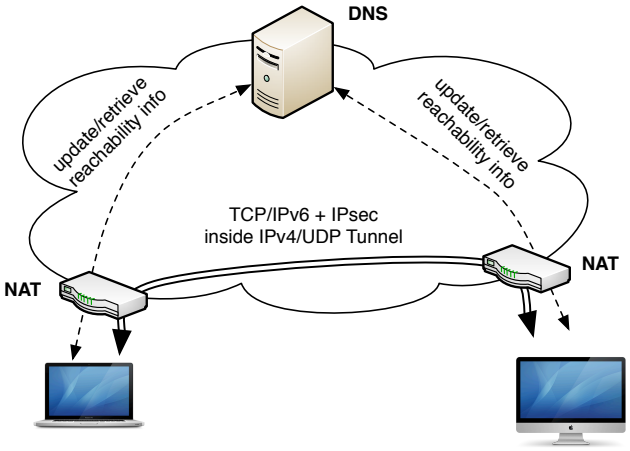


Figure 3.4: An overall picture of BTMM

A mobile uses secure DNS update to dynamically refresh its current location. Each host generates an IPv6 Unique Local Address [HH05] at boot time, which is stored in the DNS database as its topologically independent identifier. The host's current IPv4 address (which is the IPv4 address of the NAT box if the host is behind a NAT) is stored in a SRV resource record [GVE00] together with a transport port number needed for NAT traversal. Every node establishes a long-lived query session with the DNS server so that the DNS server can immediately notify each node when the answer to its query has changed. A host uses its identifier (IPv6 address) in transport protocols and IPsec security associations [KS05] and applications and uses UDP/IPv4 encapsulation to deliver data packets using information learned from the SRV resource record. Note that the locator here is the IPv4 address plus the transport port number and that the IPv6 address is only for identification purposes. In fact, it could be any form of identifier (e.g., domain name); BTMM chose to use IPv6 addresses so that its implementation can reuse existing code. The overall design of BTMM is depicted in Figure 3.4.

What made BTMM unique is that it achieves the functions such as host mobility support, NAT traversal, secure end-to-end data delivery mainly through a well engineered combination of existing protocols and software tools, instead of

developing new protocols.

3.1.6 IP multicast based mobility

Another interesting category of mobility support approaches is IP multicast based solutions.

MSM-IP, which stands for *"Mobility Support using Multicast in IP"* [MB97], aims to leverage IP multicast routing for mobility support. In IP multicast, a host can join a group regardless of to which network it attaches and receive packets sent to the group after its join. Thus mobility is naturally supported in the domains where IP multicast is deployed. Note that MSM-IP does not address the issue of feasibility of supporting mobility through IP multicast, but rather it simply shows the possibility of using IP multicast to provide mobility support, once/if IP multicast is universally deployed.

MSM-IP assigns each mobile node a unique multicast IP address as the identifier. When the mobile node moves into a new network, it initiates a join to its own address, which makes the multicast router in that subnet join the multicast distribution tree. Whoever wants to communicate with the mobile node can just send the data to the mobile's multicast IP address, and the multicast routing will take care of the rest.

Note that, due to the nature of multicast routing, the mobile node can have the new multicast router join the group to cache packets in advance before it detaches the old one, resulting in smoother handoff.

3.2 A Multi-factor Design Space

After a comprehensive overview of the mobility solutions that have been proposed or standardized so far, in this section we contrast and compare different design

choices.

3.2.1 Routing-based versus mapping-based

All existing mobility support designs can be broadly classified into two basic approaches.

The first one is to support mobility through dynamic routing. In such designs, a mobile keeps its IP address regardless of its location changes; thus, the IP address can be used both to identify the mobile and to deliver packets to it. As a result, these designs do not need an explicit mapping function. Rather, the routing system must continuously keep track of a mobile's movements and reflect its current position in the network on the routing table so that at any given moment packets carrying the (stable) receiver's IP address can be delivered to the right place.

It is also worthwhile to identify two sub-classes in routing-based approaches. One is broadcast based, and the other is path based. In the former case, either the mobile's location information is actively broadcasted to the whole network or a proactive broadcast query is needed to obtain the location information of a mobile (e.g., Columbia and Connexion); in the latter case, on the other hand, a host-based path is maintained by the routing system instead (e.g., Cellular IP, HAWAII, and TIMIP).

The second approach to mobility support is to provide a mapping between a mobile's stable Identifier and its dynamically changing IP address. Instead of notifying all the routers within a domain on every movement, a mobile only needs to update a single binding location about its location changes. In this approach, if one level of indirection at IP layer is used, as in the case of Mobile IP, it has a potential side effect of introducing triangle routing; otherwise, if the two end nodes are aware of each other's movement, it means that both ends have to support the

same mobility protocol.

Yet, there is the third case in which the protocols combine the above approaches in the hope of keeping the pros and eliminating some cons of the two. WINMO is a typical protocol in this case.

3.2.2 Operator-controlled versus user-controlled

By and large the global mobility support today is provided by cellular networks. Different from Mobile IP, cellular networks use a service model that bundles together mobility support with device control and network access control. The success of cellular market speaks loudly that the current cellular service model is viable and is likely to continue into foreseeable future. Consequently there have been many efforts in IETF in recent years to develop mobility support standards that follow the cellular networks model, and PMIP represents one of such outcomes.

One main reason for this approach is perhaps backward compatibility. By not requiring the participation of mobiles in the control-signaling process, it avoids any changes to the mobile nodes so that the mobile nodes can stay simple and all the legacy nodes can obtain the same level of mobility services as the latest mobile devices. According to the claim of cellular vendors and operators, transparent mobility support is a key aspect for success as they learn from their deployment experience.

On the other hand, most of the mobility support protocols focus on mobility support only, assuming mobiles already obtained network access. Mobile nodes typically update their locations themselves to the rendezvous points chosen by the users, and, of course, only the nodes implementing one of these solutions can benefit from mobility support. However, this class of protocols does offer users and mobile devices more flexibility and freedom, e.g., they can choose whatever

mobility services are available as long as their software supports that protocol, and they can also tune the parameters to get the services that are most suitable to them.

3.2.3 Local versus global

Discussions on mobility support often involve the concepts of local mobility support and global mobility support. Before we proceed to discuss the roles of local and global mobility support, let us first clarify the definitions of local versus global mobility.

Generally speaking, mobility support should be global in scope, i.e., a mobility protocol should enable communications with a mobile independent from the geographic or topological distances between a correspondent and the mobile. Mobile IP and HIP are examples of such protocols.

A local mobility management protocol, by definition, is to work within a *local* domain. However different communities have different interpretations regarding what defines a local domain.

One definition of local domain is a relatively small network or a geographic area (e.g. a university campus or campus network). Under such assumption, a local mobility support protocol is designed to work together with global mobility management and thus focuses more on performance issues, such as handoff delay, handoff loss, local data path, etc. Since it is typically used on a small scale with a limited number of mobile nodes, sometimes the designers can use some fine-tuned mechanisms that are not scaled with a large network (such as host route) to improvement performance. As a side effect of local mobility management, the number of location updates sent by mobile nodes to their global rendezvous points is substantially reduced. Thus, the existence of local mobility management also contributes to the scalability of global mobility management.

Mobile operators, on the other hand, define local versus global mobility as whether a mobile moves within a provider's network or across the boundary to a different provider's network, respectively. A mobile may roam across continental US and is considered as local mobility if it is still within the same operator's network, or it may be considered to have global mobility even though it may only switched to another providers network without physical movement. This, however, makes the designs no longer complementary to global mobility support. Rather than aiming at improve the user experience and scale the the global mobility support, it mainly serves to control the mobility support of the devices within a provider's network.

3.2.4 Mobility awareness

Among the various design choices, a critical one is how many entities are assumed to be mobility-aware. Stated in another way, the mobility is hidden from which parties. There are four parties that may be involved during a conversation with a mobile: the mobile itself, CN, the network, and home agent or its equivalent (additional component to the existing IP network that holds the mapping).

As routing-based approaches generally expose the movements of the mobiles to all routers and requires no participation from either the mobile or the CN, we mainly focus our discussion on mapping-based approaches here.

The first design choice is to hide the mobility from the CN, based on the assumption that the CN may be the legacy node that does not support mobility. In this approach, the IP address which is used as the mobile's identifier points to the home agent or its equivalent that keeps track of the mobile's current location. If a CN wants to send packets to a mobile node, it sets in the destination field of IP header an IP address which is a mobile's identifier. The packets will be delivered to the location where the mapping information of the mobile is kept, and later

they will be forwarded to the mobile's current location via either encapsulation or destination address translation. Mobile IP and most of its extensions, as well as several other protocols fall into this design.

The second design choice is to hide the mobility from the mobile and CN, which is based on a more conservative assumption that both the mobile and the CN do not support mobility. Protocols like PMIP and TIMIP adopt this design. The protocol operations in this design resemble those in the first category, but significant difference is that, here the mobility related signaling (e.g. update locator to the home agent) is handled by the entities in the network, rather than the mobile itself. Hence the mobile blissfully assumes that it is always in the same subnet.

The third one is to let both mobile and the CN to be mobility-aware. As a result, the network is not aware of the mobility and no additional component is required. As increasing number of mobile devices are connected to Internet (why hide mobility to them), this design choice seems to be more and more appealing. One common approach taken by this design is to use DNS to keep track of mobiles' current locations. Mobiles use dynamic DNS updates to keep their DNS servers updated with their current locations. This approach re-utilizes the DNS infrastructure, which is ubiquitous and quite reliable, and makes the mobility support protocol simple and easy to deploy. Protocols like E2E, ILNPv6 and BTMM fail into this design. Although HIP adds special purpose rendezvous servers to the network to replace the role of DNS, both mobile and CN are still mobility-aware, and hence it is also classified in this category.

3.3 Limitations of IP mobility support

In this section we examine several limitations of IP mobility support.

Although the existing IP mobility solutions differ in many ways, they nonethe-

less share several common limitations as we list below.

First, as IP is designed for point-to-point data delivery, so far the IP mobility solutions also focus only on maintaining point-to-point communication sessions. However, in the face of the increasing popularity of other communication patterns, such as information-centric and many-to-many group communications, the model inherited from IP greatly limits the capability of applications built on top of IP mobility solutions.

Second, security has always been a big challenge. Like the prevailing practice of securing the communication channels in the current Internet, the security in IP mobility solutions is tied to IP addresses. This means that the security associations must be updated or reestablished when one or both ends move, which is inefficient and fragile in mobility scenarios where devices tend to move around frequently. Also, once the other end is authenticated and the communication channel is established, whatever data comes from the channel is regarded as from the node that one intends to communicate with. However, this is vulnerable in practice as protocol designs tends to treat security as the "add-on" features and often use very basic measures that are easily broken [Ram06]. For example, Mobile IPv6 [JPA04] uses a *"return routability procedure"* to protect the location update from the mobile to the correspondent node, which simply checks whether the sender of the location update message is in fact reachable using the claimed new care-of address and the home address.

Last but not least, a basic assumption of IP mobility solutions is that the mobile always has connectivity to the infrastructure. However, in reality the mobility can easily lead to intermittent connectivity or ad hoc connectivity among a set of mobiles. Such problems are currently handled by DTN and MANET respectively, which creates a great hurdle for applications to work under any connectivity conditions.

CHAPTER 4

A New Perspective on Mobility Support

The reason behind the various shortcoming of IP mobility support is that previous attempts typically incorporated TCP/IP's host-centric communication and addressing model, which does not fit the dynamic mobile networking environment and fails to accommodate emerging communication patterns.

Meisel et al. [MPZ10] were the first to argue that MANET can be made more effective and efficient using named-data approach. [MPZ10] observed that MANET solutions typically adopted basic models from wired Internet protocol stack which was designed for wire-connected network topology, and thus are unsuited for the highly dynamic, ad hoc MANET environment. The authors then demonstrated that using named-data approach for data delivery in MANET can better utilize the broadcast nature of wireless channels and can effectively and efficiently handle both physical mobility of nodes and logical mobility of application data.

In this chapter, we take a further step forward and assess the overall picture of mobility support in NDN.

4.1 Mobility Support in NDN

NDN is particularly beneficial in providing mobility support. Mobile nodes can communicate based on what data they need, instead of trying to maintain a specific path to reach a specific node. In this section, we show that NDN, with its

ability to name individual content packet, to securely bind name to the content, to keep per-packet state in forwarders, and to have flexible Interest forwarding strategy at each node, can greatly simplify the mobility support design in various aspects.

4.1.1 Location independent data security

NDN secures the data itself, a design choice that decouples trust in data from the trust in hosts (or locations). This simple yet transformational shift is of tremendous significance in mobility support. The difficult problem of securing the ever-changing communication channels and all the boxes along the way no longer exists. Instead, one only needs to concern about the security of the data itself. The task of providing provenance is achieved by the per-packet data signatures and the data secrecy, if needed, is accomplished by encryption, leaving open only the task of trust management between data producers and consumers, but not any channels or boxes in the middle of the dynamic data delivery paths.

4.1.2 Enhance delivery with network caching

Because each data packet in an NDN network is named and signed, it can be cached in any node and used to satisfy later requests. Generally speaking, in an ad hoc environment, the distinction between routers and end hosts is blurred, as every node may need to help forward packet for other nodes. NDN's built-in caching support can bring great opportunities to enhance data delivery in dynamic ad hoc environment through opportunistic caching by any nodes.

First, when the network paths are dynamic and unpredictable traditional caching approaches, which cache the whole application-level data objects rather than segments of objects, work poorly or not at all in mobile ad hoc environments. This is due to the fact that each cached objects has to be retrieved in its entirety

from the same caching node. However, given that application-level data objects, e.g., images, videos, tend to be large in size, it is often the case that the data paths changes mid of fetching a big object, resulting in wasted effort. In NDN, since each data packet is meaningful independent of where it comes from or where it may be forwarded to, different chunks of the same large application data object can be opportunistically stored at multiple intermediate nodes to speed up future requests. Content or infrastructure providers can also set up designated long-term caching nodes for popular content, and it is entirely likely that some of the mobile user's requests get satisfied by caches on one node and others by a different node as a result of movement.

Furthermore, NDN caching also improves the performance in noisy wireless environment and smoothens the mobile's handoff. For example, as shown in Figure 4.1, the packets that are lost in last wireless hop are cached in the access point and can immediately satisfy the retransmitted Interests from the mobile, without the need to go all the way to the content provider again. Also, smooth handoff is achieved without special optimization techniques, which are required in IP mobility solutions. For instance in IP, a mobile needs to ask the old access router to tunnel packets to the new access router during the handoff (see Figure 4.2a). This usually imposes additional requirements on connectivity (e.g., a mobile may need to connect to both old and new access routers at the same time), requires both routers and the mobile to support special protocols, and potentially results in sub-optimal data paths. On the contrary, as shown in Figure 4.2b, the NDN data packets that are not delivered to the mobile during the handoff are opportunistically cached in the network and can be retrieved when the mobile retransmits the previous Interests from the new access router.

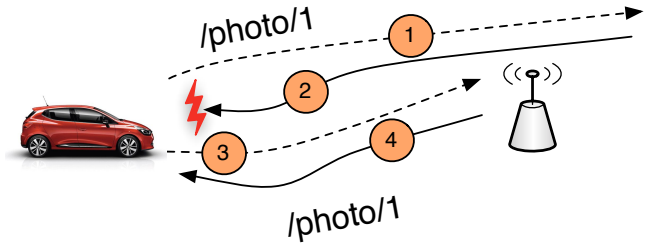
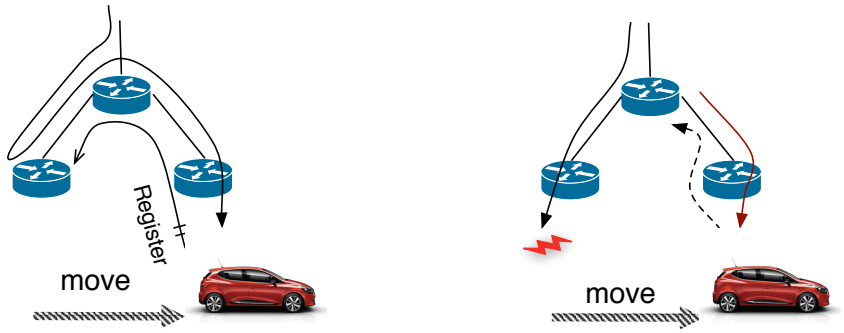


Figure 4.1: Recover last hop packet losses



(a) optimization of handoff in IP by tunneling data

(b) caching helps smoothen handoff in NDN

Figure 4.2: Handoffs in IP and NDN

4.1.3 Integrating DTN and MANET into mobility support

NDN automatically embraces ad hoc and delay-tolerant data delivery, rather than handling them by separate sets of solutions as in today's Internet. It achieves so as follows.

