Scalable Mobility Support in Future Internet Architectures

by

Xavier K. Mwangi

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Signature redacted

Author .. [illegible] ....
Department of Electrical Engineering and Computer Science
May 29, 2018

Signature redacted

Certified by.. ....
Karen R. Sollins
Principal Research Scientist
Thesis Supervisor

Signature redacted

Accepted by.. ....
Katrina LaCurts
Chair, Masters of Engineering Thesis Committee

Scalable Mobility Support in Future Internet Architectures

by

Xavier K. Mwangi

Submitted to the Department of Electrical Engineering and Computer Science
on May 29, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, we present MobileNDN, a scalable design for producer mobility support in the Named Data Network (NDN) architecture. While the initial design of NDN provided support for consumer mobility automatically via its stateful forwarding plane, a solution for producer mobility was left unspecified. The mobility support scheme we propose decouples the tasks of 1) detecting whether data may exist on a mobile producer, and of 2) forwarding interest packets towards the mobile producer. First, we use NDNS to detect zones that may be served by mobile producers. Second, based on insights from MobilityFirst, we introduce a scalable global mapping service to locate mobile producers. The combination of these two components yields a mobility solution that allows NDN's forwarding to operate and scale in the face of mobile consumers and producers.

Thesis Supervisor: Karen R. Sollins
Title: Principal Research Scientist

Acknowledgments

I would like to thank my advisor Karen Sollins for her support in pursuing this work. It is under her guidance and encouragement that, as an undergraduate and a graduate student, I have been able to grow both as researcher and a person. I would also like to thank my parents Francis Kariuki and Catherine Mwangi and my brother Timothy Mwangi for their enthusiasm about my work and endless support for my pursuits.

This work was funded under NSF Grant 1413973, "NeTS: Large: Collaborative Research: Location-Independent Networks: Evaluation Strategies and Studies".

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past 20 years, the use of the Internet has evolved on several fronts. First, traffic on the Internet has become dominated by video transfers. Cisco [1] places internet video at approximately 82% of consumer internet traffic in 2016 and predicts this ratio will continue into 2021. As a result, the Internet has in principle transformed from an end-to-end communications network to a content-delivery network. Second, as a result of the shift in computing from personal computers towards mobile devices, a significant amount of Internet traffic is sent or received by mobile devices. Cisco places mobile traffic at 7% of the total internet traffic in 2016 but has projected a massive increase to 17% by 2021. However, the Internet was not designed with these two use cases in mind. Despite this growing pressure to evolve, the core architecture of the Internet has remained largely unchanged. This enduring core is the Internet Protocol (IP) which serves as the thin waistband of the Internet. Recognizing this disconnect, the NSF created the Future Internet Architecture (FIA) initiative to promote a clean-slate approach to network architecture research. Named Data Networking and MobilityFirst are two of the resulting architectures that are of particular interest in this thesis – the primary design goal of the former is to support content-retrieval and the latter, to support for mobility.

Named Data Networking (NDN) [9] is a proposed network architecture that aims to replace the current host-centric IP design with an information-centric design for the efficient dissemination of information, eliminating location-based data retrieval.

To retrieve data, the consumer expresses an interest packet to the network and the producer (or a cache) returns the corresponding data packets to the consumer. If the consumer and producer do not move, location is hidden from them through name-based forwarding of requests and use of a symmetric reverse path for data delivery. This information-centric design affords NDN several benefits including support for in-network caching and multicast delivery. The pull-based data retrieval mechanism provides built-in support for a limited form of consumer mobility through re-issuing of an interest. However, the original NDN design left support for producer mobility unspecified. While cached data near the producer's original location provides some data availability after a producer moved, it is insufficient for many scenarios including real-time communication and dynamically generated data.

MobilityFirst [14], meanwhile, is a proposed network architecture that prioritizes support for mobile objects (eg.mobile devices). MobilityFirst describes mobility at the Autonomous System (AS) level and assumes intra-AS mobility is handled locally. Whereas naming and routing are closely linked in NDN, MobilityFirst relies on their separation. A layer consisting of globally unique identifiers (GUIDs) and a global name resolution service (GNRS) is used to link human-readable names to location identifiers. MobilityFirst requires that the GNRS supports rapid updates of this mapping and low latency queries. The resulting system supports seamless mobility in the sense that inter-AS mobility results in delays that are negligible for most, including real-time, applications.

With these two systems in mind, there are two approaches one can take to develop a network architecture that supports both content retrieval and mobility:

1. Specify a producer mobility solution for NDN
2. Introduce support for efficient content retrieval in MobilityFirst.

In this thesis, we focus on a producer mobility solution for NDN. The second approach is discussed further in Section 3.2. While we aim to support both content retrieval and mobility, content-retrieval remains the primary design goal. While mobile traffic is projected to grow, ultimately the dominance of video traffic places much more pressure on the network. After all, mobility solutions for IP have slowed in development while content delivery networks have become a core component of the current internet.

The challenge of providing scalable and time-sensitive mobility for data providers in NDN hinges on the fact that NDN is intended to hide rather than expose location information. In NDN the information that reflects how to reach the authoritative or published version of named content resides only in routing tables and is managed through a routing protocol, derived from existing Internet routing protocols. At scale and for large topological distances, these approaches are well-understood to be slow, in part to damp out minute local fluctuations. In order for an interest packet to reach a moving producer, we need an alternative plan. The problem is that a mobile producer responsible for a sub-hierarchy of the global naming hierarchy moves frequently and in quantity, yet access to that moved data should be accessible seamlessly, both efficiently and quickly.

The first part of the problem is how to recognize and keep track of those points in the hierarchy where mobility is occurring. For that we depend on a name service that handles some indication of how to reach a moved publisher in the short term, an indirection to a non-mobile attachment point. This mapping solution must be fast, efficient, and consistent with NDN's design principles.

The rest of the thesis is organized as follows. Chapter 2 provides an overview of the Named Data Networking and MobilityFirst network architecture on which our design builds on and from which it draws inspiration. Next, Chapter 3 explores proposals that aim to support both device mobility and the information-centric paradigm. Following that, Chapter 4 describes the design of MobileNDN, our solution to mobility support in Named Data Netowrking. Chapter 5 uses analytical and simulation-based techniques to explore the behavior of MobileNDN. We conlcude with Chapter 6 and present future avenues of research in Chapter 7.

The primary contribution of this thesis is a scalable producer mobility solution for NDN. This solution introduces an additional mapping system for mobile data but ensures that non-mobile data does not experience this additional cost. We show that the additional latency incurred when retrieving mobile data is low enough to support even real-time applications.

Chapter 2

Background

In this chapter, we introduce the background relevant to MobileNDN. We first describe the Named Data Networking (NDN) and MobilityFirst systems in further detail, prioritizing the system components most relevant to our work.

2.1 Named Data Networking

2.1.1 Motivation

The primary goal of NDN is to support the efficient retrieval of data objects, a use not well supported by the current IP protocol. In this discussion, we consider DNS [10] [11] to be a core component of the IP architecture. DNS translates human-friendly domain names to routable IP addresses. While DNS names do not fully name data, their support for aliases and synonyms provides a loose and multihomed binding between name and data. However, the primary disadvantage of the current IP protocol is that the packets transmitted are addressed using IP address. As a result, only the endpoints are aware of the name associated with each packet. This makes optimizations such as content request aggregation and opportunistic caching of data packets impossible. As a result, IP has had to make do with duplicated traffic from data requests and with application-level caching. NDN alleviates this issue and makes these optimizations core features of its architecture.

