2012

Prototyping & Evaluation of Content Centric Mobile Network

Xiong Xiong
Lehigh University

Prototyping & Evaluation of Content Centric Mobile Network

by

Xiong Xiong

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

Dec, 2011

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

_____

Date

_____

Thesis Advisor
Mooi Chuah

_____

Chairperson of Department
Daniel Lopresti

ACKNOWLEDGMENTS

Here I appreciate the support from my family. Without their encouragement, I would not have this honor to present this thesis. And I also appreciate the help and instructions from my advisor Dr. Mooi Choo Chuah, without whose motivation and encouragement I would not have finished this thesis. She is always helpful when I need it.

Table of Contents

LIST OF TABLES

Chapter 3

Chapter 4

LIST OF FIGURES

Chapter 3

Chapter 4

ABSTRACT

Prototyping & Evaluation of Content Centric Mobile Network

Xiong Xiong

Lehigh University, 2011

Advisor: Dr. M. Chuah

Several content centric network architectures have been proposed in recent years. However, those proposals are designed for wired network and are not suitable for dynamic mobile networks. In challenged or mobile network environments, nodes will move, appear or disappear at any time. Given such dynamic networks, it is necessary to use late binding intentional names and progressively resolve intentional names until the messages are delivered to one or several recipients.

In this thesis, we describe how we design, prototype and evaluate our secure content centric mobile network (SECON). It features a publish/subscribe based content centric delivery paradigm to preserve battery life and achieve good delivery performance. Our preliminary prototype deployed in the Emulab and ORBIT testbeds demonstrates some of the key features we have designed. We also study the content delivery performance in the ORBIT testbed.

Chapter1

Introduction

Current Internet evolves from a host-to-host communication paradigm such as ftp and telnet. At that time, computation resources were rare and expensive. Such host-to-host based applications effectively solved the problem of resource sharing. However, the internet usage today becomes more and more data centric, which means that users use internet because of its content and does not care on which machine the desired data or service resides.

Typical content creation and consumption processes today involve content discovery by the search engine, content discovery by users, and content delivery to users. Because current Internet is not content centric, search engine has very little information about when new contents appear e.g. new video streams. Often, someone has to search for a URL to trigger the search engine to look for potentially new data items at that URL. As a result, a user may not be able to discover contents with attributes that match with what he/she is looking for. In addition, network devices such as routers work at IP layer and have no concept of contents. Thus, even if two users are requesting the same content via the same router, the router will forward such requests twice, and result in delivering duplicated traffic. Future content centric network should be able to provide the ability to annotate the content such that intermediate routers can distinguish different packets, and help end users discover and retrieve their desired contents more easily.

On the other hand, new ways of media creation and consumption will emerge. Current Internet architecture will not be able to support such emerging needs. For

example, it's hard to imagine that we could provide high definition live video stream form our mobile devices to 1000 audience all over the world. This simple usage pattern may need more than one dedicated server and several gigabyte bandwidth in current Internet architecture. Future content centric network design should encourage content creation and participation by providing efficient content delivery and protection the privacy of its participants.

Similar challenges on the Internet architecture are also created by emerging location based services. Pervasive adoption of mobile devices allow location-based services offered by Foursquare, Gowalla, and Facebook Places to become parts of our daily life. However, the location based services still rely on cellular data services to provide connections between servers and their subscribers. Hence not everyone can enjoy these services. Furthermore, location based services still depend on search engines to draw traffic to them. Suppose a customer is at a mall, it is very common that he cannot access the cellular service because the cellular data service is heavily congested. Even if 3G cellular service is available, it is very possible that that customer cannot easily find the current sale promotions in this mall, because the search engine is not well optimized for finding local information that only caters for a small geographical region. In the mall example, if the user eventually locates the service, the IP packets carrying the content probably have to route through at least ten different routers from the web server in another state to the user in the mall. When another user requests the same content, this inefficient process will repeat itself. This situation can be solved if a data centric network has been deployed in the mall where the similar contents can be found at storage nodes closer to the end users.

In summary, future mobile content distribution systems should support a few key features [1]: (i) efficient content delivery, (ii) disruption tolerant networking, (iii) efficient content discovery, and (iv) data-centric security. Efficient content delivery reduce duplicated traffic in network layer and enable peer to peer mode allows close-by users to exchange interesting information without having to rely on cellular data services. Disruption tolerant networking (DTN) features [2,3] need to be supported, because of the intermittent connectivity nature of wireless devices. The intentional naming feature allows users to send messages based on attributes and not just based on fixed end point identifiers like IP addresses. Furthermore, to support emerging mobile applications, efficient content centric advertisements and search mechanisms need to be designed such that mobile users can quickly find contents (plaintext or encrypted) that they are interested in. Data centric security allows publishers to control who can access their published data items without in advance knowing who the interested users are.