First, NDN names individual data pieces and routes Interests based on application-defined names rather than network driven conventions. This enables mobile nodes to communicate based on what data they need, instead of computing a path to reach a specific node. Also, data can be cached and carried around by mobile nodes. Second, NDN fully utilizes the broadcast nature of wireless communications. The Interests can be broadcasted and whoever has the answer can reply; the overheard data can be cached by a node. Finally, it operates over any transport

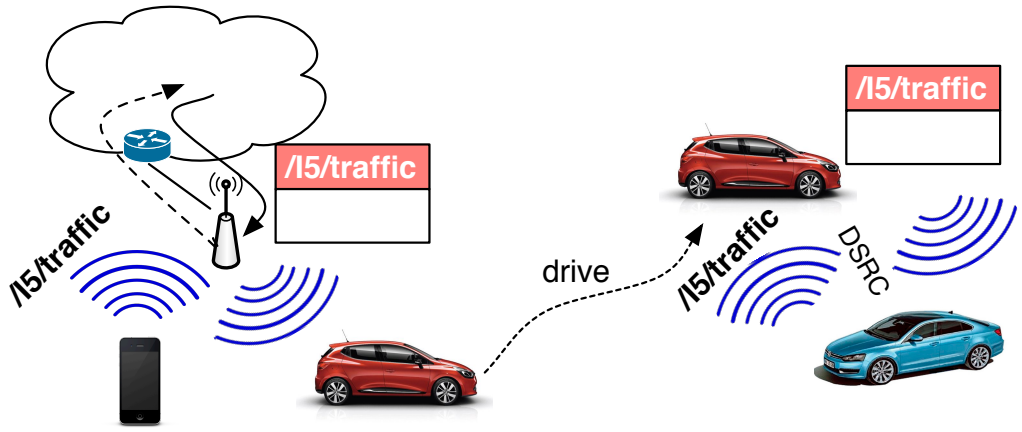


Figure 4.3: An example of unified handling of DTN and connected networks

that can deliver datagrams, and has no dependency on IP, although it can fully utilize IP connectivity when available. This allows mobile nodes to issue requests for data as soon as they have connectivity and without host address assignment.

Thus, DTN and MANET support is embedded in the NDN architecture and the communication model remains the same regardless of the types of networks, which greatly simplifies the mobile application designs. The example shown in Figure 4.3 illustrates how mobile communication under different connectivity conditions is supported in NDN. A mobile phone uses WiFi to request traffic information. The reply to this requests is happened to be overhead by a car, which opportunistically caches it for potential future use. Later, if the car encounters other cars, it can use the cached data to reply their requests for the same traffic information, even in an open field without wireless coverage using Dedicated Short Range Communications (DSRC) [Ken11, WAK12].

4.1.4 Support of mobile data consumers

Supporting mobile consumers in IP mobility solutions is not trivial. As a requirement of IP's host-to-host communication model, the consumers have to acquire IP addresses, set up connections or sessions, and perform updates whenever the

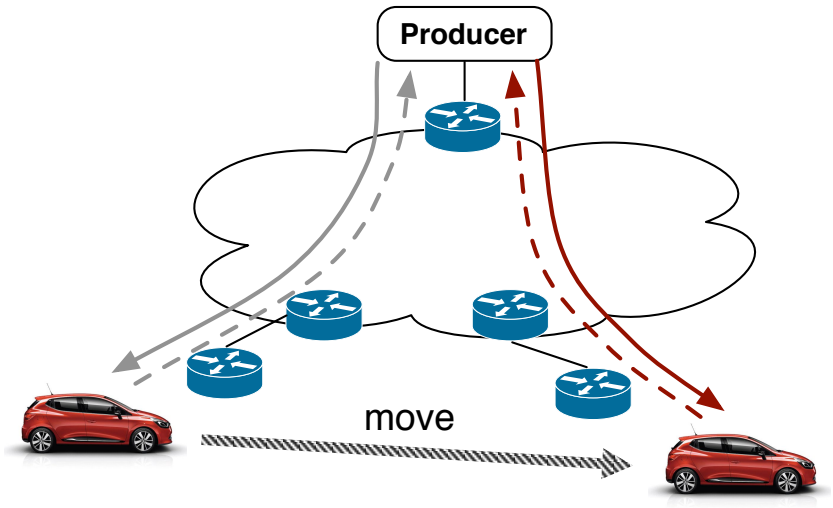


Figure 4.4: Data flows back to a mobile consumer along the reverse paths of Interests

IP addresses change.

NDN's receiver-driven communication model supports the mobile consumers in a straightforward way without creating unnecessary hassles. A consumer can send out request for data directly without address assignment or connection setup. When the Interest travels to the data producer, it sets up a temporary reverse path between the consumer and the producer, along which the corresponding data packet flows back. Such per-packet reverse paths are set up on-demand, short-lived (approximately the same to the round trip time), and are cleared once the the desired data packets come back. Hence, even if the consumers are on move, not a single entity is required to perform any kind of special operations. As depicted in Figure 4.4, the data naturally follows a different reverse path to the consumer after it moves.

4.1.5 Support of mobile data producers

In NDN, the network needs to route Interests based on data names. Unless the mobile producers dynamically announce their namespaces via routing system,

mobile producers must have a way for Interests to reach them.

The invaluable lessons learned from IP mobility support research can be applied to solve the problem. Each mobile producer can be assigned a namespace, such as `/twitter/foo`, under which it publishes its data regardless of its location. Such a namespace can be used as the stable identifier for the mobile. The locator, on the other hand, can be a name that indicates the location of the mobile (e.g. the name prefix announced by the access network). The mapping between the identifier and the locator can be provided either by broadcasting or intermediate nodes such as DNS servers.

NDN is broadcast friendly. Interests are relatively small in size, making it feasible to broadcast Interests in a restricted domain in order to track the location of a mobile producer. Moreover, if multiple consumers are interested in fetching data from a mobile producer, only one Interest would be broadcasted and the data that describes the location of the mobile would be multicasted to all consumers. Figure 4.5 illustrates an example where three consumers utilize broadcast Interests to track a mobile producer. Thus, it is the simplest and most robust way to support mobility in a broadcast domain (a small network or a broadcast overlay spanning over large networks).

The broadcast approach has even more advantages if the communication media is broadcast in nature, such as the wireless LAN, where only one Interest-Data exchange is needed even if there are multiple consumers interested in fetching the mobile producer's data.

When the broadcast is not feasible, DNS can be exploited to store the mapping between the mobile's namespace and its location. An example of DNS-based approach is shown in Figure 4.6. When the user with `/twitter/foo` namespace is visiting Stanford campus, he can discover the name prefix of the Stanford network and update the location field of the DNS record to be `/stanford`. To fetch this user's data, a consumer queries DNS and sends an Interest `/twitter/foo/`

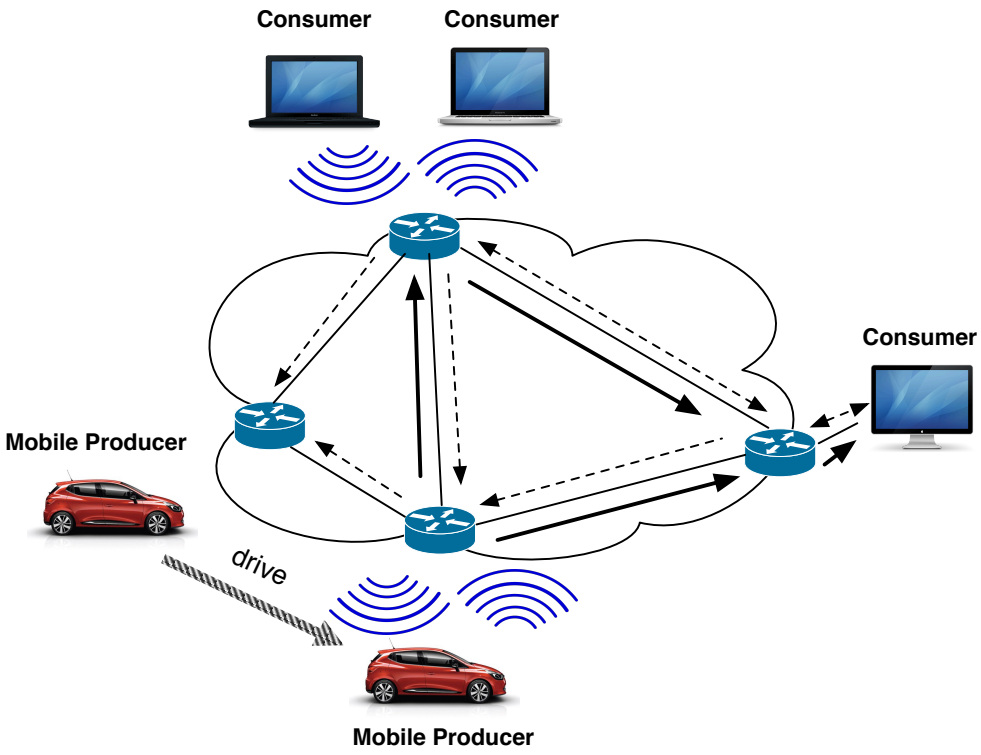


Figure 4.5: Consumers track a mobile producer using broadcast Interests, where dashed arrows are Interests and solid arrows are Data packets

`tweets/95` using the returned location name `/stanford` as *forwarding hint* [AYW12], which is not a part of the data name being requested but indicates a suggested direction for routers to forward the Interests towards the mobile producer.

4.1.6 The power of data fetching strategies

The requirements for data fetching may vary a lot depending on application scenarios. Some may wish to have reliable delivery; some may wish to fetch content in spite of intermittent connection due mobility or power off; some may wish to use whichever interface that has best connectivity or to use all interfaces simultaneously.

NDN's *forwarding strategy*, which determines how alternative paths are uti-

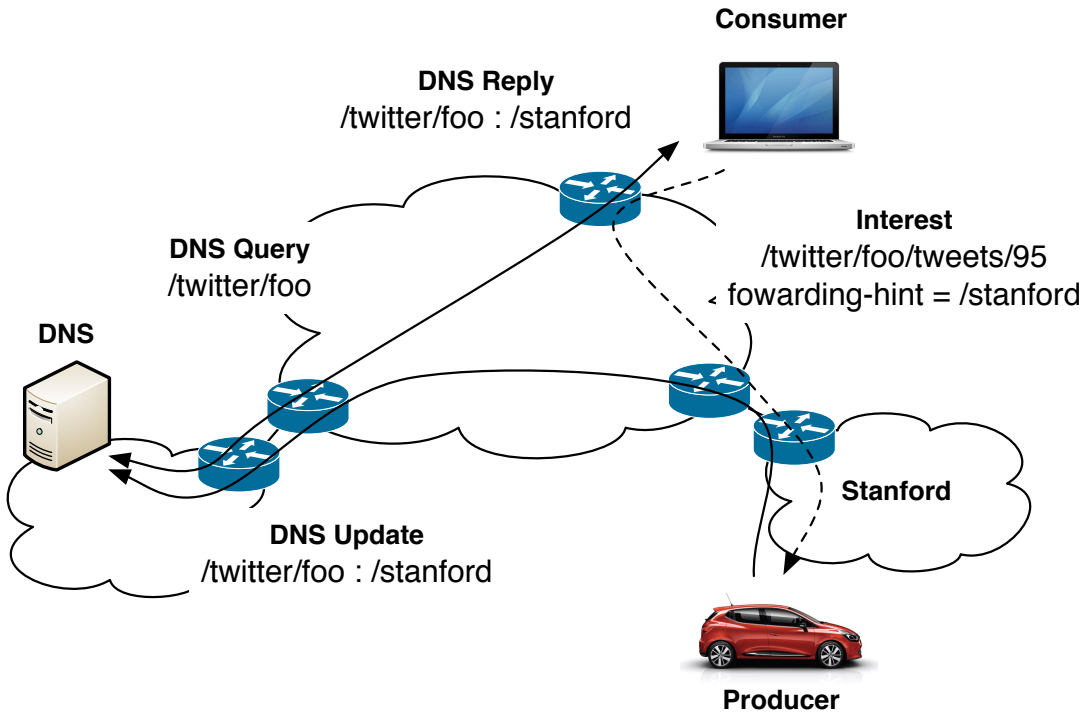


Figure 4.6: A consumer fetches data of a mobile producer using DNS-based approach

lized, suits well for such requirements on flexibility. This is enabled by the per-packet statistics and loop-free Interest forwarding, and it also takes into account different properties of the interfaces (e.g., whether it is a broadcast interface or whether the connectivity is free) and namespaces when determining the strategy of Interest forwarding. Thus, instead of a one-size-fits-all approach, different strategies could be designed to satisfy different requirements.

For example, to exploit multiple interfaces, a strategy may occasionally send the same Interests out via multiple interfaces (since there is no danger of looping) and run experiments to obtain round-trip delays of different interfaces, and then decide which one to use until it is time for the next experiment, as illustrated in Figure 4.7. Or different Interests could be sent out through different interfaces simultaneously, the portions adjusted according to the measured delays and the

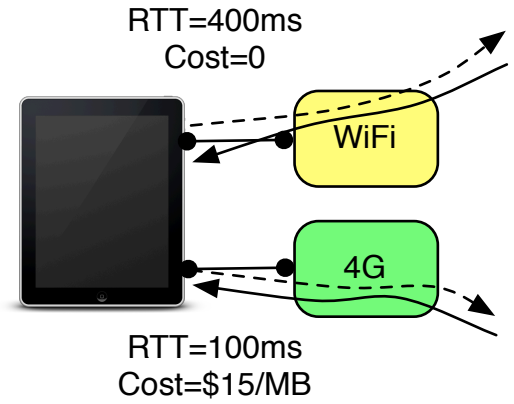


Figure 4.7: Utilizing multiple interfaces

cost, in order to utilize all the interfaces.

Another example is to explore alternative means of fetching data produced by mobiles. Various means can be adopted by the strategy. It could first use local broadcast Interest to see if the data is already cached nearby by some peer consumers. It may send Interests directly towards to mobile producers use the most recent known "locations" reported by the mobiles (which may be obsoleted as the mobiles may not even have connectivity at the time). If the mobile has some designated storage servers to backup its data, the consumer may also send Interests to such servers. In fact, the consumer could also run experiments, as shown in Figure 4.8, by occasionally employing all these measures simultaneously to determine the best way of Interests dispatching in order to increase the chance of successful data fetching with minimal delay.

4.2 Related Work

M. Meisel et al. [MPZ10] criticized the current practice of using the Internet protocol stack in the wireless ad hoc networks and argued that mobile ad hoc networks can be made more effective and efficient through NDN.

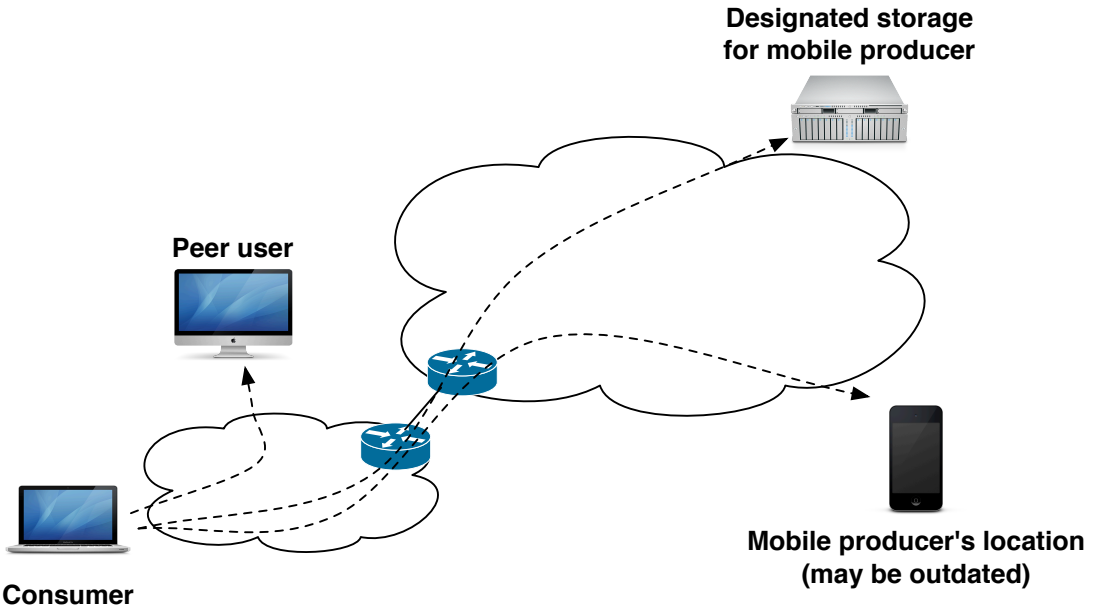


Figure 4.8: A consumer runs experiments to determine the best way to retrieve data generated by a mobile producer

G. Tyson et al. [TSR12] surveyed several new Internet architecture projects and identified benefits and challenges brought from transforming into information-centric communication in terms of mobility support.