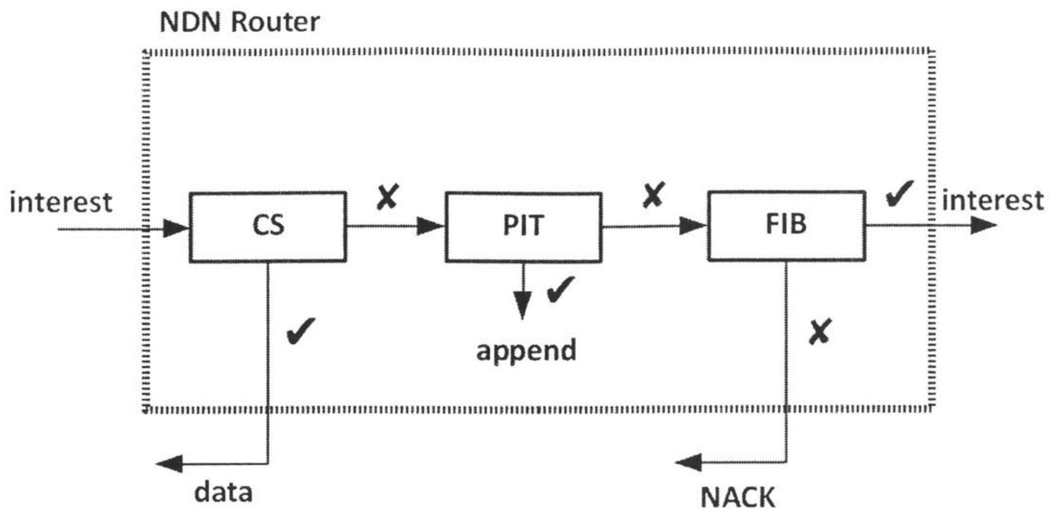

Figure 2-1: NDN Interest Pipeline

2.1.2 System Design

NDN is a network architecture that aims to replace the current host-centric IP design with an information-centric design for the efficient dissemination of information. To retrieve data in NDN, a consumer issues an *interest* packet containing the hierarchically structured human-readable name (eg. `/net/mit/www`) of the desired data. This name is used by NDN routers to forward the interest towards a copy of the corresponding *data* packet. The data packet, retrieved directly from the producer or from an in-network cache, is returned to the consumer by tracing back the path taken by the interest packet.

To support this model, every node in the network has forwarding information base (FIB) managed by a FIB update protocol, and a pending interest table (PIT). In addition, network routers contain a content store used to cache data packets by name. To forward an interest towards its producer, the FIB does longest prefix matching on the NDN name. This process leaves a trail of breadcrumbs in the PITs, used to forward the returned data packet towards the consumer. From our perspective it is interesting to note that location information is hidden in the set of FIBs along the path, and not represented explicitly and completely anywhere. The network router interest packet processing pipeline is illustrated in Figure 2-1 and involves the following steps.

1. When an interest packet reaches an NDN router, the router first performs a lookup in the content store for data which matches the requested name. If data is found, it is forwarded to the consumer along the incoming interface (eg. MAC address).
2. The router performs a lookup in the PIT for an existing pending interest with the same name. If such an entry exists, the incoming interface is added to the list of interfaces towards which the data will be forwarded when received. This is the bread-crumb trail that creates a symmetric reverse path for data delivery.
3. The router performs a lookup in the FIB to determine the outgoing interface along which the interest packet should be forwarded. If such an interface is found, the packet is forwarded. If not, a `NACK` response is sent to the consumer.

This information-centric design affords NDN several benefits including support for in-network caching and for multicast delivery. Futhermore, the pull-based data retrieval mechanism provides built-in support for a limited form of consumer mobility. That is, when a consumer moves it can simply re-issue an interest packet for the desired data. Furthermore, if the new interest packet happens to cross the path taken by the original interest packet, the new interest may be satisfied by a cached copy of the desired data.

2.1.3 Defining Mobility in Named Data Networking

Since our solution builds on NDN, we take this opportunity to clarify the meaning of mobility in this context. In an information-centric network such as NDN, the idea of producer mobility is slightly broader than the corresponding and more familiar definition in IP. In the IP network, the IP address of a mobile producer can change for several reasons. For instance, moving into a different network (and even reattaching to the same network in the case of dynamic IP) may lead to a change in IP address. Supporting seamless producer mobility in IP, therefore, involves making the current

IP address of the producer available to all consumers. In contrast, seamless producer mobility support in NDN allows consumers to retrieve data published by mobile producers. The difference between these definitions becomes more apparent when considering the various producer mobility solutions for NDN (Section 3.1). In the context of mobile devices, this definition encourages that data names not be inflexibly linked to the location of the mobile device.

2.1.4 SNAMP

Unlike MobilityFirst and IP, NDN uses human-readable hierachical names for routing. This introduces a scaling issue in NDN routing as the basic approach of using the FIB to maintain routing information requires each that router, even with name aggregation, maintain an extremely large table. Further, this increases the overhead of the FIB update protocol.

In order to scale NDN routing, SNAMP [6] divides the NDN namespace into globally routable and non-globally routable names and provides mechanism to establish an association between them. In SNAMP, there exists a set of globally routable prefixes known as the Default Free Zones (DFZ). These DFZ prefixes generally correspond to network providers or autonomous systems (eg. `/sprint`, `/att`, `/verizon`, etc.). The name prefixes outside the DFZ are not globally routable, but are routable within the context of one or more local networks associated with a DFZ. As a result, we can forward an interest packet for some non-globally routable prefix towards its corresponding DFZ context and the local network will take care of forwarding the interest packet towards the desired data. With this separation, the routers within an AS only need to contain routing information for locally reachable data prefixes. The retrieval of data located outside the AS will be guided by DFZ information.

To determine the globally reachable prefix of a data name, SNAMP uses a DNS-like system, DNS for NDN [3] (NDNS) to maintain the mapping from name prefix to a list of globally routable prefixes. These NDNS records that contain prefix to globally routable prefix mappings are called *link objects.* Upon retrieval, link objects are attached to interest packets in order to assist in routing. When a router without a default path is unable to forward an interest packet using the data name alone, it uses the globally routable prefix within the link object for forwarding. Note that if there are multiple globally routable prefixes within a link object, the first default-free router will choose the best prefix (e.g. say, using active measurements-based distance and link health information) and indicate that it should be used by the subsequent routers. Overall, data retrieval in SNAMP works as follows:

1. The consumer sends out an interest packet with the name of the desired data;
2. If the name is globally routable, forwarding proceeds as in NDN and the data is retrieved. However, if the name is not globally routable, some default-free router will be unable to forward the interest further and will return a `NACK` to the consumer;
3. The consumer queries NDNS to retrieve the link object associated with the data name;
4. The consumer attaches the retrieved link object to the interest packet and sends it out to the network.

While SNAMP was developed to scale NDN routing, it can also be viewed as a producer mobility solution for NDN: when a mobile producer moves from one network to another, it updates the link object stored in NDNS. However, as explored further in Section 4.3.1, this design does not meet our mobility goals.

2.2 MobilityFirst

2.2.1 Motivation

The primary goal of MobilityFirst is to support seamless mobility in a trustworthy fashion. The development of MobilityFirst has grown from the observation that, despite the growth of mobile devices, the existing IP architecture has limited support for mobility. First, when a mobile device moves between networks, it is assigned a new IP address which causes a break in communication visible to the protocol layers above IP. The devices that were communicating with this mobile device must

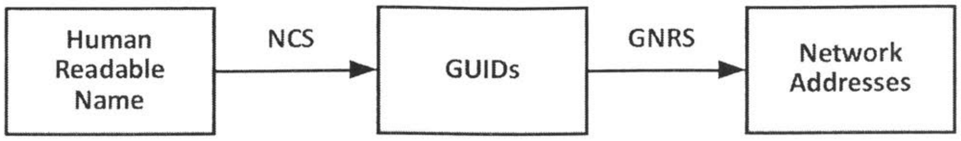

Figure 2-2: MobilityFirst Name Resolution

therefore, through some reachable intermediary, learn of the mobile device's new IP address in order to continue the communication. Second, as explored further in Section 4.3.1, the binding between human-readable names and IP addresses provided by DNS is not well suited to handling mobility.

2.2.2 System Design

MobilityFirst is a network architecture that aims to replace the current TCP/IP protocol with one that supports mobility. MobilityFirst aims to minimize delay caused by mobility events and to make mobility invisible to the application layer. MobilityFirst relies on a clean separation between human-readable names and location identifiers. Unlike NDN, MobilityFirst does not route on human readable names, but rather on network addresses (NA) such as AS or ISP identifiers. The core of the MobilityFirst protocol consists of a public-key based, flat globally unique identifier (GUIDs) namespace and a global name resolution service (GNRS). GUIDs provide a layer between human-readable names and network addresses that provides support for mobility: the map from GUID to NA managed by the GNRS can be updated. As a result, the GNRS needs to support rapid updates and queries at scale. The GNRS requirements are explored in further detail by Venkataramani et al. [17]. Section 2.2.3 briefly discusses DMap, the first proposed GNRS design.