Our SECON prototype features publish/subscribe based content centric disruption tolerant networking, publish announcement aggregation, content keyword search, data-centric security and intentional name. Combining these features, our prototype can meet the emerging mobile content delivery and content sharing need. Our experiments conducted in Emulab evaluates all these features. Also we use the ORBIT testbed to measure our prototype performance in wireless ad-hoc network.

Chapter 2

BACKGROUND

Here, we provide a brief overview of the major research efforts to develop a next generation networks in dynamic environment. We only elaborate on the most recent development in new content delivery mechanisms and intentional naming systems which provide us with intriguing ideas about the future mobile network design.

2.1 New Content Delivery Mechanisms

Internet was originally designed to share distributed resources around 40 years ago. e.g. share a printer attached to a single remote host in the organization), today most of the traffic on the Internet is associated with content delivery. Since resource and content access have fundamentally different properties, a new content delivery mechanisms should be designed in the Internet. Content Distribution Networks (CDNs) and Peer-to-Peer (P2P) networks are very popular content delivery mechanisms in recent years. However, due to the limitations of current Internet, these two solutions still have limited scalability and efficiency. In the past 3 years, researchers have also started doing some "Clean-Slate Design" which means they design Future Internet from scratch without considering any restrictions caused by the existing Internet e.g. [5],[8]. In this section, we merely highlight two approaches, namely the Data Oriented Network Architecture (DONA)[8] and the Content Centric Networking(CCN)[5].

2.1.1 DONA

The Data Oriented Network Architecture (DONA)[8] is designed to provide more coherent support for persistence, availability and authentication which the authors believe can improve data retrieval and service access:

- Persistence – once given a name for a data object or service, the name of that data object or service remains valid as long as the data or service is available.
- Availability – data or service should have a high degree of reliability and acceptable latency. Availability is usually provided by replication at endpoints, and the role of a network is to allow user requests to find nearby copies.
- Authenticity – data can be verified to have come from a particular source.

However, these three basic properties of data and service access cannot be easily achieved in our current host centric Internet.

DONA involves a major redesign of Internet naming. It proposes a strict separation of naming and name resolution : names handle persistence and authenticity, while name resolution handles availability.

DONA names are organized around principals [8]. Each principal is associated with a public-private key pair. Names are of the form P:L where P is the cryptographic hash of the principal's public key and L is a label chosen by the principal, who ensures that these names are unique. When a client asks for a piece of data with name P:L and receives the triplet <data, public key, signature>, the client can immediately verify that the data does indeed come from the principal by checking that the public key hashes to P, and that this key also generates the signature.

DONA will rely on a new class of network entities called resolution handlers (RHs) for name resolution. Each domain or administrative entity will have one logical RH. Each RH can be regarded as an border router in the Border Gateway Protocol(BGP). The RHs can be arranged in the same hierarchical structure (e.g. provider/customer/peer) as in the BGP protocol. RHs use local policy to process control messages related to DONA.

Name resolution is accomplished through the use of two basic primitives: FIND(P:L) and REGISTER(P:L). A client issues a FIND(P:L) packet to locate the object named P:L, and RHs route this request towards a nearby copy.

REGISTER commands set up the state necessary for the RHs to route FINDs effectively. REGISTER commands must be authenticated. A client node knows the location of its local RH via some local configuration. Any client that is authorized to publish a data item will send a REGISTER (P:L) command to its local RH. The local RH issues a challenge with a nonce, which a client (acting as a publisher P or a proxy for P) must sign with its P's private key, or sign with some other key and provide a certificate from P empowering this other key to register this piece of data. REGISTER commands have a TTL and hence data item registrations must be refreshed periodically. DONA also provides an UNREGISTER command so that clients can indicate that they are no longer serving some data. Each RH maintains a registration table that maps a name to both a next-hop RH and the distance to the copy. The REGISTER commands are transferred from local resolution handlers (RHs) to the RHs in the neighboring autonomous system(ASs) following some routing policies similar to the AS level routing policies of BGP.