Some proposals [HNG12, KKK12, RLZ12] has been published to handle the mobility in NDN. However, they mainly focused on traditional point-to-point communication scenarios and tried to directly map the IP mobility solutions onto NDN. Their perspective on mobility support resembles those of the IP mobility solutions and did not fully take the advantages brought by NDN's transformational architectural shift.

CHAPTER 5

Exploring New Design Patterns for Distributed Applications

In this chapter, we move on to discuss the new design patterns for distributed applications over NDN using an application-driven approach. Unlike in the current Internet, where the application-established names are meaningless to the underlying IP networks, NDN uses application defined names for data delivery. That is, application data names are both used by the network and by the applications. This empowers applications to adopt new designs that they are unable to adopt in the current Internet, which better align with the data-centric nature of application requirements.[1]

This chapter is organized as follows. Section 5.1 describes two expeditionary examples of distributed applications that we developed in order to explore the design space and identify useful building blocks for distributed applications over NDN. In Section 5.2 we present ChronoSync dataset state synchronization protocol. In Section 5.3 we show the distributed and data-centric approach of securing distributed applications. We demonstrate ChronoShare, a distributed file sharing application that is based on ChronoSync and secured in a data-centric way in Section 5.4.

[1]The mobility related problems for distributed applications are taken care of by applying the techniques described in Chapter 4. Thus, while we focus on other aspects of new design patterns of distributed application design in this chapter, the designs we proposed effectively provide mobility support for mobile devices.

5.1 Distributed Applications: Two Expeditionary Examples

Traditional transport services provide point-to-point data delivery and most of today's distributed applications, including peer-to-peer applications, heavily rely on centralized servers. To aid the development of robust and efficient distributed applications, we envision a fundamentally new building block for distributed systems which fully takes the advantages brought by NDN. To better understand the requirements of distributed applications and to explore the design space, we developed two prototypes of distributed applications that cover the popular mainstream usages, multi-party audio conferencing and group text chat.

5.1.1 Audio conference tool

Audio conferencing has had a long history starting from conference applications over APPAnet in the late 70's. In mid 90's MBONE [MB] was established and a set of multimedia communication tools [MJ95, vat] were developed based on lightweight IP Multicast delivery model. Today, as the demands soar, various commercial applications such as Skype, Google Hangout, etc., offer audio conferencing functions. Furthermore, web browsers are starting to support realtime communications capabilities through the projects such as WebRTC [Web], which is an open project supported by Google, Mozilla and Opera to enable realtime communications applications to be developed in the browser via simple Javascript APIs and HTML5.

Hence, we developed an audio conference tool (ACT), aiming to support the popular application on top of NDN in a decentralized fashion. We show through the course of design how we exploited the advantages of NDN to accommodate the communication patterns of conference applications as well as the trade-offs we made to achieve system simplicity, flexibility, and robustness.

5.1.1.1 Overview

As an audio conferencing tool, ACT must perform the following three basic tasks for each user interested in participating in a conference: reporting existing conferences (that a user is eligible to learn), delivering voice data to all participants of a conference in real time, and media data processing together with a user interface.

The first task is to collect the latest information about all existing conferences, both scheduled and ongoing ones, and the speakers of each ongoing conference that the user wants to join. This requires that each Interest packet for conference information must be propagated to everywhere across the network. ACT chooses the names for conference information data from a broadcast name space. Once a user learns about the interested conference and the speakers in that conference call, she can receive the voice data by sending Interests directly towards each speaker. Hence, ACT uses two difference name spaces, broadcast name space for conference information discovery and routable name space for voice data delivery. Interests for the former operates at the frequency of conference launching or speaker joining, while the Interests for voice data streams need to be generated at the frequency of audio packet generation which can be orders of magnitude higher.

The third task, media data processing and user interface design, is a necessary component of ACT but largely decoupled from network specifics. To focus our effort on NDN specific design issues, ACT simply adopted a client-server based open source audio application package, and embeds the modules for speakers discovery and voice data distribution in the original server code, leaving the client intact. A converted server runs on the same node with the client, and communicates with the client using the standard IP protocol stack, while at the same time communicates with other converted servers over NDN.

Figure 5.1 shows a simple block diagram of our design. A separate module handles the conference discovery and facilitates the task of joining a conference.

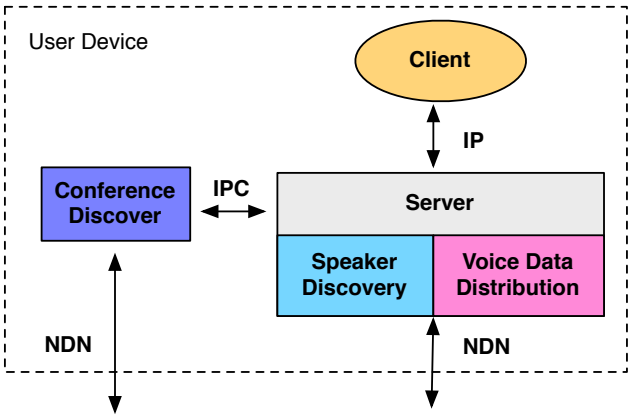


Figure 5.1: An overview of ACT design

The decoupling of conference discovery from voice media delivery gives us the flexibility to extend ACT with other features such as video, distributed whiteboard, and text messaging. In the rest of this section we describe each of the above three pieces in more detail.

5.1.1.2 Conferences discovery

ACT allows each user to retrieve a list of scheduled or ongoing conferences, and to announce a new conference to potential participants. To retrieve conference information, a user must know the name for the conference information data a *priori*, before she has any knowledge about the conferences to be discovered. Hence, conference initiators and participants must agree on the name prefix for conference description data by following some established application naming conventions. An example of the name for conference data within the NDN scope is `/ndn/broadcast/conference/conference-list`. The first name component defines the scope of the network (within NDN testbed); the second one indicates a broadcast namespace (within the NDN testbed scope); the next name component identifies the application type, and the last component specifies the data that the participant is interested in.

Following the established naming convention, a conference initiator announces the conference by creating the conference description data with a proper name under the conference-list prefix. Since there may be multiple different conference description data packets, and any of them can satisfy the Interest with name `/ndn/broadcast/conference/conference-list`, we need to ensure that a participant can learn all conferences under that prefix. Furthermore, ACT must also promptly remove finished or canceled conferences from the conference list presented to the users.

Conference announcement

A user announces a conference by creating a data object describing the conference in the *"Session Description Protocol (SDP)"* [HJP06] format according to the user's input, such as estimated starting time, media type supported, etc. The name of the data is constructed by appending the specific conference name to the conference-list prefix. This data is then stored in the application buffer until the conference ends. As an example, assuming that node A in Figure 5.2 announces a conference named "lunch-talk". Node A listens to Interests with the conference-list prefix, and whenever it receives such an Interest, it responds with the Data packet containing the conference description. As the Data packet travels to the requester(s), it may be stored at the intermediate routers to satisfy Interests from other potential participants in the future. A *"freshness"* time is specified in the Data packet which determines the maximum time it can be considered valid. In general, the conference organizer would only receive one conference-list Interest within the *"freshness"* seconds, as the Interests for the same name will be consumed by the routers that have the cached conference description data.

A conference organizer may also delegate the task of announcing the conference to a third-party node or other participants in the conference. The delegatee in turn can store the conference description data and respond to conference-list Interests on behalf of the organizer in cases when the organizer gets disconnected, or simply

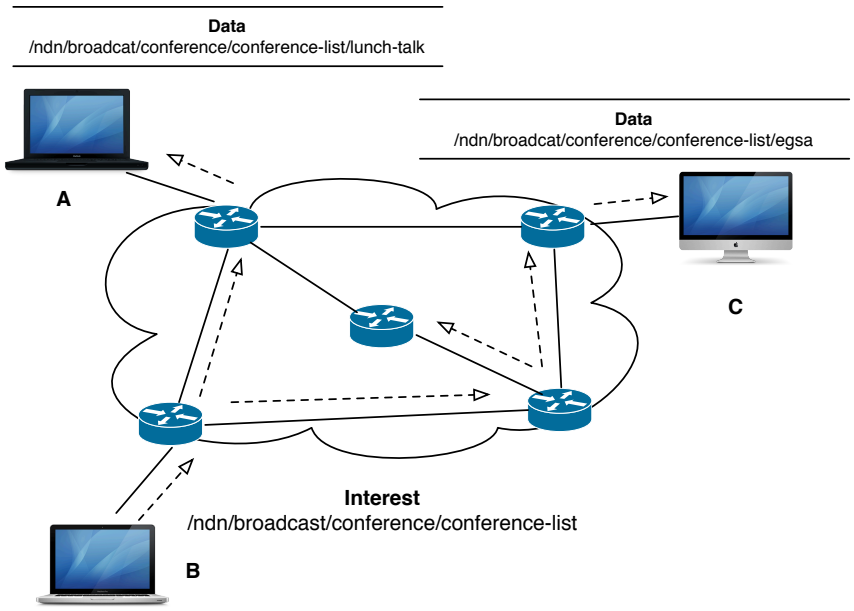


Figure 5.2: An example of conference discovery

goes offline when the conference is still ongoing.

Conference enumeration

When a user turns on ACT, the conference discovery module sends an Interest with the name of conference-list prefix, as node B in Figure 5.2 does. This Interest can be satisfied by any conference description Data packet. As multiple conference announcements may exist, the module needs to send multiple Interests to discover all existing conferences. However simply sending the same Interest again may not bring back new conference description data, as the Interest is likely be satisfied by the previous Data packet that the module just fetched and is now in the router's cache.

One way to addresses this issue is to include an *"exclude filter"* along with the Interest in the *"Selector"* field (see Figure 2.3) as a means for the requester to express explicitly the names of the data it has already received. Depending on the number of names to be excluded, the conference discovery module may simply

Figure 5.3: An Interest with exclude filter

enumerate the known conference names in the exclude filter of one Interest or split them among multiple Interests. For example, after node B in Figure 5.2 learns the conference "lunch-talk", it will send another Interest with the same name of conference-list but specify in the exclude filter "lunch-talk", to retrieve description data about "egsa" announced by node C or any other unknown conference that may exist. Figure 5.3 shows such an Interest that excludes both "lunch-talk" and "egsa".

To get informed immediately whenever a new conference is announced, ACT keeps an outstanding conference-list Interest all the time. That is, it issues another Interest excluding all known conferences as soon as the previous Interest brings back a Data packet or otherwise expires. Although a large number of ACT conference discovery modules may be running simultaneously and all sending out Interests for conference-list all the time, Interests carrying the same name will be aggregated by intermediate routers, and thus there is only one PIT entry for the name conference-list at any router.

5.1.1.3 Speakers discovery

The participants of a conference fall into two categories, the speakers who produce the voice data, and the listeners who request the voice data. ACT only needs to know the speakers in a conference in order to retrieve data from them. For small-scale conferences where every participant is likely to speak, each user can be simply regarded as both a speaker and a listener. For large conferences, generally speaking, there are fewer speakers than listeners. ACT scales well by tracking only a list of active speakers, instead of all the participants.

Speaker discovery is done when a user joins a conference. The mechanism of speaker discovery is similar to conferences discovery. An example name prefix for speaker description (speaker-list prefix) is `/ndn/broadcast/conference/lunch-talk/speaker-list`. When in "speak" mode, an ACT converted server generates and keeps in buffer the data in SDP format describing the name prefix it intends to use for voice data, media type, public key locator, etc. The name of the SDP data is constructed by appending the user name of the speaker to the speaker-list prefix. When changed to "listen" mode, ACT removes the SDP data from the buffer and stops responding to the speaker-list Interest. How a speaker responds to the Interest and how ACT discovers and keeps a list of active speakers are exactly the same as in conferences discovery, except that there is no delegation of speaker announcement.

5.1.1.4 Voice data distribution

A routable name prefix (e.g., /ndn/ucla.edu/cs/zhenkai) is used by each speaker for its voice data. Because a speaker may generate multiple data streams, further name components such as device ID and audio codec can be appended to the name prefix to identify each stream. A device ID represents the physical device that produced the data stream, and it should be unique within the local network so that the access router knows where to forward Interests for a particular data stream. A speaker should include all name prefixes for all the data streams it produces in the speaker description data.

Segments from each voice data stream produced by the speaker are sequentially named and stored in a circular buffer. A typical name for a data segment may look like: `/ndn/ucla.edu/cs/zhenkai/universe/celt/928`.

An ACT converted server sends Interests for all the voice data streams it learned from speaker discovery. When the segment number of a data stream is

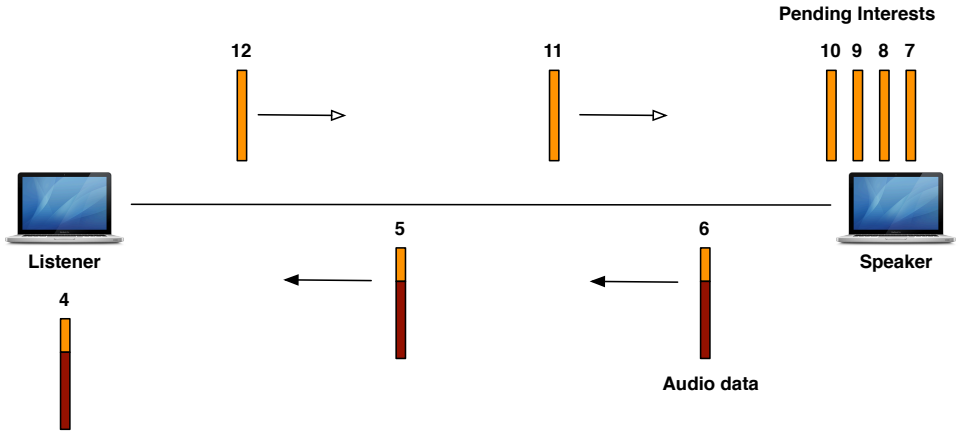


Figure 5.4: Listeners in ACT maintains a pipeline of pending Interests for each audio stream to minimize the delay.

unknown, an Interest is sent asking for the latest data under the stream name prefix (thus no cached content can satisfy this Interest). This ensures that a listener fetches the latest voice content produced by the speaker, instead of lagging behind. When a data segment from this stream is brought back, the server learns the segment number being used at the speaker and starts to explicitly set segment numbers in the future Interests.

In a network with high latency the Interest-Data round trip can delay the reception times of the voice data packets, rendering them unplayable or with low quality. To address this issue ACT takes the same pipeline approach, where the listener always maintains a certain number of outstanding Interests for each voice data stream. That is, whenever an Interest brings back a Data packet or is expired, the listener issues a new Interest, thereby restoring the number of outstanding Interests in the pipeline. Consequently, the speaker's voice data would be fetched immediately when it is generated, as depicted in Figure 5.4.

5.1.1.5 ACT Implementation

We have implemented a proof-of-concept ACT converted server as an extension to the open source project Mumble [Mum]. We made *"murmurd"*, the server part of Mumble to exchange data with other murmurds over NDN. The basic function of audio conference is working (e.g. managing speakers, retrieving voice data, etc), and trials of ACT had been carried out on NDN testbed with up to dozens of participants from all over the continental US, and the subjective voice quality was quite good.

Stepping up a level, we believe that ACT shows a promising example of how to move existing applications into NDN. ACT keeps the Mumble's client-server model intact, and modified the server to perform NDN data transport. Not only this approach leaves the entire client part untouched, but also the converted ACT server can be a stand alone module sitting between an NDN network and the existing Internet to enable an ACT conference call across the boundary.

5.1.2 Group text chat

Another widely used distributed application is group text chat. Various group text chat applications and protocols, be it proprietary or open, have been designed and implemented, among which XMPP [Sai11] is perhaps most popular due to the open standard and open systems approach, by which anyone may implement an XMPP service and interoperate with others' implementations. Although XMPP uses a client-server architecture (clients do not talk to each other directly), it is decentralized in that there is no central authoritative server, and anyone may run their own XMPP server by their own. To further deepen our understanding about the requirements and design space of distributed applications in NDN, we developed a fully distributed group text chat application based on the XMPP protocol.