In order to send a packet to a destination identified by a human-readable name, a MobilityFirst client must, as illustrated in Figure 2-2, resolve the name to a NA. First, a Name Certification Service (NCS) is used to securely bind human-readable names to GUIDs. For example, we may have our service provider bind the name "Phone:Xavier" to a GUID. Then, given a GUID, the client queries the GNRS for the current NA associated with the GUID. Following the example, the GUID associated with "Phone:Xavier" is then mapped to the IP address of the MIT network to which the phone is currently attached.

2.2.3 DMap

Since the introduction of MobilityFirst, several GNRS designs have been proposed. Auspice [16] and GMap [8] are among the more recent designs and rely on locality-based replication of GNRS-to-NA mappings. DMap [18], named for its use of direct mapping, is a GNRS design similarly backed by an in-network distributed hash table (DHT). However, its replication does not rely heavily on locality – it only requires that, for spacial locality, the mapping be also be stored within the current AS of each mobile object. While DMap lookups and updates have higher latency than those of Auspice and GMap, we found it most suitable for our NDN-based design because it does not impose any additional significant location-based semantics.

DMap uses K independent consistent hashing functions to map a GUID to K ASs within whose routers the map from GUID to NA will be stored. The gateway routers act as GNRS resolvers and use the information exchanged via the Border Gateway Protocol (BGP) [15] to maintain the DHT membership information and IP-to-AS mappings. Within each resolver, the consistent hashing function maps a GUID to a value in the IP space. Then, the BGP information allows the gateway to determine which AS owns the generated IP address. Note that during this process, the resolver must handle edge cases such as unallocated IP addresses

With sufficient replication, at internet scale, DMap has been shown to have query latencies below 100ms for 95% of queries. The storage overhead to support 5 Billion GUIDs with a replication factor $K = 5$ is only 173 Mbits per AS while the traffic overhead is a minute fraction of the current total Internet traffic. Overall, DMap is a relatively simple but effective GNRS design that we found useful for mapping in our producer mobility solution for NDN (Section 4.3).

2.3 Summary

1. NDN is an information-centric network architecture that uses a pull based data retrieval model
2. NDN runs into a FIB scaling issue solved by SNAMP. SNAMP uses a DNS-like mapping system to map names that are not globally routable to names that are globally routable.
3. MobilityFirst is a network architecutre that prioritizes mobility. It relies on the separation of human-readable names and location identifies, and uses a GUID layer to link these two namespaces and to enable mobility.

Chapter 3

Related Work

We explore prior work towards supporting both content retrieval and mobility within the NDN and MobilityFirst projects. As mentioned in Chapter 1, the two approaches to this end involve supporting producer mobility in NDN and supporting content retrieval in MobilityFirst.

3.1 Producer Mobility Support in NDN

There has been significant work towards introducing producer mobility to NDN. Zhang et al. [21] characterize the solutions into two broad classes within which there are two primary types of designs.

3.1.1 Mobile Producer Chasing

This approach to producer mobility involves having the consumer communicate with the mobile producer. That is, this technique relies on the interest packet reaching the mobile producer as in standard NDN. Both the Mapping and Tracing implementations rely on a fixed and globally routable node to act as a rendezvous.

Mapping

The mapping solution is similar to that found in Mobile IP [13] and involves having the rendezvous point keep track of the mobile producer's location. In effect, the rendezvous point maps the mobile producer's stable data name to its temporary location-based name. In this case, the rendezvous point can be a "home agent" closely associated with the mobile producer or, as in SNAMP, can be part of the network infrastructure.

Tracing

The tracing solutions make use of NDN's stateful forwarding plane to create a breadcrumb trail through which interests can be forwarded. When a mobile producer moves, it sends trace commands towards the rendezvous point which establishes a reverse path (within the FIB or PIT) from the rendezvous point to the mobile producer. If an interest packet crosses a matching trace, the interest will be forwarded along the trace towards the mobile producer. This solution can be thought of as an NDN-specific extension to the mapping solution which replicates the mapping information along the path from rendezvous to mobile producer.

3.1.2 Data Rendezvous

The data rendezvous solution to producer mobility is supported specifically by NDN's information-centric architecture and is, therefore, markedly different from the mobility solutions for IP. Recall that, as discussed in Section 2.1.3, mobility in NDN involves having the consumer rendezvous with the data. In contrast to the mobile producer chasing approach where the data moves along with the mobile device, data rendezvous solutions make the data available from a stable location.

Data Depot

The data depot solution is most similar to cloud storage solutions. Instead of serving the data directly, the mobile producer uploads the data onto a data depot. A data depot is similar to a rendezvous in that it attracts interests packets for data produced by the mobile producers it handles. However, instead of forwarding interest packets towards the mobile producer, the data depot is able to directly serve the data. Note that even in this case, due to in-network caching, interest packets may be satisfied before reaching the data depot.

Data Spot

The data spot solution is designed for data that can be generated by any node at a certain location. For example, traffic conditions in a certain geographic ("spot") region can be satisfied by any vehicle in the region. The mobility handled in this case is when nodes enter and exit the spot region. Routing in this scenareo is done using a geography-aware routing algorithm and having nodes check whether they are in the spot region (eg. by using GPS) before responding with the generated data.

3.2 Content Retrieval Support in MobilityFirst

MobilityFirst, as discussed in Section 2.2, is driven by the surge in mobile hosts but is designed generally to support any mobile object. As detailed by Zhang et al [20], content can be considered a first-class object in MobilityFirst. After all, the NCS that maps human readable names to GUIDs can be used to name content. The GNRS which translates GUIDs to NAs, therefore acts as a data locator service much like DNS. If we assume that GUIDs are 160 bits as in [18], then the flat namespace can support up to 2^{160} reachable data objects which is, in principle, more than enough address every byte on the current internet.

Furthermore, the transport layer of MobilityFirst divides data into large chunks whose headers contain the destination GUID. Unlike in IP where the IP address makes the requested data opaque, the GUID can be related to the requested data. Therefore, in-network storage can be used to cache data chunks. The MobilityFirst project has proposed GSTAR [12], a storage-aware routing protocol that uses in-network storage to buffer data chunks in order to overcome disconnections and link quality fluctuations. It's a testament to the difference in design priorities that MobilityFirst uses in-network storage to address disconnections caused by mobility while NDN opts uses in-network storage to cache data for accelerated retrieval.

3.3 Summary

1. In the scope of this thesis, are two general approaches to achieve both content retrieval. support and mobility support: introduce producer mobility to NDN or introduce content-support to MobilityFirst. We focus on the former.
2. Most producer mobility support solutions in NDN either chase the mobile producer or make the data produced by mobile nodes available at a fixed location.
3. MobilityFirst can, in principle, support named content retrieval.

Chapter 4

MobileNDN: System Design

In this chapter, we introduce the design of MobileNDN, a network architecture that aims to provide support for efficient content retrieval and for device mobility. To achieve these goals at scale, MobileNDN builds on the ideas in SNAMP and draws insights from the MobilityFirst project. By building on SNAMP, an extension of NDN, MobileNDN inherits the support for content retrieval that the information-centric design of NDN provides. The challenge that remains is to support producer mobility in SNAMP. In brief, we must answer the following questions.

1. How is producer mobility made visible in the network? (Section 4.2)
2. How do we determine the location of the mobile producer? (Section 4.3)
3. How does the consumer retrieve data served by a mobile producer? (Section 4.5)

This chapter begins with an overview MobileNDN's design, describing the primary modules and how they interact to support producer mobility. We follow with the mobility model, clarifying how MobileNDN keeps track of mobility information for data served by mobile producers. We then introduce caching, and in particular, detail its use in detecting mobility events quickly. We follow by specifying the mapping system that maintains and serves the location information for each mobile object. Putting this information together, we describe in further detail the process by which consumers retrieve data served my mobile producers. We close by describing a security model through which a consumer can certify that the mapping from a name prefix to a location identifier is authorized by all relevant authorities.