Similar to the IP longest prefix matching, RHs use longest content prefix matching for selecting next-hop RH to forward FINDs. The FIND message is routed across the RHs from lower level RHs to higher level RHs just like an IP packet is routed across ASs in Internet. If RHs do not support caching, and the FIND command has been resolved to a particular server, then a direct data transfer between the client and the server will begin.

Besides describing how FINDs and REGISTERs are forwarded, the authors also discuss some extensions that can improve content delivery e.g. caching, subscriptions, and ways to avoid misbehaving or overloaded the servers.

By routing on data names, DONA can achieve reliability and self healing properties using much simpler design than the current Internet. Flat cryptographic names make informal identification harder, but they make formal authentication easier. Also, the DONA mechanism of late binding of data to the server host achieves data persistence (data is available as long as it exists).

2.1.2 CCN

Content Centric Networking(CCN) is discussed in Networking Named Content(NNC) [5, 9] which develops the idea of route-by-name further. In content access context, the users usually do not care about where the data comes from as long as the data is authentic and the data access is fast. Thus, named data is a better abstraction for today's communication problems than named hosts. The routers in today's Internet are not aware of the contents they are transferring. As a result, the same contents are requested from the

source content providers thousands of times which waste lots of valuable resources. CCN has no notion of host at its lowest level - a packet “address” names content, not location. By incorporating the idea of in-network storage and receiver-driven transport, CCN can simultaneously achieve scalability, security and performance.

CCN communication is driven by the consumers of data. There are two CCN packet types, Interest and Data. A consumer asks for content by broadcasting his interest using the Interest packet over all available connectivity. Any node hearing the interest and having data that satisfies it can respond with a Data packet. Data is transmitted only in response to an Interest. Any information about an Interest packet is removed from the network once a matching Data packet is sent to the querying node.

The CCN nodes maintain three main data structures: the FIB (Forwarding Information Base), Content Store (buffer memory) and PIT (Pending Interest Table).

In CCN, only Interest packets are routed and, as they propagate upstream towards potential Data sources, they leave a trail of ‘bread crumbs’ for a matching Data packet to follow back to the original requester(s). Each PIT entry is a bread crumb. PIT entries are erased as soon as they have been used to forward a matching Data packet (the Data ‘consumes’ the Interest). PIT entries for Interests without a matching Data are eventually timed out.

When a CCN node receives an UI(user interest message), it first matches the content name to its Content Store. If the content is found in the Content Store, then a data packet is sent back and the Interest will be discarded. If the content is not found in the Content Store, the content name will be matched against the Pending Interest Table. If

there is an exact-match PIT entry, the Interest's arrival face will be added to the PIT entry's Requesting Faces list and the Interest will be discarded. Otherwise, if there is a matching FIB entry then the Interest needs to be sent upstream towards the data. The FIB entry can have multiple outgoing interfaces. If the message arrival interface is also in the outgoing interface list of FIB entry, then the arrival face is removed from the outgoing interface list, because we don't want to send this UI message back to the previous hop. If the resulting list is not empty, the UI message is sent out all the faces that remain and a new PIT entry is created based on the Interest and its arrival face. If resulting list is empty, it is discarded (this node does not have any matching data and does not know how to find any). When the data packet that consumes the Interest arrives at the node, it is replicated, and sent out on all the interfaces that have an entry for the content in the PIT.

When a data packet is received, a CCN node first checks whether there is a matching pending interest or not. If not, it will discard this data packet. If yes, then it will insert this data packet to its Content Store and send it out to all pending interest arriving interfaces.

2.2 Intentional Naming Systems

In challenged or mobile network environments, nodes can move, appear or disappear at any time. It is not possible to keep track of network topology change. Given such dynamic networks, MIT's researchers proposed Intentional Naming System[10], a resource discovery and service location system for dynamic and mobile networks of devices and computers. P. Basu et al [6] adopts similar idea and designs a distributed

progressive resolution procedure called GRAIN that persistently delivers messages to intentional names with geographic and role attributes in the Disruption-tolerant networks(DTN) environments. Later GRAIN becomes an IETF Internet-Draft[11]. I. Stoev et al[7] implement the Lehigh University implementation of the late-binding router, called *lu-grain* which supports intentional name resolution.

INS uses an intentional name language based on a hierarchy of attributes and values, allowing nodes that provide a service to precisely describe what they provide and consumers to easily describe what they desire. INS integrates name resolution and message routing, operations that have traditionally been kept separate in network architectures. This late binding capability allows INS applications make routing decisions at message delivery time rather than at message generation time. This binding is “best-effort” since INS provides no guarantees on reliable message delivery.