Similar to ACT's approach of adapting the existing code, we embeds in an open-source XMPP server implementation, *"jabberd"* [jab], the modules for users discovery and message distribution over NDN[2]. Meanwhile the adapted server communicates over IP to any XMPP-compatible chat client, for instance Adium, Empathy, iChat, Pidgin, etc. [Adi, Emp, iCh, Pid], that is running on the same node.

This use case, though seems to be only a trivial variance of audio conferencing, proved to impose quite different requirements, in that a large number of users may be actively involved in the chat, their messages intermingled, whereas in audio conferencing it is usually that one speaker speaks for a relatively long duration during which others passively listen to the speech. Thus, instead of only tracking audio streams from a small number (usually one) of speakers, the group text chat application has to fetch messages dispersed among many users. A straightforward approach to handle this issue [ccn] is to use a global sequence number, and whoever wants to send a message to the rest of the group simply increments the sequence number by one and publishes the message under the name constructed by appending the sequence number to a name prefix in broadcast namespace (for example, `/ndn/broadcast/textchat/group1/35`). However, this simple approach has two major drawbacks: 1) the message data is under a broadcast name, which is undesirable as any request for such data will have to be broadcasted, even when it is unnecessary; 2) it does not handle simultaneous data generation, where more than one users try to send messages.

We thus resort to an alternative approach, to keep a roster of users in a group and to send an Interest with routable name to each user to request their messages, if there is any. While this approach handles correctly the naming of the data and the simultaneous data generations, it creates a long list of names in the exclude

[2]As the groups, or chat rooms, can be discovered using the conference discovery module ACT, we focus on the text chat application itself.

filter and a mesh of pending Interests for messages, which, although mitigated by the Interest aggregation, is still costly and inefficient.

5.2 ChronoSync: Efficient Dataset State Synchronization

In the development of the two expeditionary applications, we noticed that dataset synchronization is an inherent requirement by distributed applications. For example, the group text chat problem can be trivially solved if there is a convenient way to synchronize the chat data produced by the all the users in a chatroom; in ACT, there is also a need to synchronize the conference lists and speaker lists; in joint editing applications users have to synchronize the revisions of the files.

Many of such distributed applications demand efficient and robust synchronization of datasets among multiple parties. The research community has been working on distributed system synchronization since the early days of the Internet [Lam78] and has produced a rich literature of solutions. Quite a few popular applications like Dropbox and Google Docs, on the other hand, are implemented based on a centralized paradigm, which generally simplifies the application designs and brings many other advantages, but also results in single points of failure and centralized control of the data. At the same time, a number of different peer-to-peer solutions [AS04], including the recently announced BitTorrent Sync service [bita], represent another direction in the searching of efficient dataset synchronization solutions, which requires the maintenance of a sophisticated peer-to-peer network overlay structure, whose mis-matching with underlying topology undermines the benefits of application-level multicast, and often the critical nodes for participants rendezvous.

In this section, we describe ChronoSync, an efficient and robust protocol to synchronize dataset state among multiple parties in NDN. We first present an overview of ChronoSync components, and then explain the naming rules in Sec-

tion 5.2.2. Section 5.2.3 shows how ChronoSync maintains the knowledge about the dataset and Section 5.2.4 describes how the changes to the dataset propagate to all participants. Section 5.2.5 and 5.2.6 discuss how ChronoSync handles simultaneous data generations and network partitions. Section 5.2.7 presents evaluation results.

To better illustrate the basic components of the ChronoSync design, we use a group text chat application, ChronoChat, as an example throughout the paper. While a real chat application includes a number of essential components, such as roster maintenance, in our example we introduce only elements that are directly relevant to ChronoSync.

5.2.1 Overview

In the core of any ChronoSync-based application there are two interdependent components, as shown in Fig. 5.5: the ChronoSync module that synchronizes the state of the dataset and the application logic module that responds to the change of the dataset state. In ChronoChat, ChronoSync module maintains the current user's knowledge about all the messages in the chatroom in the form of a *digest tree*, as well as history of the dataset state changes in the form of a *digest log*. After ChronoSync module discovers that there are new messages in the chatroom, it notifies ChronoChat logic module to fetch and store the messages.

To discover dataset changes, the ChronoSync module of each ChronoChat instance sends out a *sync Interest*, whose name contains the state digest that is maintained at the root of the digest tree. Generally, with the help of digest tree and digest log, ChronoSync can infer dataset changes directly and reply to the sync Interest with the data containing the changes, which we henceforth refer to as *sync data*. In cases of network partitioning, ChronoSync also uses *recovery Interests* and *recovery data* to discover the differences in the dataset state.

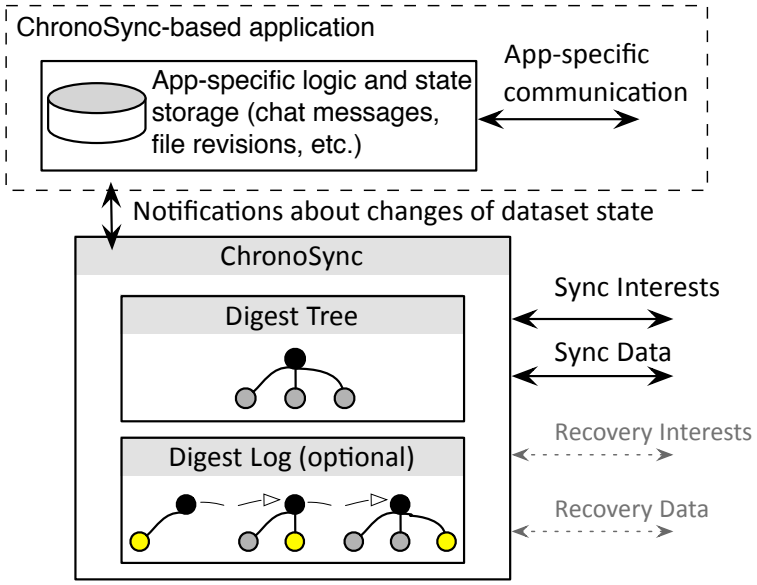


Figure 5.5: ChronoSync overview

ChronoSync focuses solely on facilitating the synchronization of the knowledge about new data items in the dataset, leaving the decision on what to do after ChronoSync discovers state changes at the application's discretion. For example, the sync data in ChronoChat brings back the names of messages newly added to the chatroom, and thus a user's knowledge of the dataset is brought up to date. However, the user may decide whether to fetch all the missing messages or just the most recent ones, if the total number of missing messages is large (e.g., after recovery from a network partition).

5.2.2 Naming rules

One of the most important aspects of application design in NDN is naming, as the names carry out several critical functions. The name carried in an Interest packet is used by the network to figure out where to forward it and to determine which process to pass it to when it reaches the producer. Also proper naming rules can greatly simplify the design of applications.

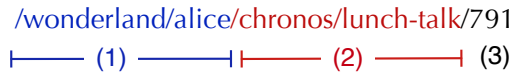
There are two sets of naming rules in ChronoSync: one for application data names and one for sync data names.[3]

We design the application data names to have routable name prefixes so that the Interests can be forwarded towards the producers directly. These prefixes can be constructed by appending one or more components under a prefix assigned by the Internet provider. For example, part (1) of the chat data name in Fig. 5.6a is such a prefix. The purpose of the part (2), which includes the application name and the chatroom name, is to demultiplex the Interest once it reaches the data source: it identifies the process that is responsible for handling such Interests.

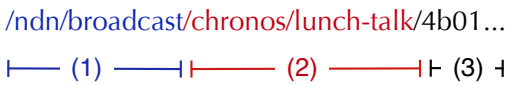
The data generated by a user is named sequentially. For example, in ChronoChat, the initial message from a user to the chatroom has sequence number zero and whenever a new message is generated, be it a chat message or user presence message, the sequence number is incremented by one. As a result, the complete knowledge of the user can be compactly represented by just one name. Assume the name shown in Fig. 5.6a is the latest chat data name used by *Alice*. We can infer from the naming rules that *Alice* has produced 792 pieces of chat data to this chatroom, with sequence numbers ranging from 0 to 791.

Similarly, the name for sync data (Fig. 5.6b) also consists of three parts. Part (1) is the prefix in the broadcast namespace for a given broadcast domain. A broadcast prefix ensures that the sync Interests are properly forwarded to all participants of a group, as it is often impossible to predict who will cause the next change to the dataset state. Part (2) serves the purpose of demultiplexing (similar to that of the application data name), and the last part carries the latest state digest of the Interest sender.

[3]The naming for recovery data will be discussed in Section 5.2.6.



(a) An example of chat data name



(b) An example of sync data name

Figure 5.6: Naming rules of ChronoSync

5.2.3 Maintaining dataset state

The application dataset can be represented as the union of the subsets of data generated by all producers.

Since the knowledge of a data producer can be solely represented by its name prefix and the latest sequence number, ChronoSync tracks the latest application data name of each producer in order to maintain up-to-date knowledge of the dataset. For the sake of simplicity in writing, we refer to the latest application data name of a producer as its *producer status*.

Inspired by the idea of Merkle trees [Mer], ChronoSync uses a digest tree to quickly and deterministically compress knowledge about the dataset into a crypto digest, as illustrated in Fig. 5.7. [4] Each child node of the tree root holds a cryptographic digest calculated by applying, for example, SHA-256 hash function over a user's producer status. Recursively applying the same hash function to all child nodes of the root results in the digest that represents state of the whole dataset, which we refer to as the *state digest*. To ensure that every participant calculates the same state digest when observing the same set of producer statuses, the child nodes are kept in the lexicographic order according to their application

[4] While we use an one-level hash tree here, a more canonical form of hash trees can be used if the applications demand different naming rules.

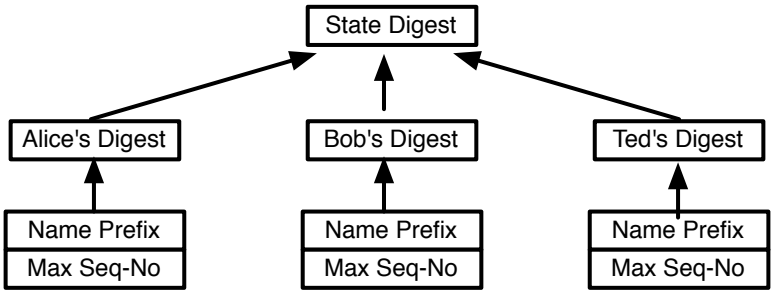


Figure 5.7: An example of digest tree used in ChronoChat

data name prefixes.

The digest tree is always kept up-to-date to accurately reflect the current state of the dataset. Whenever a ChronoChat user sends a new chat message or learns about the name of a new message from another participant, the corresponding branch of the digest tree is updated and the state digest is re-calculated.

As an optimization, each party keeps a *"digest log"* along with the digest tree. This log is a list of key-value pairs arranged in chronological order, where the key is the state digest and the value field contains the producer statuses that caused the state change. An example of digest log is illustrated in Table 5.1. The log is useful in recognizing outdated state digests. For example, when a user resumes from a temporary disconnection and sends out a sync Interest with an outdated state digest, other parties, if recognizing the old digest, can quickly infer the differences between the dataset states and promptly reply the sender with missing data names.

Although the digest log facilitates the process of state difference discovery in many cases, it is not essential ensure the correctness of the ChronoSync design. Depending on the available resources, applications can set an upper bound on the size of the digest log, purging old items when necessary.

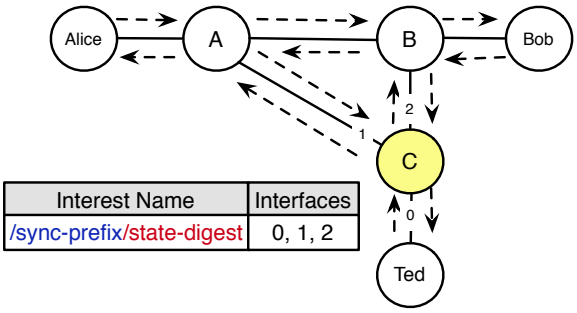
State Digest	Changes
0000...	Null
9w35...	[Alice's prefix, 1]
...	...
23ab...	[Bob's prefix, 31], [Alice's prefix, 19]
05t1...	[Bob's prefix, 32]

Table 5.1: An example of digest log

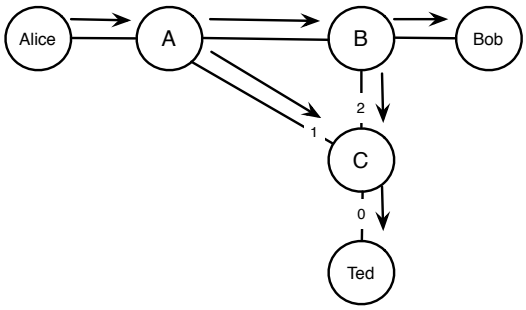
5.2.4 Propagating dataset changes

To detect dataset changes as soon as possible, every party keeps an outstanding sync Interest with the current state digest. When all parties have the same knowledge about the dataset, the system is in a *stable state*, and sync Interest from each party carries an identical state digest, resulting in efficient Interest collapsing in NDN routers [ZEB10]. Fig. 5.8a shows an example of a system in stable state, where there is no ongoing conversation in a chatroom.

As soon as some party generates new data, the state digest changes, and the outstanding Interest gets satisfied. For example in Fig. 5.8b, when *Alice* sends a text to the chatroom, ChronoSync module on her machine immediately notices that its state digest is newer and hence proceeds to satisfy the sync Interest with sync data that contains the name of text message. Because of the communication properties of NDN, the sync data is efficiently multicasted back to each party in the chatroom. Whoever receives the sync data updates the digest tree to reflect the new change to the dataset state, and sends out a new sync Interest with the updated state digest, reverting the system to a stable state. Meanwhile, the users may send Interests to request for *Alice*'s text message using the data name directly. In other more complex applications, the sync data may prompt the applications to perform more sophisticated actions, such as fetching a new version of a file and



(a) For each chatroom, at most one sync Interest is transmitted over a link in one direction. There is one pending sync Interest at each party in the stable state. Due to space limit, only the PIT of router C is depicted.



(b) The state change caused by *Alice*'s new message is multicasted to other two users following the PIT entries set up in routers by sync Interests

Figure 5.8: State change propagation in ChronoSync

applying changes to the local file system.

Normally, the state digest carried in the sync Interest is recognized by the Interest recipients: it is ether the same as the recipient's current state digest or the previous one if the recipient just generated new data. However, even in a loss-free environments, out-of-order packet delivery can result in receiving sync Interests with digests that cannot be recognized. For instance, in Fig. 5.8b, *Ted*'s sync Interest with the new state digest (after incorporating *Alice*'s sync data into digest tree, not shown on the figure) may reach *Bob* before he receives *Alice*'s sync

data, due to the possible out-of-order delivery in the transmission.

To cope with this problem, ChronoSync employs a randomized *"wait timer"* T_w, with value being set approximately on the order of the propagation delay. More specifically, a recipient sets up the wait timer T_w when an unknown digest is received and postpones the processing of the corresponding sync Interest until the timer expires. In the example mentioned above, *Bob*'s state digest would become the same as the new digest after *Alice*'s reply reaches him, before T_w expires.

5.2.5 Handling simultaneous data generations

In simultaneous data generation cases, more than one data producer reply to the outstanding sync Interests. As one Interest can only bring back one piece of data in NDN, simultaneous data generations would partition the system into two or more groups, with each group maintaining a different state digest, depending on whose sync data they have received. At the same time, users in different group will not be able to recognize each other's state digest. This is illustrated in Fig. 5.9, where *Alice* and *Bob* reply to the sync Interests at the same time and only *Bob*'s sync data reaches *Ted*. Thus, the new state digest of *Alice* is different from that of the other two.

This problem can be solved with the *exclude filter* [exc], which is one of the selectors that can be sent along with the Interest to exclude data that the requester no longer needs. When the wait time T_w times out, *Ted* proceeds to send a sync Interest with the previous state digest again, but this time with an exclude filter that contains the hash of *Bob*'s sync data. Routers understand that Bob's sync data, although has the same name as the one carried in the sync Interest, cannot be used as the reply to the Interest. As a result, this sync Interest brings back *Alice*'s sync data from router C's cache. Similarly, *Alice* and *Bob* also retrieve each other's sync data with the help of the exclude filter. At this point, all three

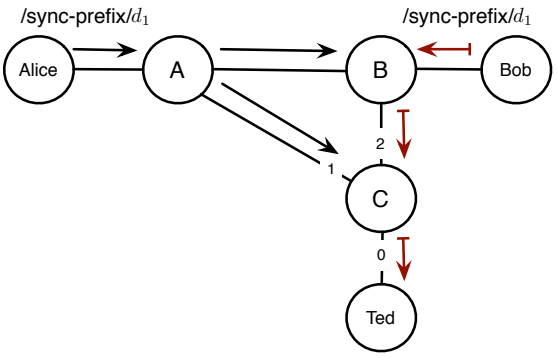


Figure 5.9: An example of simultaneous data generation

users have obtained the knowledge about the simultaneously generated data and compute an identical state digest.