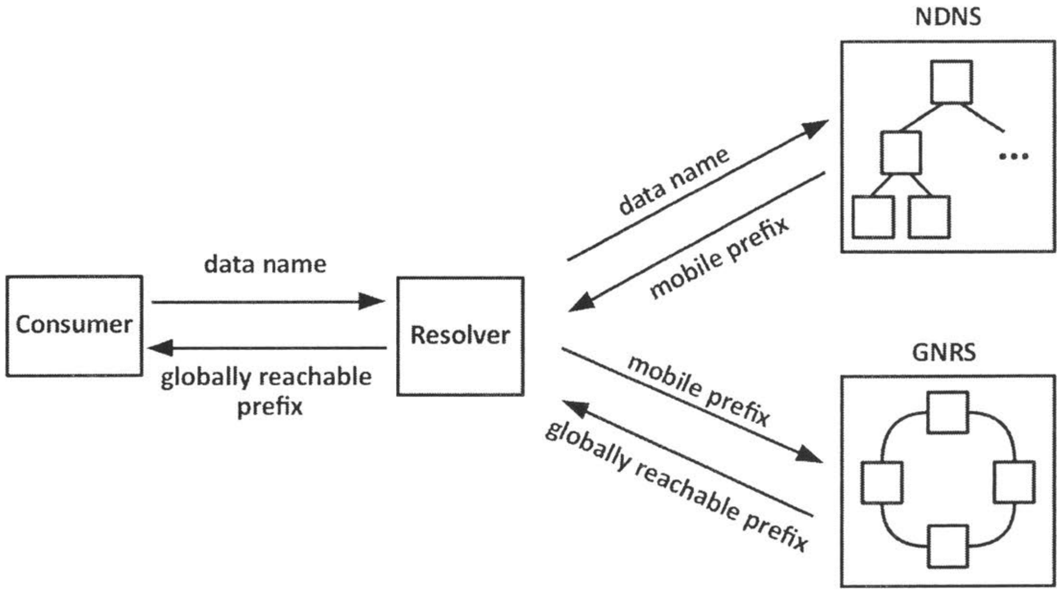

Figure 4-1: For mobile data, the the location of named data is resolved using a combination of NDNS and a GNRS. We use NDNS to determine whether the data exists under a mobile prefix and GNRS to determine the globally reachable prefix associated with the mobile prefix.

4.1 Design Overview

In order to minimize data retrieval delay in the face of mobility, we present approaches to minimize *time to detect* that a mobility event has occurred and the *time to respond* to a mobility event. MobileNDN:

1. Defines and makes discoverable the names that may be mobile, and
2. Quickly maps mobile data names to their routable prefixes.

This decoupling comes from the observation that the information required to support each of these two mechanisms have different lifetimes. In general, the fact that data exists on a mobile producer is long-lived information, while the location of the mobile producer is short-lived information. In Section 4.3, we observe that these two types of data require different mapping systems.

MobileNDN defines the name prefixes that may move and uses DNS for NDN [3] (NDNS) to maintain and serve this information. Figure 4-1 illustrates the name resolution process in MobileNDN. Then, given a data name, MobileNDN can determine the name prefix that may have moved and consults a Global Name Resolution Service (GNRS) to retrieve the current reachable prefix of the mobile name prefix. Consider the example in Figure 4-2. To ensure seamless mobility, when the mobile producer of the prefix `/net/mit/www` moves from the `/verizon` network to the `/att` network, several steps must take place.

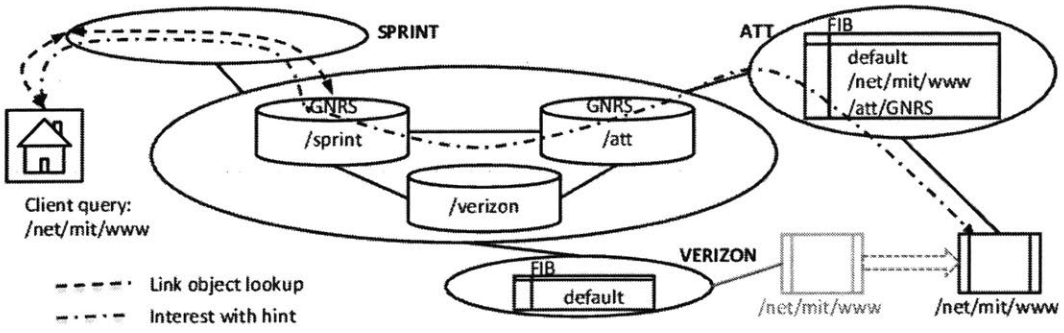

Figure 4-2: Mobility Scenario.

1. The mobile producer updates the GNRS with its new globally reachable prefix, `/att`.
2. The consumer in the `/sprint` network quickly detects that the data under the prefix `/net/mit/www` has become unreachable.
3. The consumer determines which of the three possible prefixes has become unreachable (i.e. `/net/mit/www` or `/net/mit` or `/net`). In this example, it is the `/net/mit/www` prefix that has become unreachable.
4. The consumer queries the GNRS for the current globally reachable prefix of `/net/mit/www`. The GNRS replies that the associated globally reachable prefix is `/att`.
5. The consumer attaches `/att`, a routing hint, to the interest packet for `/net/mit/www`

and sends it out again.

4.2 Mobility Model

How is producer mobility made visible in the network?

In this section, we describe what changes logically in our system when a physical device moves. We propose a model for identifying mobile portions of the NDN namespace. This definition of mobility in NDN discussed in Section 2.1.3 and the fact that we are taking a mapping approach has the following two implications to our design.

1. The consumer must be able to send an interest packet to the mobile producer. This is necessary because the only existing copy of the desired data may be located at the mobile producer. This imposes a requirement similar to that in IP.
2. In providing mobility support, we must not break NDN's data retrieval model. That is, we must use the existing hierarchical content names and take advantage of the existing forwarding and caching infrastructure. This supports the extended definition of mobility as it allows us to retrieve data from in-network caches and from data replicas.

4.2.1 Location Identifiers

In NDN, when a producer moves, the only observable change in the network is that the producer's data may not be reachable based on old routing information. In effect, all the location information is hidden within router FIBs. In small (i.e. local) networks, we can rely on a quickly converging FIB update protocol to update the FIBs when a mobility event occurs. Unfortunately, we can not rely on the FIB update protocol at internet scale. For one, a design that requires all nodes maintain routing information for all prefixes in the network runs into scale issues even if we assume aggregation of names. This problem is even worse in the context of mobility which, effectively, causes the de-aggregation of name prefixes. Secondly, even if the FIB scaling issue didn't exist, any FIB update protocol would struggle to converge in a timeframe that meets our seamless mobility goal of ~100ms handover latency. Therefore, we must introduce a mapping system from name prefixes to their location.

To reintroduce location in a way that integrates well with NDN, we follow the model described in SNAMP. In MobileNDN, there exists a set of globally routable prefixes known as the Default Free Zones (DFZ). These DFZ prefixes generally correspond to network providers or autonomous systems (eg. `/sprint`, `/att`, `/verizon`, etc.). The name prefixes outside the DFZ are not globally routable, but are routable within the context of one or more DFZ. As a result, we can forward an interest packet for some non-globally routable prefix towards its corresponding DFZ context and the local network will take care of forwarding the interest towards the desired data. This model provides a clearer understanding of how producer mobility is reflected in the network: when a mobile producer moves from one network to another its reachable prefix changes.

In order to determine the globally routable prefix associated with a name prefix, SNAMP uses NDNS to maintain this mapping. As discussed further in Section 4.3.1, using NDNS to keep track of this changing information is inadequate in our context. In that section, we describe an alternative mapping solution. For the rest of this chapter, we refer to a generic mapping system that translates a name prefix to its corresponding DFZs. However, there still remains the issue of identifying which name prefixes may move.

4.2.2 Mobile Zones

While human-readable hierarchical names for content are generally desirable, their use in NDN introduces complexity when considering producer mobility. Consider the example in Figure 4-2 where the globally routable prefix associated with the name prefix `"/net/mit/www"` changes from `"/verizon"` to `"/att"`. In SNAMP's data retrieval model, when the consumer sends an interest packet containing a stale link object, it will receive a `NACK` reply. This indicates that, even upon reaching a router within the "`/verizon`" network, the interest could not be forwarded further. However, at this point, it is unclear which prefix became unreachable. Is it "`/net`" or "`/net/mit`" or "`/net/mit/www`"? With this ambiguity unresolved, the mapping system would have quickly to determine the globally reachable prefix for each of these name prefixes.

To resolve this ambiguity, in MobileNDN, a consumer must know, apriori, which prefixes can move. To accomplish this, MobileNDN introduces *mobile zones* to NDNS. A mobile zone is a contiguous portion of the NDNS namespace that is served by at least one mobile device. Similarly, we call link objects associated with mobile zones *mobile link objects.* Conceptually, mobile zones are exactly the namespace prefixes that a mobile producer advertises. That is, during the process by which a mobile node acquires the authority to publish under a namespace prefix, that namespace prefix becomes a mobile zone. Given that NDNS does not support the rapid updates of link objects that seamless mobility requires, mobile zones may be unreachable by relying on FIB and NDNS information alone. However, the fact that a zone is mobile is information that can be stored in NDNS.