If there are more producers involved in a simultaneous data generation event, multiple rounds of sync Interests with exclude filter have to be sent. Each of such Interest has to exclude all the sync data of a particular state digest known to the requester so far.

5.2.6 Handling network partitions

When network partitions happens, the users become physically divided (as opposed to the logical division in the simultaneous data generation case) into multiple groups. Although within each group users may continue to communicate due to ChronoSync's decentralized design, there is a challenging synchronization problem when the network partition heals: parties in different groups accumulated different subsets of data and it is impossible for them to recognize each other's state digests. Different from what happens in simultaneous data generations, where multiple users reply to the same sync Interest with different sync data, during network partitioning an unknown number of sync data with different state digests may have been generated by multiple parties, rendering the exclude filter ineffective in determining the differences of dataset. Hence, when the Inter-

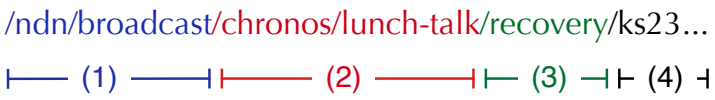


Figure 5.10: An example of recovery Interest

est with exclude filter times out (such Interests should have very short lifetime, as the sync data, if there is any, should already be cached in the routers), ChronoSync infers that the network partitions have happened and measures have to be taken to resolve the differences. Depending on specific application requirements, various set reconciliation algorithms [EGU11, Y 03, J 02] can be used to solve this problem.

For applications such as ChronoChat, for example, ChronoSync resorts to a simple but effective recovery procedure, outlined as follows. The recipient of the unknown digest sends out a recovery Interest, as shown in Fig. 5.10. It is similar to a normal sync Interest, but has a "recovery" component before the digest and includes the unknown state digest, instead of the one in the root of the local digest tree. The purpose of such an Interest is to request missing information about the dataset from those who produced or recognized the unknown state digest. Those who recognize the digest (e.g., having it in their digest log) reply the recovery Interest with the most recent producer status of all users, and others simply ignore the recovery Interest. Upon receiving the recovery reply, the recipient compares the producer statuses included in the reply with those stored in the local digest tree and updates the tree whenever the one in the reply is more recent. This recovery procedure guarantees that the system will revert to the steady state within few recovery rounds (e.g. one round for two groups that have different state digests).

5.2.7 Evaluation

To understand characteristics and tradeoffs of the ChronoSync protocol, we conducted a number of simulation-based experiments of the group text chat service (ChronoChat) using NS-3 [ns3] with ndnSIM module [AMZ12], which fully implements the NDN communication model. In particular, we are interested in confirming that ChronoSync propagates state information quickly and efficiently, even in face of network failures and packet losses. To get a baseline for the comparison, we also implemented a simple TCP/IP-based approximation of the centralized Internet Relay Chat (IRC) service, where the server reflects messages from a user to all others. For simplicity of the simulation, we did not implement heartbeat messages for either ChronoChat or IRC service simulations. Also, chat messages in the simulated ChronoChat application are piggybacked alongside with the sync data. That is, when, for example, Alice sends a new message, her ChronoChat app not only notifies others about the existence of a new message, but also includes the actual message data in the same packet.

In our evaluations we used the Sprint point-of-presence topology [SMW04], containing 52 nodes and 84 links (Fig. 5.11). Each link was assigned measurement-inferred delay, 100 Mbps bandwidth, and drop-tail queue with the capacity of 2000 packets. As the size of the text message is usually small, there is no congestion in the network. All nodes in the topology act as the participants of a single chatroom. The traffic pattern in the room was determined based on the multi-party chat traffic analysis by Dewes et al. [DWF] as a stream of messages of sizes from 20 to 200 bytes with inter-message gap following the exponential distribution with the mean of 5 seconds.

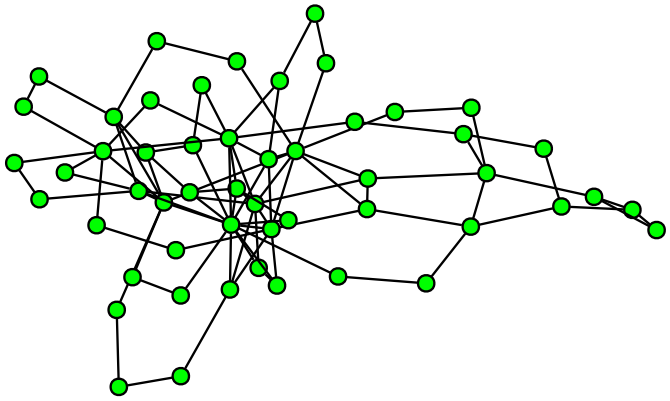


Figure 5.11: Sprint point-of-presence topology

5.2.7.1 State synchronization delay

ChronoSync-based applications are fast in synchronizing dataset state. To evaluate this property quantitatively, we define *state synchronization delay* to be the the time interval between the message generation and discovery of this message by all of the chatroom participants. We performed 20 runs of the chatroom simulation with 52 participants that produced together a total of 1000 messages under various network condition. Each individual simulation run featured different sets of messages injected to the chatroom, with different inter-message delays, different message sizes, and different order of participants talking. In the IRC case, we randomly chose one of the nodes in the topology as the position of the central server for each run.

Performance under normal network conditions

As an initial step, we evaluated ChronoChat under normal network conditions without network failures or packet losses, which allowed us to understand the baseline performance of ChronoSync protocol.

Since in ChronoChat sync data always follows optimal paths built by outstanding sync Interests, the synchronization delay is significantly lower, compared to that of the client-server based IRC implementation, as shown in Fig. 5.12: for

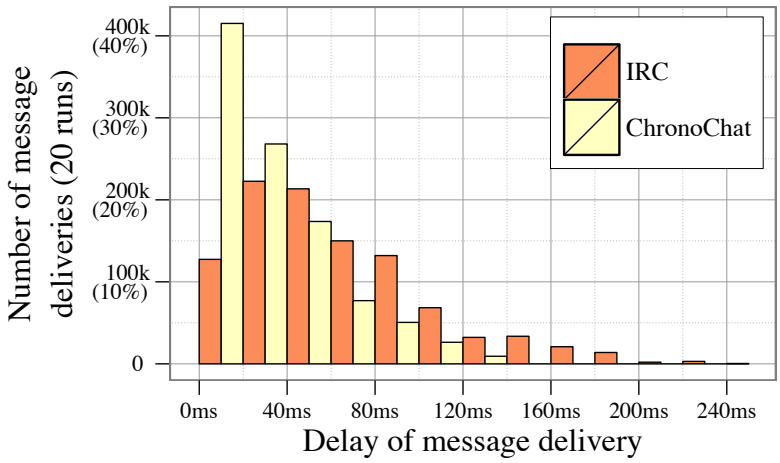


Figure 5.12: Distribution of message delays

ChronoChat, more than 40% of all messages sent in 20 runs experienced delay less than 20 ms, compared to ≈13% of messages in IRC case for the same delay range.

Performance in lossy environments

We evaluated ChronoChat in lossy network environment, with varying level of per-link random packet losses, ranging from 1% to 10%. Fig. 5.13 summarizes the simulation results in form of cumulative distribution function graphs for ChronoChat and IRC services (for better visual presentation, x-axis is presented in the exponential scale and y-axis is in the quadratic scale). A conclusion can be made from these results that the performance of ChronoChat stays practically unaffected if the network experiences moderate levels of random losses (≤1%). Moreover, even if network conditions deteriorate and random losses increase to abnormally high values (5%–10%), ChronoChat continues to show significantly shorter state synchronization delay, compared to IRC-like systems.

Overall, regardless of the random loss rate value, more messages in ChronoChat experienced smaller delay, compared to those in IRC. This trend is more clear as the loss rate grows: the percentage of messages with small delay drops rapidly in IRC and in ChronoChat it drops more gracefully. However, a careful

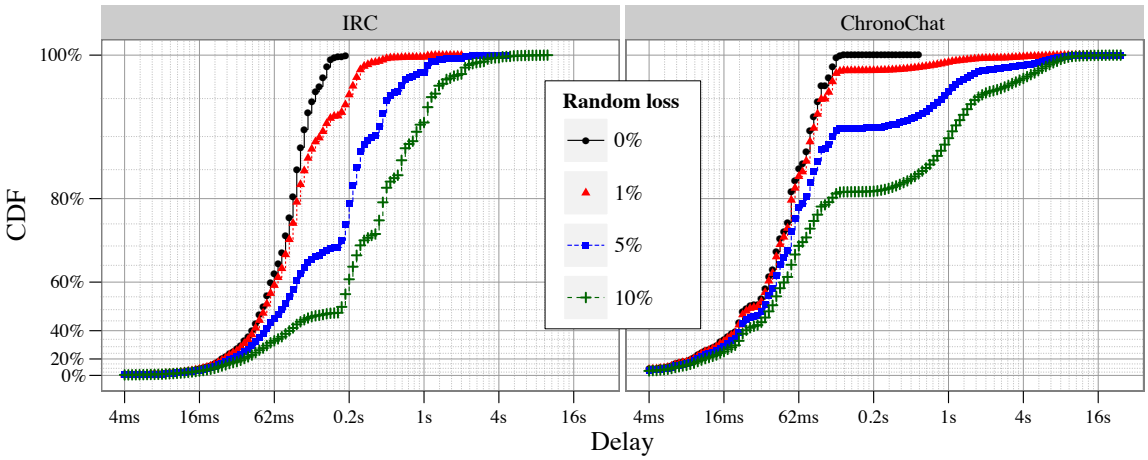


Figure 5.13: Packet delivery delay in face of packet losses

reader may note that there is a small fraction of messages in ChronoChat that experienced longer delay, compared to IRC. This is because ChronoChat uses NDN's pull based model: a receiver needs to discover a new state first in order to request for it, as opposed to TCP/IP where the source keeps (re-)sending the data packets until it is acknowledged by the receivers. In cases where the sync Interests or sync data are dropped so heavily that some participants are not aware of the state change, it has to wait until these participants re-express sync Interests or another state change occurs before the message can be disseminated to all users. We believe that adaptive adjustment of sync Interest re-expression interval, depending on application requirements and network conditions, should be able to keep synchronization delay within reasonable ranges.

5.2.7.2 Synchronization resiliency to network failures

Another key feature of ChronoSync is its serverless design, which means that users can communicate with each other as long as they are connected. Even in the case of network partitioning, the group of participants in each partition should still be able to communicate with each other, and when the partition heals, different groups should synchronize the chatroom data automatically.

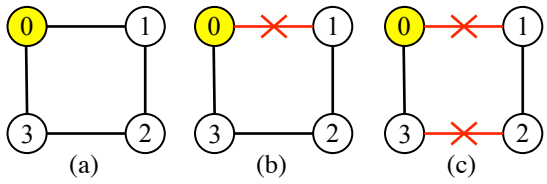


Figure 5.14: Simple 4-node topology with link failures (link delays were chosen uniformly at random in the interval 1–2 ms)

Basic verification of link failure resiliency

To verify this property we conducted a small-scale 4-node simulation with link failures and network partitioning (Fig. 5.14). The total simulation time of 20 minutes was divided into 5 regions: 0–200 seconds with no link failures (Fig. 5.14a), 200–400 seconds with one failed link between nodes 0 and 1 (Fig. 5.14b), 400–800 seconds with two failed links between nodes 0, 1 and 2, 3 (partitioned network, Fig. 5.14c), 800–1000 seconds with one failed link between nodes 2 and 3, and finally 1000–1200 seconds period with no link failures.

The results are depicted in Fig. 5.15, visualizing node 0's knowledge about the current states of all other participants as a function of time. This figure not only confirms that the parties within a connected network continue to communication during the partitioning event, but also the fact that when the network recovers from partitioning, the state is getting synchronized as soon as Interests start flowing through formerly failed links.

Impact of link failures

To quantify the effect of network failures on ability of text chat participants to communicate with each other, we again used our 52-node topology that is now subjected to varying level of link failures. In each individual run of the simulation we failed from 10 to 50 links (different set of failed links in different runs), which corresponded to ≈10% and ≈50% of the overall link count in the topology. We performed 20 runs of the simulation for each level of link failures, counting the

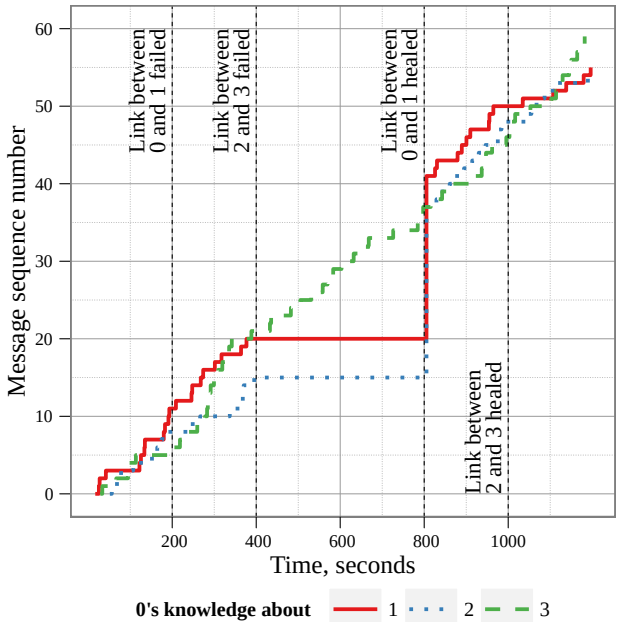


Figure 5.15: ChronoChat performance in face of link failures (sequence number progress)

number of pairs that are still able to communicate. As shown in Fig. 5.16, we use a violin plot[5] for this graph to highlight a bimodal nature of the distribution for the percent of communicating pairs in the centralized IRC service: with significantly high probability the users were almost not able to communicate at all (notice the portion of the violin plots near the bottom of the y-axis for IRC). ChronoChat, being completely distribute, always allows a substantial number of pairs able to communicate. For any centralized implementation, like IRC, there is always a single of point of failure and the communication can get completely disrupted even with a small level of link failures.

5.2.7.3 Network utilization pattern

To understand how the fast state synchronization and robustness to links failures in ChronoSync relates to the network utilization, for the same sets of experiments

[5]The violin plot is a combination of a box plot and a kernel density estimation plot. Wider regions represent higher probability for samples to fall within this region.

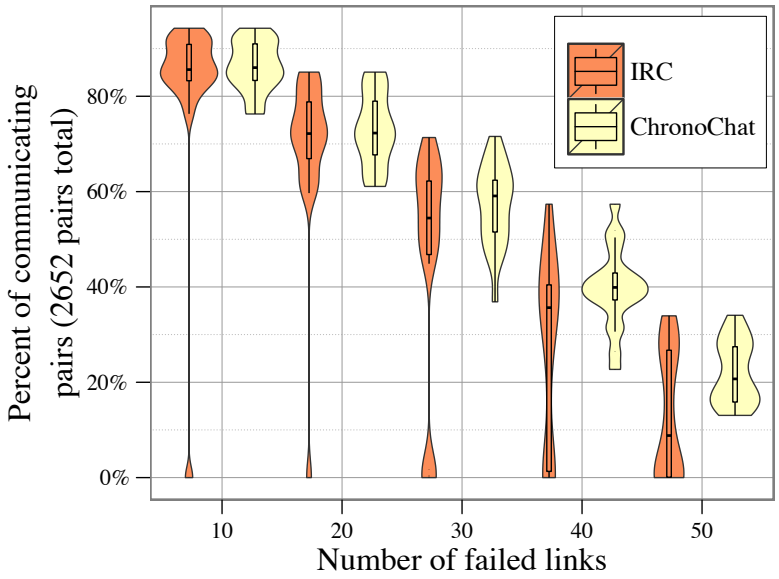


Figure 5.16: Distribution of the number of communicating pairs versus number of failed links (violin plot)

we collected statistics about the number of packets transferred over each link in the topology (we call it *packet concentration* for a link). When counting the packets, we included both Interest and data packets in ChronoChat, and both TCP DATA and ACK packets in IRC. The obtained data for our 52-node topology experiment is summarized in Fig. 5.17,[6] where the links were ordered and visualized by the packet concentration value (with 97.5% confidence interval).