Using NDNS, a nameserver that is authoritative for a zone can report that children zones are mobile. We define a new NDNS record type, `MOBILE`, to hold mobile zone information. At a minimum, the `MOBILE` record contains a boolean value that indicates whether a zone a mobile. In our example, when "`/net/mit/www`" becomes a mobile zone, a `MOBILE` record is created and stored in the parent zone, usually reachable by "`/net/mit/NDNS`". Mobile zone discovery then proceeds, as in DNS, through an iterative resolution process.

4.2.3 Nested Mobile Zones

One may notice the definition of mobile zones leaves the possibility of nested mobile zones. For example, we may want for "`/net/mit/www`" and "`/net/mit/www/images`" to both be mobile zones, but served by different mobile producers. For simplicity, we originally hoped to disallow such a scenario but found that it imposed too large a restriction on namespace organization. Under such a constraint, to accomplish the result in the example above, the zone administrator would be required to explicitly have the mobile zones "`/net/mit/www/not-images`" and "`/net/mit/www/images`". At an extreme, this restriction may encourage names that hold non-permanent information such as "`/net/mit/mobile/www`" or worse "`/net/mit/<device-id>/www`". Therefore, MobileNDN supports nested mobile zones by resolving a data name fully using NDNS and the GNRS. When a mobile zone is encountered, the GNRS is used to locate the reachable prefix of the mobile zone, and NDNS resolution continues from within the mobile zone (assuming no zone delegation to a parent).

In addition, if we disallowed nested zones, then zone administrators would be required to change the data name in order to change the zone's mobility status. Exposing this implementation detail to consumers leads to increased complexity. Aside from making caching less efficient the primary challenge introduced is that there may be consumers in the network who rely on the original data name. This type of naming is undesirable because we hope, in general, that names in an information-centric network be long-lived.

4.3 External Link Object Store

How do we determine the location of the mobile producer?

Mobile zones and non-mobile zones each require a mapping system with different properties. As MobileNDN can identify which zones are mobile, it can handle the retrieval of mobile link objects differently. Mobile link objects change frequently and, therefore, the mapping system that maintains them must support rapid retrieval and update of link objects. Figure 4-3 illustrates the resulting link object lookup process.

4.3.1 NDNS Challenges

As discussed in Section 2.1.4, while SNAMP was developed to scale NDN forwarding, it can also be thought of as a producer mobility solution. That is, when a mobile

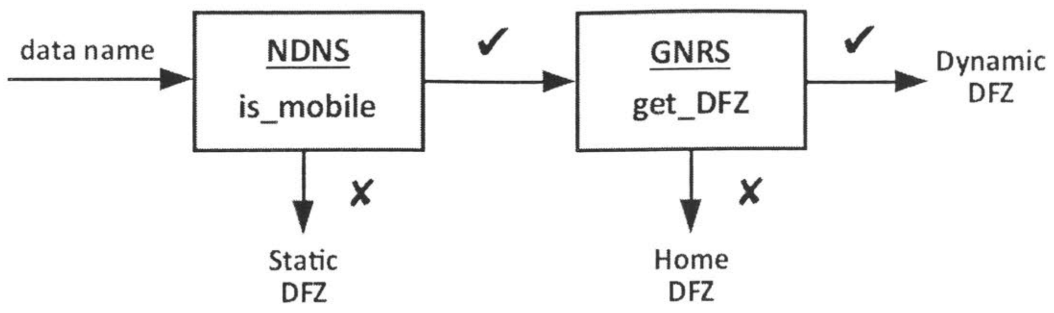

Figure 4-3: This figure illustrates link object lookup process in MobileNDN. First, we query NDNS to determine whether the data is in a mobile zone. If it is not, the standard SNAMP link object is returned. If the zone is mobile, we perform a GNRS lookup. If the GNRS lookup succeeds, we have receive the mobile link object. If it fails, we conclude that the mobile zone is at "home" and use its home link object (retrieved from NDNS). Likely, the home and static delegations will be equivalent, but this distinction is made for flexibility.

producer moves the link objects in NDNS can be updated. Equivalently, suppose that suppose mobile link objects are stored in NDNS (i.e. NDNS serves as the GNRS). Also, assume that the NDNS machines are powerful enough to handle the query and update load they receive. Even with this assumption, this approach has limitations that motivate the separation of NDNS and the GNRS. The primary issue is the use of caching in NDNS.

Caching

NDNS relies on caching to support low-latency queries. When a caching resolver resolves a NDNS name, both it and the requesting stub resolver cache the returned records. The disadvantage, however, is that this leads to stale responses: the record in the cache may be out of date. This is unacceptable for mobile link objects which are updated frequently.

No Caching

We can disable caching in NDNS by specifying that mobile link objects have a TTL lower than our mobility delay goal. However, this would require that the query travel to the NDNS server hosted locally by the mobile producer's administrator. Such an approach can not meet our latency requirements as the zone's object store might be located far from the consumer.

This issue suggests that each mobile zone must maintain a globally distributed mobile link object store with sufficient replication to support rapid responses to queries. We propose to use in-network storage for this purpose.

4.3.2 Direct Mapping of Mobile Zones

Inspired by DMap, we propose a GNRS based on direct mapping for resolving mobile zone locations in NDN. For simplicity, we assume that each prefix in the DFZ is associated with a single autonomous system (AS). As in DMap, we store the map from mobile zone prefix to globally reachable prefix within the network routers. In DMap, the gateway routers act as GNRS resolvers. In MobileNDN, given a mobile zone prefix, a gateway router uses K consistent hashing functions to generate K random values. These random values are then mapped uniformly to the known DFZ prefixes, and the mobile zone mapping stored within the routers of the corresponding ASs. To accomplish this, we have the gateway routers of each AS advertise the names `"/GNRS"` and `"/GNRS/<DFZ-prefix>"`. Note that in this design, each gateway router maintains a list of all DFZ prefixes in order to perform the uniform map from random number to DFZ prefix. This is reasonable as the gateway routers necessarily maintain this information in their FIB and can all establish a consistent ordering using lexicographical sort.

In the process of link object resolution using NDNS, if a consumer (or recursive resolver) receives a `MOBILE` record as a reply, it must query the GNRS for the link object instead. The consumer sends the query to its local GNRS resolver (found at `"/GNRS"`) which performs the translation to DFZ and forwards the query to one of the K GNRS responsible for the mobile zone (found at `"/GNRS/<DFZ-prefix>"`).

4.3.3 Home Zones and Garbage Collection

We can leverage the mobility patterns of human-owned devices to decrease the working size of the GNRS. In particular, evidence shows that mobile devices spend most of their time in a few locations [19] – their "home" networks. Within NDNS `MOBILITY` records, we can store a map from mobile zones to their globally routable home prefix. When a mobile producer moves to one of these home networks, it updates the GNRS to clear the corresponding mobile delegation information stored within. As a result, when a consumer wants to retrieve data from a mobile producer which is at home, the GNRS query returns a `NACK` which lets the consumer know that the delegation information in NDNS is usable.

4.4 Caching

In Section 4.3, we observe that NDNS alone is unsuitable for the storage of mobile link objects because it relies on caching to achieve low latency of queries. There are circumstances, however, where caching can be used safely in MobileNDN.

4.4.1 Mobile Link Objects and `MOBILITY` records

By design, MobileNDN prohibits the caching of mobile link objects by caching resolvers – they are only allowed to cache `MOBILE` records. However, we allow consumers to cache `MOBILE` records and mobile link objects. As a result, retrieving data served by a mobile producer does not require queries to the GNRS unless the mobile producer moves. This follows from the fact that name resolution only occurs in response to a `NACK`. During normal operation, the mobile link object, as would a standard link object, is reused by the consumer for retrieving data within a zone. When the consumer receives a `NACK`, it knows that its local cached copy of the mobile link object is stale and performs a lookup in the GNRS.

4.4.2 Stale Mobile Link Objects

Since we allow consumers to cache mobile link objects, they will eventually become stale. In MobileNDN, this is not an issue because the consumer performs a GNRS lookup upon receiving a `NACK`. However, we notice that it is not unlikely for an interest packet sent with a stale mobile link object to be satisfied by a cached copy of the data.