The results presented in Fig. 5.17 show that ChronoChat more or less equally utilizes all of the available network links between participants. [7] Results for network utilization in IRC case show a completely different pattern. A few links close to the server have high packet concentrations, with value as large as ≈90,000 packets (≈90 times of the total number of messages in the chatroom) in the link directly adjacent to the server. Many links that are close to clients have a low

[6]the figure summarizes data about experiments under ideal network conditions, but results in lossy environments show similar trends

[7]When not all nodes participate in chat sessions, the Interest forwarding strategy would ensure that links that are not on the path between participants, will not be unnecessarily utilized. The specific implementation of a such strategy is one of our future research directions.

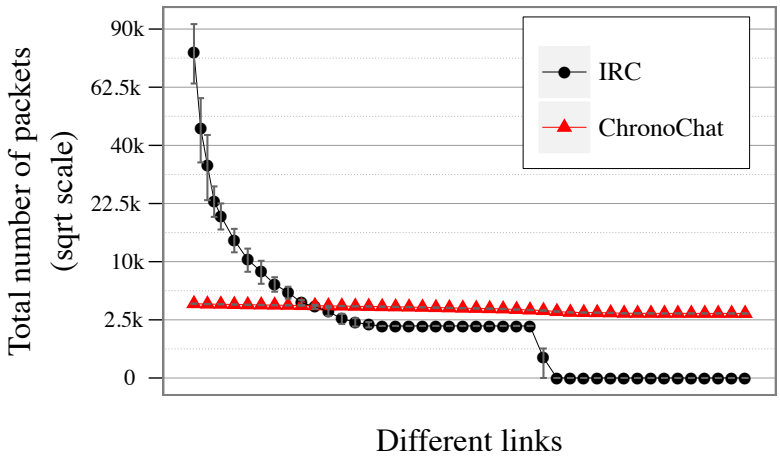


Figure 5.17: Number of packets in links (packet concentration)

packet concentration, while some links, which are not on the shortest path between clients and the server, are not utilized at all.

5.2.7.4 Overall overhead

The difference between network utilization patterns in ChronoChat and IRC highlights an important design trade-off of ChronoSync protocol. As the primary objective of ChronoSync is to synchronize the state in a complete distribute manner as fast as possible, and with ability to mitigate network failures, it utilized more links in the topology compared to IRC. At the same time, as ChronoSync does not have triangular data distribution paths and NDN architecture ensures that each piece of data travels over a link no more than once, the overall overhead in ChronoChat can be even lower than that of the centralized solutions which are generally considered to be efficient in network utilization. For example, the cumulative sum of packet concentrations presented in Fig. 5.18 shows that in our experiments, where sync Interests are distributed by broadcast, ChronoChat still has considerably lower overall overhead compared to that of the IRC service.

Note that ChronoSync also features application-specific trade-offs, which can be directly related to the overall overhead. In particular, when an application

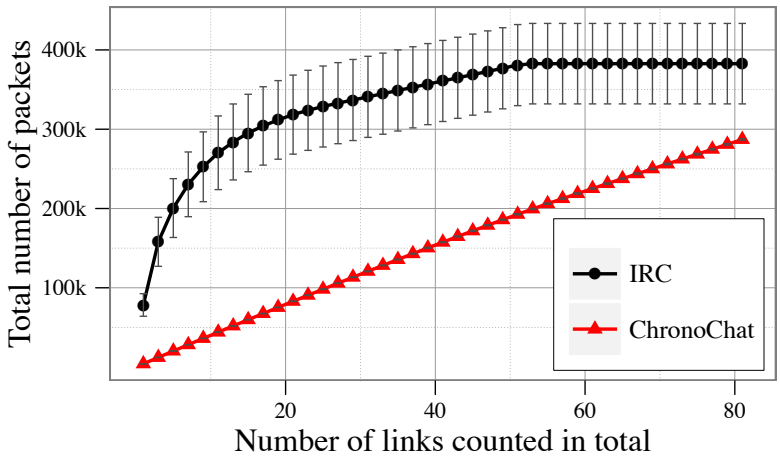


Figure 5.18: Cumulative sum of per-link packet concentrations

seldom generates new data and can tolerate certain synchronization delay, it is not necessary to always keep an outstanding sync Interest. Instead, the sync Interests can be expressed with longer intervals to reduce the overall overhead.

5.3 Securing Distributed Applications

There is also a need to provide security assurance to distributed applications. Most applications today secures communication channels using protocols such as SSL/TLS [FKK11, DR08] and IPSec [KS05]. Very often, such conventional approaches heavily rely on the centralized controllers, which does not align well with the nature of distributed applications.

In this section, we propose a completely distributed and data-centric security design to achieve data provenance and access control in the absence of a central controller.

As described in [ZEB10], NDN distinguishes the *use* of public keys, i.e. encryption and signature verification, and *trust management*, which provides an infrastructure for users to verify the public keys. NDN assumes that each party is associated with one or multiple keys and each application uses those keys to

secure data. Trust management, on the other hand, is not confined within specific applications, and is subject to different policies by different people and different organizations. Therefore, trust management can and should be provided as separate and independent component[8]. Assuming the trust relationship is established, the data provenance and access control are managed through the use of public keys, rather than by setting up sessions from the central controller.

5.3.1 Data provenance

All NDN Data packets are digitally signed, and the name in each Data packet is cryptographically bound to the corresponding packet content. This ensures both the integrity and provenance of each Data packet.

5.3.2 Access control

We propose an encryption-based access control scheme that allows only the eligible participants to decrypt the information about a dataset shared by a group. We call the user who initiated a sharing group the *"organizer"* for that group. For example, the organizer could be the one who started a conference in ACT, or the one who created a shared folder in a file sharing application. We assume that organizer knows the identity of all users who are allowed to join the group it creates. Organizer is the only entity with the permission to add or remove participants and to devise and enforce the access control policies.

5.3.2.1 Participants control

When an organizer starts a sharing group, it generates a public/private key pair (K_{ge}, K_{gd}) to distribute confidential information within the group, where K_{ge} is used for encryption while K_{gd} is used for decryption. Confidential information is

[8]There is ongoing research on trust management in NDN.

encrypted by organizer using K_{ge} and can only be accessed by those who obtain K_{gd}.

Organizer keeps the encryption key secret and distributes the decryption key K_{gd} to all legitimate participants in encrypted form to prevent outsiders from accessing it. An example of a typical Data packet to securely distributed the decryption key is shown in Figure 5.19. We hence forth call such data that determines who are the eligible participants the *"participants control data"*.

All encryption of K_{gd} (one per participant) are included in a single Data packet. Although doing so increases the packet size, it allows a better utilization of the multicast and caching capabilities built in NDN. The hash values of the eligible participants' public keys are also included together with the encrypted K_{gd}. In this way, each user can determine whether he/she is among the legitimate participants without performing any decryption.

The underlying encryption scheme must prevent users with the knowledge of K_{gd} to determine the value of K_{ge}. This can be achieved using RSA-OAEP [BR94]. In particular, given $N = p \cdot q$ where p and q are safe primes, in our instantiation the encryption exponent e (only known to organizer) is chosen uniformly at random from all the values $1 < e < \phi(N)$ such that e is co-prime with respect to $\phi(N)$. Unfortunately, this does not allow us to adopt some of the common optimization related to RSA [JK03]. It is not possible to select an exponent e with low hamming weight, since participants would be able to determine its value based on the knowledge of N.

5.3.2.2 Application data encryption

Usually the application data, such as voice data in a conference call or files in a shared folder, is much higher in volume compared to participants control data. Moreover, while there is only one organizer for each group, there are likely multiple

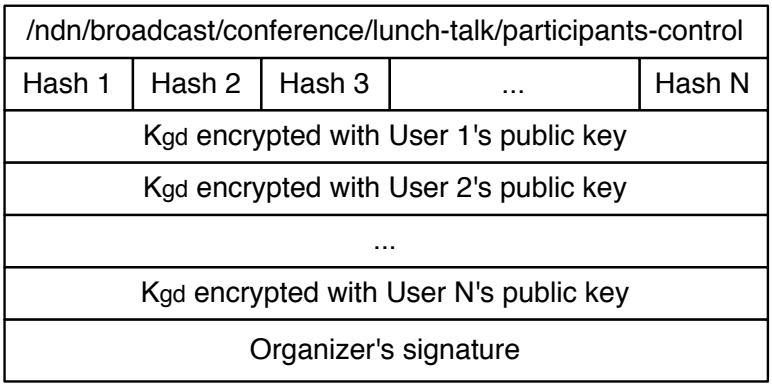


Figure 5.19: An example of participants control data

data producers; in many distributed applications all users produce data at some time. Therefore the asymmetric encryption approach used in securing participants control becomes infeasible for securing the actual application data, mainly due to two reasons. First, according to the protocol above, each data producer has to generate a key pair (K_{pe}, K_{pd}) and distribute K_{pd} to all other participants. This requires each producer to have complete knowledge of the other participants in the group, which may not be the case. Besides, letting each producer distribute a private key also incurs significant overhead. Second, asymmetric encryption imposes higher computation overhead compared to symmetric encryption. Doing asymmetric encryption for each Data packet raises concerns about the computation overhead, especially on devices with limited resources (e.g. smart phones, tablets).

Based on the above consideration, we propose to use symmetric keys for application data encryption. Organizer establishes the key for application data, which is encrypted using K_{ge}, and participants obtain and use the same key to decrypt data from others and to encrypt their own packets.

5.3.2.3 Key revocation

Key revocations can be used to force selected participants to leave a group in distributed applications. All participants in a group should keep an outstanding Interest for new participant control data and the organizer can generate such a Data packet at any time for the key revocation. All the users that are still eligible for participating will fetch the updated keys immediately.

In order to distribute a new asymmetric key pair for a conference, organizer uses the current K_{ge} to encrypt the participants control data, which indicates the asymmetric key revocation and includes an new key K'_{gd}, encrypted using eligible users' public keys. K_{ge} is used to encrypt the data so that the participants are assured that the key revocation is legitimate, as the conference organizer is the only one who knows K_{ge}. The recipients then check whether they are still allowed to participate and, if they are, successfully decrypt K'_{gd}.

To issue a new symmetric key, which supersedes the current one, organizer simply distribute the new key encrypted with K_{ge}.

5.4 ChronoShare: a ChronoSync-based File Sharing Application

To validate that the new design patterns can support complicated distributed applications, in this section we describe the design and implementation of ChronoShare, a distributed secure file sharing application based on ChronoSync. There are quite a few products providing similar services, like Dropbox, SparkleShare, Google Drive, SkyDrive, Bittorrent Sync, etc. [Dro, Spa, Gooa, Sky, Bitb], due to a tremendous demand for file sharing among multiple personal devices and among people who work on the same project. Unlike most other products which are based on centralized design, ChronoShare is completely distributed, allowing

Term	Definition
Shared folder	The folder that contains the files to be shared.
Participant	Usually a user account on a device.
Sharing group	A group of participants that can access the shared folder.
Organizer	The participant that started a sharing group and has the authority on determining who is eligible for accessing the shared folder.
Folder state	The state of the set of files in the shared folder, including file paths, version, owners, timestamps, etc..
Action	A change made by a participant to the file state, usually by modifying a file; for example, updating/deleting a file.

Table 5.2: Terminology for ChronoShare

service to continue working without requiring a particular device to the present.

For the sake of simplicity in writing, we first define the terminology in Table 5.2.

5.4.1 From ChronoSync to ChronoShare: the overview

Starting from an initially empty shared folder, the state of the folder is altered whenever an action from a participant is applied. Thus, the current state of the folder can be determined by applying all actions from all participants in a deterministic way to an empty folder.

Participants in ChronoShare name their actions to the shared folder sequentially under their unique name prefixes, and ChronoSync tracks the actions generated by all participants, and keeps a log about the most recent sequence number for actions of each participant, as show in Figure 5.20.

Whenever a participant changes the state of the shared folder, the file monitoring daemon creates an action according to the change, and ChronoSync ensures

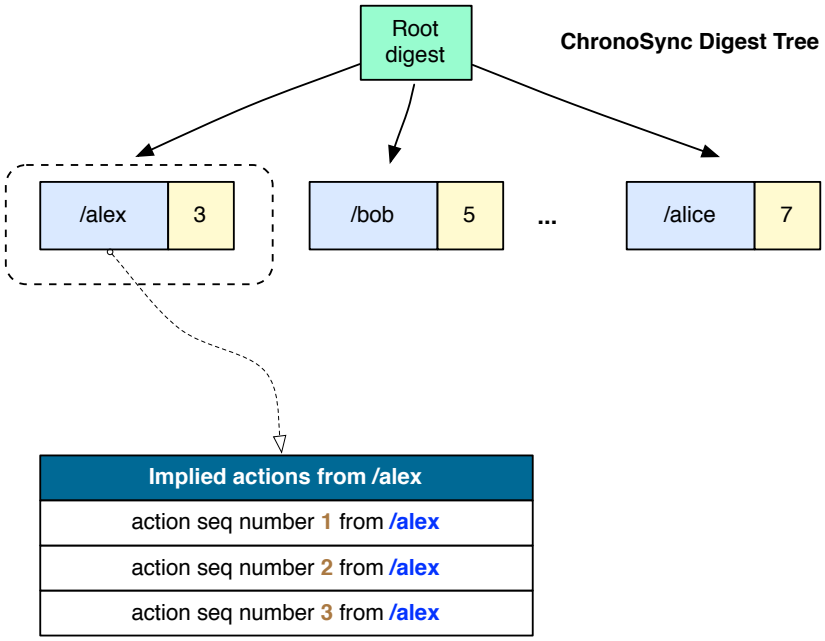


Figure 5.20: Using ChronoSync to track actions to shared folder

that others in the group immediately learn the participant's latest sequence number for action data. Others then fetch the actual action data, which describes in detail what changes have been make to which file, and apply the same action to the shared folder on their local file system. We further enforce measures to guarantee that participants always apply the same set of actions in the same order to their local folder, and as a result, they maintain the same set of files in the shared folder.

Figure 5.21 illustrates the overall picture of ChronoShare's working mechanism. ChronoSync maintains the digest tree and digest log, from which the producer status of action data for each participant is stored. From the digest tree all the names of the action data from all participants can be inferred. After the actions are fetched, an *"action log"* stores all the action data generated by the participants in a sharing group, and the shared folder on each participant's local file system is updated according to the actions in the action log. For example, the shared folder tree shown in Figure 5.21 is the result of applying the actions in the

action log to an empty folder.

Version control function is also provided so that users can always checkout previous versions of files.

5.4.2 ChronoShare design

The detailed design of ChronoShare components is as follows.

5.4.2.1 Action log

The action log keeps record of all the actions applied to the shared folder. An example of the action log is shown in Table 5.3. Each row describes an action to a single file. The name uniquely identifies an action, which could be taken directly from name of the action Data packet. The type field has two values, *UPDATE*, which creates a new file or update an existing file, or *DELETE*, which deletes an existing file. The filename field indicates the relative file path in the shared folder on the file system. The version and parent action fields are used in version control (see Section 5.4.2.3); when a file is created, the action has version number zero and no parent action; the version number is incremented whenever an action is applied to the file, and the parent action records the name of the previous action. The next field is the file data name prefix for NDN Data packets of the segments of the file associated with the action. The meta field keeps the meta information of the file on the file system, including timestamps, permission, file size, segments count, etc..

Actions are only for files[9], and each action would produce a new version of a file. Although this may seem inefficient in the first glance, it aligns well with the concept of the data immutability in NDN. Furthermore, it simplifies the pro-

[9]Directory structures information is maintained through the filename, which includes the path of the enclosing directory for a file. Same as Git [Git], empty directories are treated like files

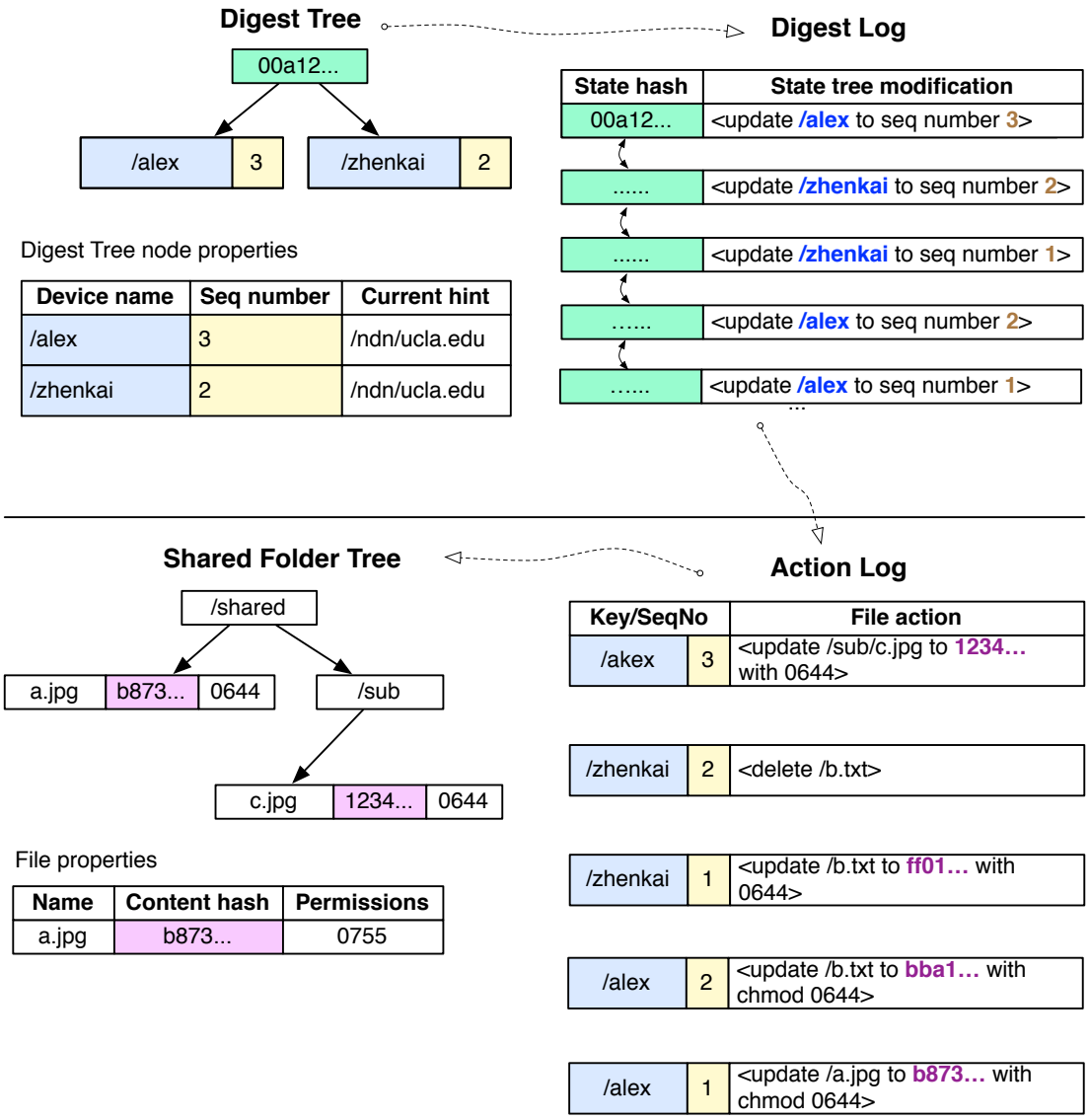


Figure 5.21: The overall picture of ChronoShare's working mechanism

Name	Type	Filename	Version	Parent Action	File Data Name Prefix	Meta
/alex/action/3	UPDATE	/sub/c.jpg	0	NULL	/alex/file/b873...	...
/zhenkai/action/2	DELETE	/b.txt	2	/zhenkai/action/1	NULL	...
/zhenkai/action/1	UPDATE	/b.txt	1	/alex/action/2	/zhenkai/file/ei26...	...
/alex/action/2	UPDATE	/b.txt	0	NULL	/alex/file/h534...	...
/alex/action/1	UPDATE	/a.jpg	0	NULL	/alex/file/093a...	...

Table 5.3: An example of the action log

cessing of the actions in that actions are independent of each other. That is, the current status of a file can be determined solely by an action, without the need to taking considerations of the previous or other actions. As a result, as long as the participants have the same view of the set of actions, they can determine according to the same rule what is the latest action for each file (and thus the latest version of file) in the shared folder, and thus maintains the same set of files.

5.4.2.2 Handling local and remote actions

ChronoShare monitors the state of the shared folder by recording in a table the information about the latest version of the files, which includes both the meta information about the file on the filesystem and the hash of the file content. When a user makes changes to the local shared folder, the file system monitoring daemon notifies the ChronoShare about the file paths that have been changed. ChronoShare compares the meta data as well as the content hash of the changed files on the file system with what stored in the folder state table to determine the action data. If a new file is added or an existing file is updated, it updates the file state table and publishes an *UPDATE* action Data packet. Similarly, if a file is deleted, the file state is updated and a *DELETE* action Data packet is published. If the action is an *UPDATE*, ChronoShare additionally needs to publish a new version of the file first. Afterwards, the ChronoSync module would immediately distribute this sequence number to other participants, prompting them to perform further operations, which is described in the following text.

Whenever the sequence number of another user's action data increases, ChronoSync triggers the fetching for the missing pieces of action data. If the file state table is changed after applying the action, and the user wishes to keep the actual files in the local file system, ChronoShare would fetch and assemble the new versions of the changed files.

5.4.2.3 Version control

ChronoShare also provides version control service, and is capable of handling conflicts about the changes to files.

To have a consistent folder state among participants in a sharing group, all participants must determine which is the latest action for a file according to the same rule.

ChronoShare enforces a partial order based on per-file versioning:

- each action for a specific file name receives a monotonically increased version number
- an action with the larger version number supersedes actions with smaller version numbers. To break tie when there are more than one actions with the same version number, "larger" version belongs to an action with numerically larger user prefix and sequence number.

The order is partial because actions for different files are not related to each other and cannot be ordered. Whenever conflicts happen, ChronoShare resolves automatically according to the orders of the conflicting actions For advisory purposes, each action also has a timestamp which indicates time when it was generated, as well as a pointer to the previous action on the same file. Both parameters are auxiliary to help users decide which version of the file is more favorable and are advisory as they only get exposed to a user when the user wants to check the history versions of a file, and the automatic conflict solving is based on version number only.

Note that under such a version control scheme, although there is only one "most recent" version of a file at any time, the history of a file is not linear. Rather it would have "branches" if multiple users updated a file before they got a chance to synchronize with each other. Thus users can always check out the

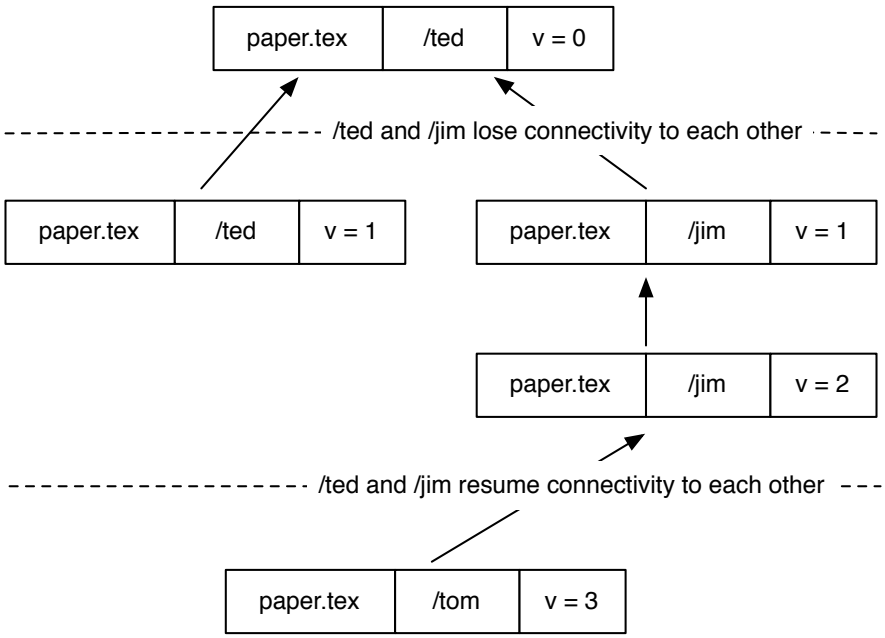


Figure 5.22: An example of branching of file history

desired version of a file if the automatic conflict resolution is not optimal.

Figure 5.22 shows an example of version control. Ted and Jim independently updated the paper when they did not have connectivity to each other. After both of them obtain connectivity, ChronoShare automatically solves the conflict and picks Jim's version as the most recent version. This decision is acceptable to the authors, so the next update action to the file has Jim's action as the parent.

Later, if the authors think Ted's version is actually better, they can always exam the history and check out his version and continue revising the paper based on that version.

5.4.2.4 Data storage and retrieval

A participant needs to store his own actions and file segments in a storage in case others send request for them later. Moreover, when receiving actions and file segments from others, a participant also puts them into the storage so that he

can serve them on behalf of others when they are not available. To retain data provenance all the actions and file segments are stored in forms of NDN Data packets, allowing recipients to follow the trust chain and verify the signatures carried in the Data packets for the provenance. When an Interest for an action or a segment of file comes, a participants checks the storage if the corresponding Data packet exists, and responds to the Interest if the corresponding Data packet is found. As a file sharing application, ChronoShare also needs to store the latest versions of the files (the raw bytes) in the shared folder on the file system. This can be done by retrieving and assembling all segments of a file from the storage.

Normally, an action or file segment can be fetched using their name directly, which would result in efficient multicast data distribution. Do note that this does not necessarily require infrastructure support. In fact, if the participants are within the same wireless broadcast domain, all communications can happen directly without any assistance from the infrastructure.

To handle the device mobility, ChronoShare uses ChronoSync to propagate the forwarding hint for a participant that is visiting a foreign network. Other participants can serve data produced by an offline participant, and thus handling the device offline problem. Although ChronoShare does not mandate such a device to be functioning, an always-on device, such as the home server or a server in the cloud, could greatly help if a large portion of the participants in a group are mobile and have intermittent connectivity.

5.4.2.5 Access control

ChronoShare uses the encryption based access control described in Section 5.3.

The organizer would collect the public keys of all eligible participants in a sharing group and disseminate the asymmetric decryption key K_{gd} securely to them while retaining the asymmetric encryption key K_{ge} to himself. A symmetric

key (such as an AES 256-bit key) is then distributed after being encrypted by K_{ge}. Only the eligible participant will be able to get the symmetric key and both the action data and the file data are encrypted using the symmetric key. The organizer should periodically change the symmetric key and use new asymmetric key pair if some participants are no long eligible to access the shared folder.

5.4.3 Implementation

We implemented ChronoShare in Mac OS X and Ubuntu with user interface similar to those provided by commercial counterparts (such as Dropbox). We use database to store NDN packets, and assembles the latest version of files to the shared-folder (but there is ongoing research to use Fuse [Fus] to eliminate the duplication storage). We use NDN-JS [STC13] based web interface for history browsing and version checkout. Additionally, WiFi ad-hoc communication is also supported.

5.5 Discussions

In this section we discuss the remaining issues with our exploration of the new design patterns.

5.5.1 Broadcast Interest in large networks

Without mandating a central node, ChronoSync relies on the broadcast sync Interests to efficiently exchange state digests. However, it is unrealistic to assume that sync Interests could be broadcasted to the parties scattered in large networks, such as the Internet. A possible solution is to build an overlay broadcast network. As shown in Figure 5.23, each network with users of ChronoSync-based distributed applications sets up a gateway node, which knows how to forward sync Interests to

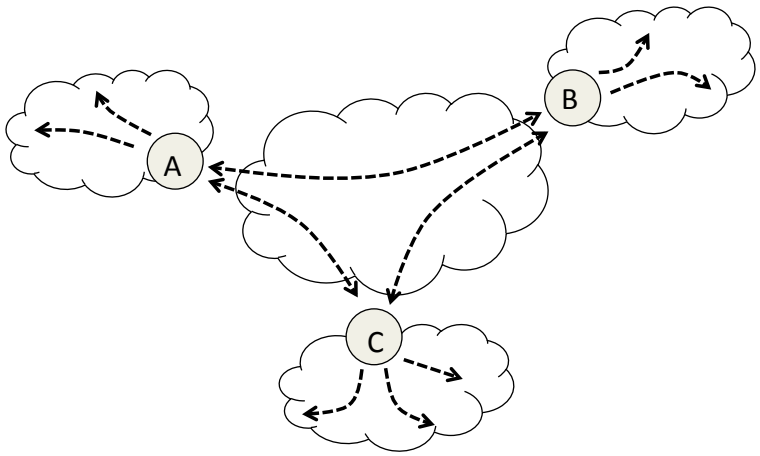


Figure 5.23: Overlay broadcast network for ChronoSync

other overlay gateways[10]. The issue of how gateways learn each other's presence is out of scope for this paper. A gateway node relays the sync Interests received from its local network to gateways in other networks. Vice versa, the sync Interests received from other gateway nodes would also be broadcasted in the local network. As a result, the broadcast of sync Interests is confined to networks where such Interests are desired to be received.

Another way to proceed is to ask the gateway routers to announce a broadcast name prefix through routing, which remotely resembles Multicast OSPF [Moy94]. It works in a similar fashion as the overlay approach, but there is not need for special handling in order to relay Interests among gateways. The concern of this approach is that it increases the routing instability.

5.6 Related Work

There is an extensive amount of research to bring multicast functionality, and in particular reliable multicast functionality [FJL97, LGT98, PSL97] to the Inter-

[10]This could be achieve either through tunneling or with forwarding hint.

net. NDN architecture, based on which ChronoSync is designed, natively solves the multicasting problem, but requires applications to be implemented using a pull-based data delivery model, i.e., users need to explicitly request for data. ChronoSync protocol gives an opportunity for distributed applications to efficiently discover names for dynamically generated data.

To some extent, the design of ChronoSync protocol was inspired by the CCNx Synchronization protocol (*ccnx-sync*): the protocol to facilitate automatic synchronization of data collections in CCNx repositories [Pro]. However, ChronoSync and ccnx-sync are completely different protocols with different objectives—synchronizing knowledge about the data collections versus synchronizing the data collection itself.

The key building block of ChronoSync design—a compact representation of the knowledge about the whole dataset as a hash value—is based on concept of Merkle trees (hash tree) [Mer]. [11] The Merkle tree is widely used in many different areas, including file systems to verify/maintain integrity of the data on disk [ZRA10], anti-entropy mechanism in distributed key/value stores [G 07], and many others.

Another component of ChronoSync design (reconciliation of knowledge about the date collection) is closely related to numerous research efforts that aim to efficiently discover differences in files, folders, and databases: RSYNC [TM96, AST02], CDC in LBFS [MCM01], TAPER [JDT05] to name a few. However, in most cases, the nature of distributed applications for which ChronoSync was designed allows efficient difference discovery without resorting to any complex state reconciliation procedures. For applications where the probability of simultaneously data generation is not negligible, ChronoSync supports the use of exclude filter to better handle such cases. If the use of exclude filter is not desirable, ChronoSync allows the use of any of the existing or new promising algorithms,

[11] While the current design uses a very simplistic form of hash tree, it can be generalized to a more canonical form of the Merkle trees.

for example the algorithm crafted by Eppstein et al. [EGU11], in addition to the simple state reconciliation approach described in Section 5.2.6.

There is also a rich literature of peer-to-peer solutions [AS04]. In general these solutions are designed to run over today's TCP/IP network and build an application level overlay to interconnect peers. Such an overlay can be subject to frequent changes as users join and leave, and are unaware of the underlying network topological connectivity. Even though some solutions offer application level multicast data delivery, the resulting data distributions tend to be inefficient due to the mismatch between the overlay and the underlay network topology.

CHAPTER 6

Conclusion

With the number of mobile devices growing by leaps and bounds, the Internet is becoming mobile. Meanwhile, the popular applications, such as Youtube, Dropbox, Google Doc, have dramatically transformed the communication patterns in the Internet, making it increasingly limiting and difficult to conform to IP's point-to-point communication model. NDN is a proposed future Internet architecture that is better suited to modern networks and aims to accommodate emerging communication patterns. However, there has been no systematic effort on providing essential building blocks, which fully exploit the benefits brought by NDN, to support distributed applications in a new era of mobile Internet. This thesis represents an important step towards addressing these challenging problems.