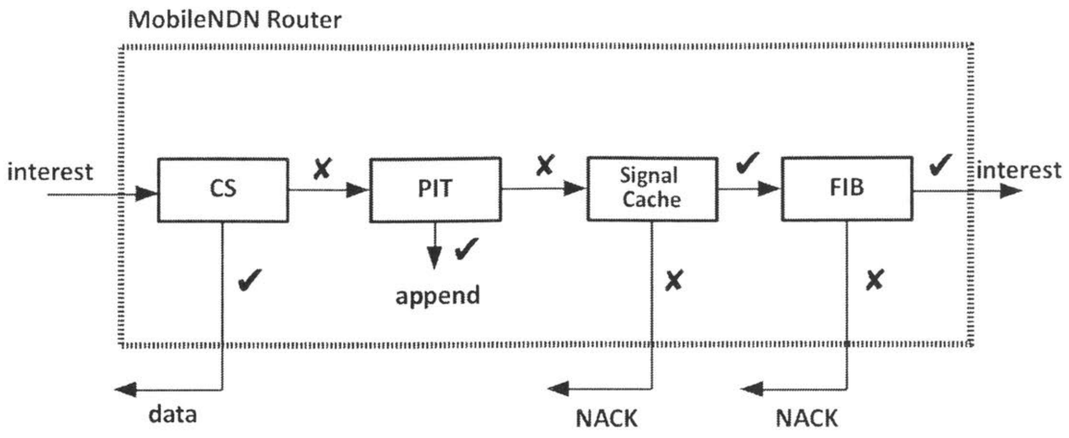

Figure 4-4: MobileNDN Interest Pipeline with Mobility Signals

After all, such an interest packet would be forwarded to the mobile producer's old AS within which a cached copy of popular data is likely to exists. In contrast, when retrieving data from a recently moved mobile producer using an up-to-date mobile link object, it is unlikely that the path taken contains a cached copy of the data. This suggests that it may be useful to, for a brief period, retain a slate mobile link object and, for each desired piece of data, send two interest packets. One interest packet will contain the new link object and the other, the stale link object.

4.4.3 Mobility Signals

We introduce the use of mobile link objects as *mobility signals*. As it stands, for the consumer to detect that a mobile producer has moved, it must first send out an interest packet containing a stale link object and then receive a `NACK`. In general, this process may incur non-negligible cost: the original attachment point of the mobile producer may be far and the lifetime of the packet can lead to a significant wait before the `NACK` arrives back at the consumer. To alleviate this issue, we allow for link objects and the time of their retrieval to be cached in routers.

To this end, we modify mobile link objects to include a timestamp of their creation time. As illustrated in Figure 4-4, when a router processes an interest packet with a link object, it also checks whether it has a link object for the same mobile zone. If the router has a newer link object, it returns a `NACK` to the consumer. If it does not have a link object for the mobile zone, or the link object it has is older than that in the link object, it forwards the interest packet as it normally would. Overall, this allows consumers to learn that a mobile producer has moved based on other consumers having recently queried the GNRS for this information earlier. This mechanism has similar characteristics to the caching of standard data within routers. That is, it is ultimately more effective for popular data and compounded by the effects of temporal and spacial locality of data requests.

4.5 Retrieving Data in MobileNDN

How does the consumer retrieve data from a mobile producer?

The process of retrieving data in NDNS differentiates between mobile zones and non-mobile zones. When a zone is not mobile, data retrieval takes place as described in SNAMP. That is, if an interest carries a name that is not in the DFZ and can not be satisfied by a cached copy along the way, NDNS takes over. A default-free router will reply to this interest packet with a `NACK`. This indicates that, to forward the packet further, the router requires additional information in the form of a link object. MobileNDN, using the NDNS, retrieves and verifies the link object. If the zone is mobile, we discover this information as the NDNS resolution takes place, and delegate part of the resolution to the GNRS. After the link object is retrieved, the consumer will place it into the original interest packet and send it out. When this modified interest packet reaches a router that is unable to forward using the interest name, it forwards by selecting the best among the prefix delegations in the link object. On subsequent NDNS resolutions for this name, we retrieve a cached copy of the `MOBILE` record which indicates that we should query the GNRS for the link object.

There are two points that we must address. First, a word is in order regarding names in the DFZ. If a prefix such as `/att/www` moves, an interest for data within this prefix will result in a `NACK`. However, standard SNAMP does not require that such names be placed in NDNS. In MobileNDN, we require that upon becoming a mobile zone, prefixes within the DFZ are also placed in NDNS and by, extension, the GRNS. Second, we may also come across a situation where FIB information is outdated and conflicts with link object information. Such a situation may occur when the consumer and mobile producer are initially in the same network, but the mobile producer moves. In this transient state, an interest packet may end up looping within a network. However, interest lifetime guarantees that eventually the consumer receives a signal that the interest has expired. We treat this signal in the same way as we do a `NACK` and rely on the FIB update protocol to guarantee that this transient state is short-lived and rare.

4.6 Security

4.6.1 Secure Delegation of Link Objects

As the designers of SNAMP noted, all critical information must be signed. MobileNDN builds on the chain-of-trust model used by NDNS. The trust model used by NDNS follows the chain of trust model used in DNSSEC [7]. In this model, all zone records are signed by at least one *data signing key* (DSK), and DSKs are signed by at least one *key signing key* (KSK). The KSKs are, in turn, self-signed (for well-known zones) or signed by *delegation keys* (D-KEY) stored in the parent zone. From this, we directly inherit certification of the `MOBILE` record stored in NDNS.

We use this model to address the following threats:

1. **An attacker may map a zone she doesn't own to a DFZ**
   We can consider each mobile device that owns a mobile zone to extend NDNS. When a mobile device becomes the owner of a mobile zone, it generates a DSK that is used to sign the mobile link objects it places in the mobile-link object store. These DSKs are then signed by the zone's KSK.
2. **An attacker may map a zone she owns to a DFZ without the DFZ's**

knowledge

We must certify that the DFZ we attach to is aware that we are routing traffic towards it. We assume there is a process by which the mobile device acquires permission to route data towards the DFZ. If our scheme involved appending link objects to the data name, this would be analogous to the process of acquiring permission to advertise under the DFZ's prefix. During this process, the DFZ signs the mobile device's DSK with a KSK. For simplicity, we can assume that DFZs are well known and can provide self-signed KSKs. If, not, however, we can rely on a chain of trust model similar to that used in NDNS.

We note that this method introduces significant space overhead as each mobile delegation must now be signed. In general, this attack is less severe than the others and can allow us to choose efficiency over security.

3. **An attacker may modify link object in-flight to and from the link object storage**

   The integrity of the data in-transit is guaranteed by having the mobile-link object signed by the mobile device's DSK.
4. **An attacker may unmap a zone she doesn't own**

   This effectively mounts a DoS attack on the mobile zone. We require each mobile-link store to verify an update before updating its records.
5. **An attacker may attempt to reverse the direction of the map** We must consider the direction of the map during during verification. For instance, keys signed by entities outside the DFZ can only be mapped from.

4.6.2 Link Object Security

SNAMP considers security carefully and observes that, even with signed link objects, the possibility of cache poisoning remains. By modifying the link object within an interest packet, a malicious router can redirect the interest towards a malicious producer. This malicious producer generates incorrect data which gets cached along the routers along the path traversed by the interest packet. While this threat can be resolved by having hop-by-hop verification of the signed link object, this approach is prohibitively computationally expensive. SNAMP chooses to cache this possibly incorrect data but to identify it using a name and link object pair instead of by the name alone. As a result, requests for the same data name but containing a different link object can not be satisfied by the cached copy. While effective in a low-mobility scenario, this security model presents some challenges.

First of all, the original (unsecured) SNAMP design [4], considered allowing in-network routers to modify the link object within an interest packet to various effects. The MobilityFirst project has a similar mechanism wherein, upon discovering that it is unable to route a packet, an in-network router queries the GNRS for the current location of a GUID. However, this can not be accomplished in MobileNDN as a result of the security model – we can not, without excessive cost, allow link objects to be modified safely. The key difference that makes this difficult is that in MobilityFirst, the discovery and use of the network address associated with a GUID is local, while in MobileNDN, the discovered link object is embedded in the interest packet and is used by all subsequent routers.