Recognizing the limitations of the existing IP mobility solutions, we approach the problem of mobility support from a new and different angle. Traditional IP mobility support suffers from various shortcomings as a result of TCP/IP's host-centric communication and addressing model, which does not fit the dynamic mobile networking environment and fails to accommodate emerging communication patterns. NDN, on the other hand, greatly simplifies the design for mobility support as it replaces IP's point-to-point communication model with receiver driven data delivery based on application-specified names. As a result, mobile nodes are no longer mandated to acquire IP addresses, which keep changing during the movement, and requests for data can be issued as long as there is connectivity without the hassle of host address assignment. Also, it enables mobile nodes to

communicate based on what data they need, instead of maintaining a dynamic path to a single node. Thus, mobile nodes can efficiently use broadcast wireless media and seamlessly use multiple interfaces. Moreover, each piece of data in NDN is named and signed, and hence can be cached by any node, greatly enhancing data delivery in dynamic environment and providing intrinsic data security that is no longer dependent on locations. With such features, NDN architecture naturally embraces delay-tolerant and ad hoc networks, allowing applications to operate using the same communication model regardless of the types of the networks. NDN faces similar problems as those in IP mobility support for mobile producers, because the scaling problem of routing table size still exists despite that NDN routes on names. The invaluable lessons learned from IP mobility support research can be applied to solve the problem, and NDN's flexible strategies allow consumers to devise best approaches to increase the chance of successful data fetching from mobile providers.

Although NDN architecture frees applications from the constraints imposed by IP's point-to-point communication model, it is still challenging to design distributed applications that best take advantage of NDN. To lower the hurdle of distributed applications development, we presented ChronoSync, a dataset synchronization protocol for distributed applications running over NDN networks. Leveraging on NDN's interest-data packet exchanges for fetching named data, ChronoSync effectively names the state of a dataset by its digest at a given time. Carrying the name of the dataset state, each sync interest is broadcasted to all participants in a synchronization group to solicit "data" that reports changes in the dataset. The design takes a completely distributed approach, and the resulting ChronoSync protocol removes both single point of failure and traffic concentration problems commonly associated with centralized designs. Observing the limitations of conventional security approaches that secure communication channels and heavily rely on centralized controllers, we also proposed a decentralized

and data-centric security design that is better suited to the distributed nature of applications.

Looking forward, there are many opportunities to build upon the basic framework of supporting mobile and distributed applications laid out in this thesis.

First, while we have assessed the overall picture of mobility support in NDN, there is still a need to further identify and develop strategies to serve different kinds of application requirements. For example, both vehicular networking and wireless health monitoring are a good fit to apply the NDN mobility support approach; yet, the requirements for the former emphasize on economical factors (e.g. always use the interface with lower cost), while the latter requires more considerations on energy consumption and privacy issues.

Second, applying ChronoSync to develop a wide range of distributed applications is the best way to access its capability and to identify what needs to be improved. The initial ChronoSync design emerged during our effort of developing a chatroom application to run over NDN. Since then we have also used ChronoSync in developing different applications, such as ChronoShare. We see potential of ChronoSync in supporting a variety of applications. For example, multi-party audio conferencing can leverage ChronoSync to propagate the speaker information and instruct listeners to fetch the audio streams from active speakers; resource discovery applications such as zero configuration networking can be significantly simplified if built on top of ChronoSync.

Third, given the heterogeneous nature of application requirements, this work may not provide all the solutions needed to solve the problems encountered in developing distributed applications in the wild. We hope that this work can help stimulate more discussions on the design space of distributed applications over NDN and identifying and implementing more useful building blocks.

Lastly, we have not covered analytic proof of correctness about the ChronoSync

approach. Although we have conducted extensive empirical verification of the correctness of ChronoSync both in simulations and in the NDN testbed, a sound analytic proof could provide better assurance to developers that intend to use ChronoSync for their critical services.

References

[ABH07] R. Atkinson, S. Bhatti, and S. Hailes. "A Proposal for Unifying Mobility with Multi-Homing, NAT, and Security." *MobiWAC*, 2007.

[Adi] "Adium: Open source instant messaging application for Mac OS X." http://adium.im/.

[AMZ12] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. "ndnSIM: NDN simulator for NS-3." *NDN Technical Report-0005*, 2012.

[And06] L. Andrew. "A Border Gateway Protocol 4 (BGP-4)." 2006.

[app] "Apple's App Store passes 50 billion downloads." http://www.usatoday.com/story/tech/2013/05/16/apple-app-store-itunes/2165737/.

[AS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. "A survey of peer-to-peer content distribution technologies." *ACM Comput. Surv.*, 2004.

[AST02] S. Agarwal, D. Starobinski, and A. Trachtenberg. "On the scalability of data synchronization protocols for PDAs and mobile devices." *IEEE Network*, **16**(4), 2002.

[AYW12] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. "Scaling NDN Routing." *NDN Technical Report NDN-0004*, 2012.

[bita] "BitTorrent Sync." http://labs.bittorrent.com/experiments/sync.html.

[Bitb] "BitTorrent Sync: automatically sync files via secure, distributed technology." http://labs.bittorrent.com/experiments/sync.html.

[BP93] Pravin Bhagwat and Charles Perkins. "A Mobile Networking System based on Internet Protocol (IP)." *Mobile and Location-Independent Computing Symposium*, 1993.

[BR94] M. Bellare and P. Rogaway. "Optimal Asymmetric Encryption." In *EUROCRYPT*, 1994.

[ccn] "CCNCHAT." http://www.ccnx.org/releases/latest/doc/manpages/ccnchat.1.html.

[CZW11] Stuart Cheshire, Zhenkai Zhu, Ryuji Wakikawa, and Lixia Zhang. "Understanding Apple's Back to My Mac (BTMM) Service." *IETF RFC 6281*, June 2011.

[DR08] Tim Dierks and Eric Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.2." *IETF RFC 5246*, August 2008.

[Dro] "Dropbox." https://www.dropbox.com/.

[DWF] Christian Dewes, Arne Wichmann, and Anja Feldmann. "An Analysis of Internet Chat Systems." *IMC'03*.

[DWP05] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu, and Pascal Thubert. "NEMO Basic Support Protocol." *IETF RFC 3963*, January 2005.

[EGU11] D. Eppstein, M.T. Goodrich, F. Uyeda, and G. Varghese. "What's the difference? Efficient set reconciliation without prior context." *Proc. of SIGCOMM*, 2011.

[Emp] "Empathy Internet Messenger." https://help.gnome.org/users/empathy/stable/.

[exc] "CCNx Techincal Documentation: CCNx Interest Message." http://www.ccnx.org/releases/latest/doc/technical/InterestMessage.html.

[FFM12] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. "Locator/ID Separation Protocol (LISP)." *IETF draft-ietf-lisp-24*, November 2012.

[FJL97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing." *IEEE/ACM Transactions on Networking*, December 1997.

[FKK11] Alan Freier, Philip Karlton, and Paul Kocher. "The Secure Socket Layer (SSL) Protocol Version 3.0." *IETF RFC 6101*, August 2011.

[FLM11] Dino Farinacci, Darrel Lewis, David Meyer, and Chris White. "LISP Mobile Node." *IETF draft-lisp-mn-06*, October 2011.

[Fus] "Filesystem in Userspace." http://fuse.sourceforge.net/.

[G 07] G. DeCandia et al. "Dynamo: Amazon's highly available key-value store." In *ACM SIGOPS Operating Systems Review*, 2007.

[GEN01] Antonio Grillo, Pedro Estrela, and Mario Nunes. "Terminal Independent Mobility for IP (TIMIP)." *IEEE Communications Magazine*, 2001.

[Git] "Git – fast version control." http://git-scm.com/.

[GLD08] Sri Gundavelli, Kent Leung, Vijay Devarapalli, Kuntal Chowdhury, and Basavaraj Patil. "Proxy Mobile IPv6." *IETF RFC 5213*, August 2008.

[Gooa] "Google Drive." https://drive.google.com/.

[goob] "Google I/O 2013: 900 million Andriod activations and 48 billion app installs." http://androidcommunity.com/google-io-2013-900-million-android-activations-and-48-billion-app-installs-20130515/.

[GVE00] Arnt Gulbrandsen, Paul Vixie, and Levon Esibov. "A DNS RR for specifying the location of services (DNS SRV)." *IETF RFC 2782*, February 2000.

[HH05] Robert Hinden and Brian Haberman. "Unique Local IPv6 Unicast Address." *IETF RFC 4193*, October 2005.

[HJP06] Mark Handley, Van Jacobson, and Colin Perkins. "SDP: Session Description Protocol." *IETF RFC 4566*, July 2006.

[HMY08] Xin Hu, Morley Mao, and Yang Yang. "Wide-Area IP Network Mobility." *IEEE INFOCOM'08*, 2008.

[HNG12] Frederik Hermans, Edith Ngai, and Per Gunningberg. "Global source mobility in the content-centric networking architecture." *Proceeds of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design*, 2012.

[iCh] "iChat." http://www.apple.com/support/ichat/.

[IDM91] John Ioannidis, Dan Duchamp, Gerald Q. Maguire, and Jr. "IP-based Protocols for Mobile Internetworking." *ACM SIGCOMM CCR*, 1991.

[J 02] J. Feigenbaum et al. "L^1-different algorithm for massive data streams." *SIAM Journal on Computing*, 2002.

[jab] "jabberd14, XMPP server implemented in C/C++.".

[JDT05] Navendu Jain, Mike Dahlin, and Renu Tewari. "Taper: Tiered approach for eliminating redundancy in replica synchronization." In *Proc. USENIX Conference on File and Storage Technologies*, 2005.

[JK03] Jakob Jonsson and Burt Kaliski. "Public-Key Cryptography Standards (PKCS) 1: RSA Crypotography Specifications Version 2.1." *IETF RFC 3447*, February 2003.

[JPA04] David Johnson, Charles Perkins, and Jari Arkko. "Mobility Support in IPv6." *IETF RFC 3775*, June 2004.

[JST09] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard. "Networking Named Content." *CoNext'09*, December 2009.

[Ken11] J. B. Kenney. "Dedicated Short-Range Communications (DSRC) Standards in the United States." volume 99, pp. 1162 – 1182, July 2011.

[KKK12] Do hyung Kim, Jong hwan Kim, Yu sung Kim, Hyun soo Yoon, and Ikjun Yoem. "Mobility support in content centric networks." *Proceeds of the second edition of the ICN workshop on Information-centric networking*, 2012.

[Koo09] Rajeev Koodli. "Mobile IPv6 Fast Handovers." *IETF RFC 5568*, July 2009.

[KS05] Stephen Kent and Karen Seo. "Security Architecture for the Internet Protocol." *IETF RFC 4301*, December 2005.

[Lam78] Leslie Lamport. "Time, clocks, and the ordering of events in a distributed system." *Commun. ACM*, 1978.

[LGT98] Li-Wei H Lehman, Stephen J Garland, and David L Tennenhouse. "Active reliable multicast." In *Proc. of INFOCOM*, 1998.

[MB] Mike Macedonia and Don Brutzman. "MBONE, the Multicast BackbONE." http://www-mice.cs.ucl.ac.uk/multimedia/projects/mice/mbone_review.html.

[MB97] Jayanth Myscore and Vaduvur Bharghavan. "A New Multicasting-based Archicture for Internet Host Mobility." *ACM Mobicom'97*, 1997.

[MCM01] Athicha Muthitacharoen, Benjie Chen, and David Mazieres. "A low-bandwidth network file system." In *ACM SIGOPS Operating Systems Review*, 2001.

[Mer] Ralph Merkle. "Method of providing digital signatures." *US Patent 4309569*.

[Mil13] Geoffrey Miller. "N=billions: The smartphone revolution in the behavioral sciences." http://cyber.law.harvard.edu/events/luncheon/2013/03/miller, March 2013.

[MJ95] Steven McCanne and Van Jacobson. "vic: A Flexible Framework for Packet Video." *ACM Multimedia '95*, 1995.

[MNJ08] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas Henderson. "Host Identity Protocol." *IETF RFC 5201*, April 2008.

[Moy94] John Moy. "MOSPF: Analysis and Experience." *IETF RFC 1585*, March 1994.

[MPZ10] Michael Meisel, Vasileios Pappas, and Lixia Zhang. "Ad Hoc Networking via Named Data." *MobiArch'10*, September 2010.

[Mum] "Mumble." http://mumble.sourceforge.net/.

[ns3] "ns-3: a discrete-event network simulator for Internet systems." http://www.nsnam.org.

[Per96] Charles Perkins. "IP Mobility Support." *IETF RFC 2002*, October 1996.

[Pid] "Pidgin, the universal chat client." http://www.pidgin.im/.

[Pro] ProjectCCNx. "CCNx Synchronization Protocol." `http://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html`.

[PSL97] Sanjoy Paul, Krishan K. Sabnani, JC-H Lin, and Supratik Bhattacharyya. "Reliable multicast transport protocol (RMTP)." *IEEE Journal on Selected Areas in Communications*, **15**(3):407–421, 1997.

[Ram06] Kishore Ramachandran. "Mobile IP - deployment after a decade." 2006.

[RLZ12] Ravishankar Ravindrant, Samantha Lo, Xinwen Zhang, and Guiqiang Wang. "Supporting seamless mobility in Named Data Networking." *5th Internation workshop on the Network of the Future*, 2012.

[RPT02] R. Ramjee, T. La Porta, S. Thuel, K. Varadhan, and S.Y. Wang. "HAWAII: A Domain-based Approach for Supporting Mobility in Wide-area Wireless Newtorks." *IEEE/ACM Transcations on Networking*, 2002.

[RTB97] Yakov Rekhter, Susan Thomson, Jim Bound, and Paul Vixie. "Dynamic Updates in the Domain Name System (DNS UPDATE)." *IETF RFC 2136*, April 1997.

[Sai11] Peter Saint-Andre. "Extensible Messaging and Prescence Protocol (XMPP) : Core." *IETF 6120*, March 2011.

[san13] "Global Internet Phenomena Report - 1H 2013." `http://www.sandvine.com/downloads/documents/Phenomena_1H_2013/Sandvine_Global_Internet_Phenomena_Report_1H_2013.pdf`, 2013.

[SB00] Alex Snoeren and Hari Balakrishnan. "An End-to-End Approach to Host Mobility." *ACM Mobicom '00*, 2000.

[SCE08] Hesham Soliman, Claude Castelluccia, Karim ElMalki, and Ludovic Bellier. "Hierarchical Mobile IPv6 (HMIPv6) Mobility Management." *IETF RFC 5380*, October 2008.

[Sky] "SkyDrive." https://skydrive.live.com/.

[SMW04] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. "Measuring ISP topologies with Rocketfuel." *IEEE/ACM Transactions on Networking*, **12**(1), 2004.

[Spa] "SparkleShare: self hosted, instant, secure file sync." http://sparkleshare.org/.

[STC13] Wentao Shang, Jeff Thompson, Meki Cherkaoui, Jeff Burke, and Lixia Zhang. "NDN.JS: a JavaScript client library for Named Data Networking." *NOMEN '13*, 2013.

[SXM00] Randall Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. "Stream Control Transmission Protocol." *IETF RFC 2960*, October 2000.

[Tes] "NDN Testbed." http://www.named-data.net/testbed.html.

[TM96] Andrew Tridgell and Paul Mackerras. "The rsync algorithm." *TR-CS-96-05*, 1996.

[TSR12] Gareth Tyson, Nishanth Sastry, and Ivica Rimac. "A survey of mobility in Information-Centric Networks: challenges and research directions." *Proceeds of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design*, 2012.

[TYT91] Fumio Teraka, Yasuhiko Yokote, and Mario Tokoro. "A Network Architecture Providing Host Migration Transparency." *ACM SIGCOMM CCR*, 1991.

[Val99] Andras Valko. "Cellular IP: A New Approach to Internet Host Mobility." *ACM SIGCOMM CCR*, 1999.

[vat] "vat: LBNL Audio Conferencing Tool." http://ee.lbl.gov/vat.

[WAK12] Lucas Wang, Alexander Afanasyev, Romain Kuntz, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. "Rapid Traffic Information Dissemination Using Named Data." In *Proc. of NoM'12*, 2012.

[Web] "WebRTC." http://www.webrtc.org.

[WVM06] Ryuji Wakikawa, Guillaume Valadon, and Jun Murai. "Migrating home agents towards internet-scale mobility deployments." *ACM CoNEXT '06*, 2006.

[XKW02] Wei Xing, Holger Karl, Adam Wolisz, and Harald Muller. "M-SCTP: Design and Prototypical Implementation of An End-to-End Mobility Concept." *5th Internation Workshop The Internet Challenage: Technology and Applications*, 2002.

[Y 03] Y. Minsky et al. "Set reconciliation with nearly optimal communication complexity." *IEEE Trans. Info. Theory*, 2003.

[ZEB10] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James Thornton, Diana Smetters, Beichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. "Named Data Networking (NDN) Project." *NDN Techincal Report NDN-0001*, October 2010.

[ZRA10] Yupu Zhang, Abhishek Rajimwale, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. "End-to-end data integrity for file systems: a ZFS case study." In *Proc. USENIX conference on File and storage technologies*, 2010.