Furthermore, the use of a name and link-object pair to identify data in caches decreases the effectiveness of the cache. When the link object associated to some mobile zone changes, a consumer will transition to requesting data within the mobile zone using this updated link object. While using the stale link object, caches can satisfy interest packets. However, upon transitioning to the updated link object, the existing data in the caches will no longer be able to satisfy the interest packets. If the mobile producer is not constantly moving, the caches will be eventually repopulated with data identified using the updated link object. However, to accommodate constantly moving producers we may consider allowing the possibility of invalid data in caches and require that consumers perform verification. Ultimately, both approaches waste space in the cache, either in the form of invalid data or in the form of unusable valid data.

4.7 Summary

- We define mobile zones and make the mobile zone associated with a data prefix identifiable using NDNS.
- We use DMap to map mobile zones to their current globally routable prefix.
- We introduce the cached mobile link object information as mobility event signals.
- We present a security framework for mobile link object using the chain of trust model.

Chapter 5

Evaluation

In this section we evaluate performance and overhead of MobileNDN. We first use a comparison to MobilityFirst's operation under mobility to demonstrate the effectiveness of our mobility solution. Simulation studies show that the use of mobility signals decreases the time to detect that a mobility event. Lastly, we analyze the storage and system overhead that MobileNDN introduces and investigate how this load is distributed among the various autonomous systems.

5.1 Data Retrieval Latency

We provide a comparative evaluation of the mobility solution in MobileNDN to that provided by MobilityFirst. To compare these two systems we measure the number of *network transits*, the delivery of a packet which provides no latency guarantees. We contrast network transits with GNRS queries, the latter whose latency is generally bounded.

Consider the scenario, in both MobileNDN and MobilityFirst, where upon sending a packet that represents a request for data, the mobile producer moves before the packet can be received. We assume that the mobility event occurs after initial communication has been established and therefore ignore the startup costs of initially discovering the locations of the nodes. The results are summarized in Table 5.1

In the process of data retrieval, MobileNDN incurs four network transits and one

	Network Transits	GNRS Queries
MobileNDN	4	1
MobilityFirst	3	1

Table 5.1: Mobile transits and GNRS queries resulting from a mobility event.

GNRS query.

1. Two network transits arise in expressing an interest with a stale link object, and from the returned `NACK`.
2. At this point the consumer makes a GNRS query.
3. Two additional network transits arise from re-expressing the interest with an updated link object and from the retrieval of the requested data.

In contrast, MobilityFirst incurs three network transits and one GNRS query.

1. One network transits arises in sending the data request packet towards the mobile producer
2. Since the producer has moved, some in-network router makes a GNRS query.
3. Two additional network transits arise from rerouting the request towards the producer's new location and from the producer returning the requested data.

Notice that even if we could have MobileNDN routers perform the GNRS query and modify the link object (Section 4.6.2), we would still effectively require four network transits due to path stretch. That is, the mobile producer in MobileNDN must return data along the elongated symmetric reverse path while in MobilityFirst, the data takes a direct (ideally, optimal) path from the producer to consumer.

Although we are unable to bound network transit latency, we have some control over GNRS performance. With replication of link objects in five ASs (K = 5), DMap has been shown to achieve latencies within 100ms [18]. While GMap [8] and Auspice [16] achieve lower query latency, their reliance on geolocation makes them less natural in NDN. Overall, compared to standard operation, MobileNDN introduces the additional latency of two network transits and a GNRS lookup. In effect, this a very minor disruption in network operation that is indistinguishable from the infrequent packet loss observed in networks.

5.2 Mobility Signals

We introduced mobility signals in Section 4.4 to decrease the time to detect that a mobility even has occurred. The goal of this change is to decrease the cost of the two network transits that arise from expressing an interest with a stale link object.

5.2.1 Methodology

To perform our simulation, we used ndnSIM [5], an NS-3 based discrete time NDN simulator. We generated data traces by modeling data requests according to a Zipf distribution and assumed a uniform probability of mobility. The simulation was conducted on a generated topology of 2000 ASs with an average 10ms latency connections.

The simulation followed the mobility caching scheme detailed in Section 4.4. We found that storing this information in the cache used for data led to lower overall improvements (Figure 5-1). Therefore, we introduced a separate cache to the ndnSIM forwarder for useful mobile link object information (i.e. timestamps). Modifying this forwarder also allowed us to change the interest and data handling pipelines. As the simulation ran, we kept track of the responses to each expressed interest packet and the round trip time.

5.2.2 Results

We see in Figure 5-2, that with a dedicated cache, mobility signals decrease the time to detect a mobility event quite significantly. In particular, focusing on the low-latency regime, mobility signals allow for the `NACK` to be returned in under 100ms for over 20% of interests sent with stale packets. Without mobility signals, less than 4% of these interest packets can receive a `NACK` in under 100ms.

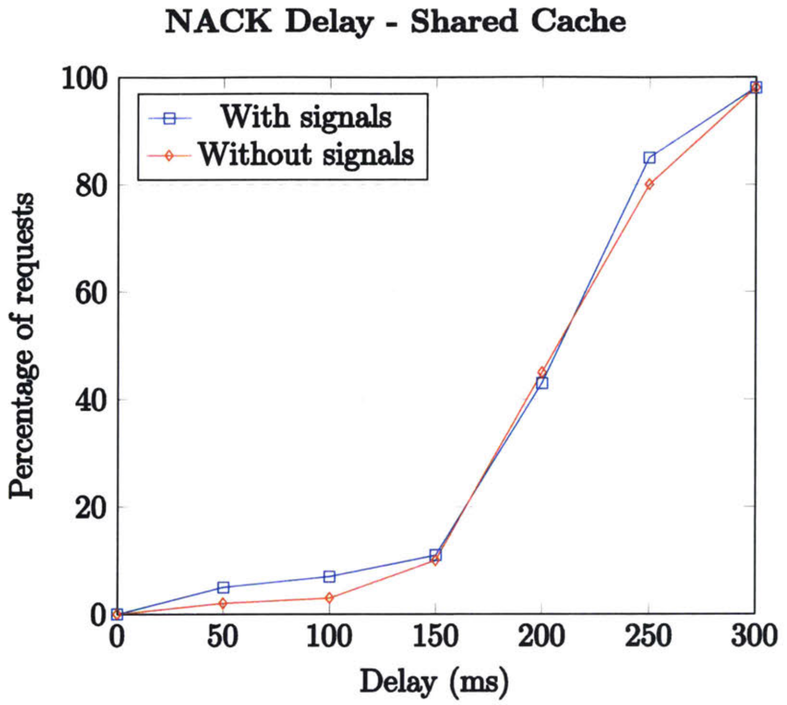

Figure 5-1: When using a shared cache, for a small percent of interest packets (corresponding to very popular and mobile data), using mobility signals decreases the delay between expressing an interest to receiving a `NACK`. This only slightly reduces the global time to detect the data producer has moved

5.3 Overhead and Load Distribution

To evaluate the overhead of MobileNDN, we first provide an estimate of the mobile link object size. In this analysis, we do not consider the cost of the discussed security mechanisms (Section 4.6), but needless to say, the inclusion of signatures and appropriately sized keys would dominate. For simplicity, we assume that names are ASCII encoded and consist of at most 10 components, each of at most length 30 characters. This can be encoded in approximately 2480 bits. To support multi-homing, we assume a maximum of four delegations per mobile zone. In total, each mobile link object will require ~1.5KB.

To estimate the total capacity required by the GNRS and load per AS, we assume a total of 10 billion mobile devices each of which advertises 10 mobile prefixes. In this use case, the total size of data in the GNRS comes out to ~150TB. If we assume NDN at internet scale is organized similarly to the current internet, divided uniformly

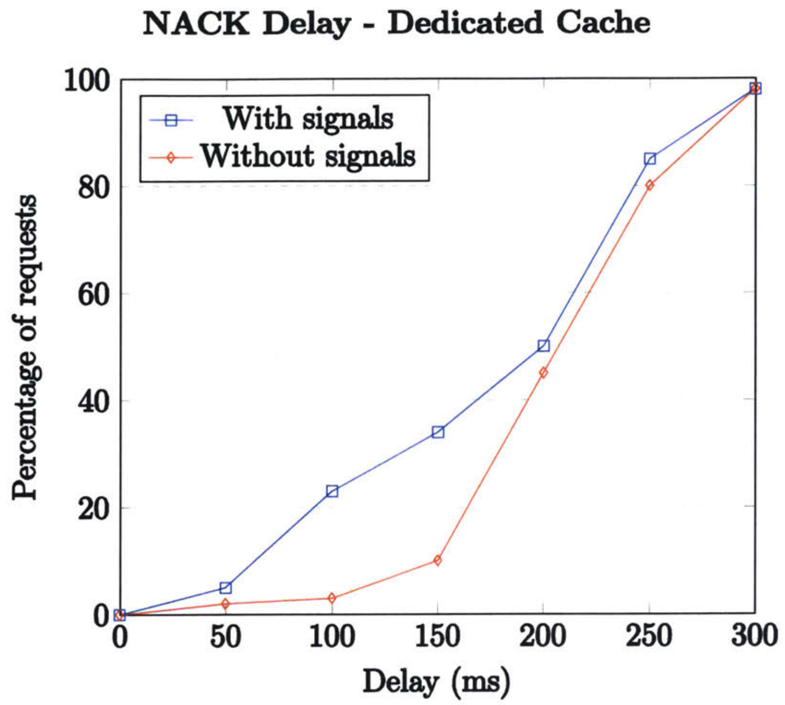

Figure 5-2: When using a dedicated cache, for a large percent of interest packets, using mobility signals decreases the delay between expressing an interest to receiving a `NACK`. This directly reduces the time to detect the data producer has moved

among the ~60000 autonomous systems, this amounts to ~2.5GB per autonomous system. Our design assumed a simple one DFZ prefix per autonomous system to achieve this uniformity, but a more sophisticated advertising scheme may break this uniformity. For example, placing popular prefixes in the DFZ would lead to ASs that serve popular data to store a greater proportion of the GNRS data.

We also consider the network traffic overhead of MobileNDN. In addition to the assumption above, we also assume that each mobile device moves approximately 100 times per day. This means that the 150TB working data set of the GNRS is updated 100 times a day which leads to a traffic overhead of ~14Gb/s. This is miniscule compared to the ~30000Gb/s of *mobile* traffic as of 2016 [2].

5.4 Summary

- Ignoring mobility signals, latency is slightly higher in MobileNDN when compared to MobilityFirst because NDN uses symmetric reverse paths for data delivery.
- Mobility signals help reduce the time to discover that a mobility event has occurred.
- The overhead of supporting MobileNDN is reasonable as each AS only needs to maintain and serve a few gigabytes worth of data.

Chapter 6

Conclusion

In this thesis we presented MobileNDN, a future internet architecture that supports efficient content retrieval and mobility. To meet these goals, we chose to focus on introducing a producer mobility solution to NDN. We described the key details of our design and evaluated its effectiveness and overhead using both analytic and simulation-based studies. The effectiveness and feasibility of the design was demonstrated using both analytic and simulation-based studies.

The primary insight is that there are two key pieces of information in the context of NDN mobility. First is whether the producer of a piece of data is mobile. This tends to be long-lived information and can therefore be stored in a system like DNS which does not support frequent updates. Second is the location of the mobile producer. This is information that may change rapidly and unexpectedly. A system like DNS is, therefore, unsuited for this purpose. However, this separation allowed us to introduce a second mapping system – a global name resolution service (GNRS).

The primary metrics of concern in this thesis were:

1. Time to detect that a mobility event has occurred.
2. Time to respond to the mobility event.
3. Space and traffic overhead.

To begin, our design exposed location information to the network by dividing the namespace into globally routable prefixes and non-globally routable prefixes. Further, we allowed some prefixes to be designated as mobile zones. The mapping from data name to mobile zone is maintained using a DNS implementation in NDN. As this resolution is part of our critical path, it influences the time to detect a mobility event. In general, caching allows for DNS queries to be sufficiently responsive. Following that, we used a GNRS implementation called DMap to map mobile zones to globally routable prefixes (location identifiers). This impacts the time to respond to a mobility event, but with appropriate replication, DMap is shown to respond to most queries in under 100ms. We then introduced mobility signals by using in-network caching of the mappings from data names to location identifiers. This aimed to further decrease the time to detect that a mobility event has occurred. Our simulation study showed that using a dedicated cache for these mobile link objects allows for over 20% detections to happen in under 100ms compared to the 4% without caching. Lastly, we demonstrated that at 2.5GB storage overhead per AS and global traffic overhead of 14Gb/s, our system has quite reasonable overhead.

Overall, we presented a design for producer mobility support in MobileNDN and demonstrated that it operates at scale with reasonable overhead.

Chapter 7

Future Work

In this chapter, we present some possible directions for future work. As a design that aims to bridge the gap formed by the different design decisions of NDN and MobilityFirst, there are areas that remain to be investigated.

First of all, an end-to-end evaluation would increase confidence that MobileNDN meets its design goals. In the IP space, we have a clear understanding of how the network is divided into ASs and the protocols that manage the intra- and inter-AS information. In SNAMP and by extension, MobileNDN, we do not have a clear understanding of how this division will look at scale. This is important because it influences metrics such as per-AS storage overhead, per-AS load and the complexity of mapping mobile zones to globally routable prefixes. Furthermore, the forwarding method that involves attaching a link object to an interest packets still requires some work. For instance, the intersection of this forwarding mechanism and security leads to some caching inefficiencies.

Secondly, there are additional features of MobilityFirst whose inclusion in NDN may be helpful. Some examples include, the hop-by-hop reliable transport and GSTAR routing algorithms which ensure reliability in the face of network disruptions caused by mobility events. In NDN, the opportunistic caching of data packets can serve as a buffer for this disruption, but the degree to which this is the case is certainly an interesting direction of study. This information would inform whether additional reliability mechanisms are necessary.

Lastly, while we did not investigate content support in MobilityFirst extensively, the architecture appears promising in supporting content retrieval. The use of in-network storage in GSTAR suggests that caching in the form supported by NDN is possible.

Bibliography

[1] Cisco visual networking index: Forecast and methodology, 2016âĂŞ2021. Technical Report 1465272001663118, 2017.

[2] Cisco visual networking index: Global mobile data traffic forecast update 2016-2021. Technical Report 1454457600805266, 2017.

[3] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang. Ndns: A dns-like name service for ndn. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9, July 2017.

[4] A. Afanasyev, C. Yi, L. Wang, B. Zhang, and L. Zhang. Scaling ndn routing: Old tale, new design. Technical Report NDN-TR-0004, UCLA, Los Angeles, CA, USA, July 2013.

[5] Alexander Afanasyev, Ilya Moiseenko, Lixia Zhang, et al. ndnsim: Ndn simulator for ns-3.

[6] Alexander Afanasyev, Cheng Yi, Lan Wang, Beichuan Zhang, and Lixia Zhang. Snamp: Secure namespace mapping to scale ndn forwarding. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 281–286. IEEE, 2015.

[7] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Dns security introduction and requirements. RFC 4033, RFC Editor, March 2005.

[8] Y. Hu, R. D. Yates, and D. Raychaudhuri. A hierarchically aggregated in-network global name resolution service for the mobile internet. Technical Report WINLAB-TR-442, Rugtgers - The State Univeristy of New Jersey, North Brunswick NJ, USE, March 2015.

[9] Van Jacobson, D. K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca L. Braynard. Networking named content. In *In CoNEXT âĂŹ09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[10] P. Mockapetris. Domain names - concepts and facilities. STD 13, RFC Editor, November 1987.

[11] P. Mockapetris. Domain names - implementation and specification. STD 13, RFC Editor, November 1987.

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

[13] C. Perkins. Ip mobility support for ipv4, revised. RFC 5944, RFC Editor, November 2010.

[14] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet. *SIGMOBILE Mob. Comput. Commun. Rev.*, 16(3):2–13, December 2012.

[15] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). RFC 4271, RFC Editor, January 2006.

[16] Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, Arun Venkataramani, David Westbrook, and Aditya Yadav. A global name service for a highly mobile internetwork. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 247–258, New York, NY, USA, 2014. ACM.

[17] Arun Venkataramani, Abhigyan Sharma, Xiaozheng Tie, Hardeep Uppal, David Westbrook, Jim Kurose, and Dipankar Raychaudhuri. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–9. IEEE, 2013.

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

[19] S. Yang, J. Kurose, S. Heimlicher, and A. Venkataramani. Measurement and modeling of user transitioning among networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 828–836, April 2015.

[20] Feixiong Zhang, Kiran Nagaraja, Yanyong Zhang, and Dipankar Raychaudhuri. Content delivery in the mobilityfirst future internet architecture. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pages 1–5. IEEE, 2012.

[21] Yu Zhang, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. A survey of mobility support in named data networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 83–88. IEEE, 2016.