INS supports intentional anycast and multicast. A message that is requested to be delivered to the “optimal” service node that satisfies a given intentional name, is considered to be an intentional anycast message. The metric for optimality is application-controlled and reflects a property of the service node such as its current load. A second type of message delivery, intentional multicast, is used to deliver data to all service nodes that satisfy a given name, for example, the group of sensors that have all recorded sub-zero temperatures.

To learn and share information about names, the INRs communicate via a name discovery protocol. The protocol uses periodic updates to convey name information, and uses triggered updates for fast changes. INS can also be optimized to implicitly learn

about names by inferring information from source message headers of messages traversing the INR network.

Different from the hierarchical organization of attributes and values in INS, BBN's GRAIN uses the ontology to express attributes and relationships. An ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.

Intentional names in GRAIN can be described using fragments of a Horn clause as used in logic programming [12]. A Horn clause is of the form "if term1 AND term2 AND ... then term"; here term1, term2, etc. constitute the body and term constitutes the head which becomes true if the body evaluates to true.

Unlike the intentional anycast and multicast features in INS, GRAIN supports three basic primitives: One, All or Any. A bundle in GRAIN can be in one of two states, STEM or FLOOD. In the STEM state, GRAIN routes the bundle toward the eventual destination while trying to use as few network resources as possible. When the bundle arrives within the target region defined by (location, distance), GRAIN switches the bundles state to FLOOD and also apply a predicate filter within that target region (location, distance) to further minimizes the network usage.

Chapter 3

SECON Design

In this chapter, we first describe our major distinguishing publish/subscribe based data delivery feature and why this feature is important in mobile content network. Then, we present detail descriptions of what intentional name is and how it works. Thirdly, we present a security solution designed by other team members in our data centric network group. Last, we describe our system architecture and detailed design for each component.

3.1 Publish/Subscribe Based Content Delivery

When talking about content centric network, we have to first discuss how one can identify a content. First, we give an example of a content name in SECON that describes a user's personal web page : "secon://lehigh.edu/~xix209/index.html". Our content names are compatible with URI[14]. It has two parts: a globally routable name such as "lehigh.edu", and a content specific part such as "~xix209/index.html". Unlike CCNx, we don't include segmentation and version in our names. Instead, we allow application layer to handle segmentation and version. We argue that the properties of a content should not just include segmentation and version. They should be very rich and expressive. So we allow a metadata descriptor to be attached to each content data. In the metadata descriptor, application can specify keywords or application specific properties.

A key difference between SECON and CCNx is that our data centric network is based on a publish/subscribe operation. We have three basic messages: user interest(UI), content data(CD) and content publish announcement(CPA). Just like CCNx, the data flow

of SECON is driven by the consumer. If there is no user interest, then no data will flow. CCNx achieved flow balance by not allowing a node to receive any content without a prior user expressed interest. CCNx is very efficient at delivering content from a central server to many consumers, but not so good at uploading contents. We argue that consumer driven data flow is not suitable in mobile computing environments and may discourage mobile users from contributing contents. The solution from CCNx is a three way content uploading process: end users who want to upload contents first sends a user interest of uploading contents to the server. If the server agrees to accept this content, then the server will send out a user interest to the client so that the data content of client can flow back to the server. In current mobile computing environments, mobile users have intermittent network connections and many of them may not have data plan and they want to off load the traffic to free Wi-Fi service. Adopting publish/subscribe based data delivery scheme can effectively solve this problem. At the publish phase, the mobile content can be uploaded to the nearest SECON server, so that the mobile publisher can upload high volume contents quickly via high bandwidth WiFi connections. At the subscribe phase, the nearest SECON server can help a mobile consumer pre-load data contents even before that consumer has immediate contact with this SECON server.

Figure [1] shows an example of publish/subscribe system for a vehicular ad hoc network environment where buses are equipped with WiFi and/or 3G devices. Current cellular bandwidth is not sufficient enough to support high definition video on demand service on a fast moving bus, however, in our publish/subscribe system, the mobile consumers can pre-load his favorite content in advance. It first sends out an encrypted and self verifying user interest to the bus he will take. The bus will forward this user

interest and pre-load the content when it has high bandwidth connectivity such as being connected to a Wi-Fi access point at bus terminal. When the mobile content consumer gets on the bus, he can immediate retrieve the content via the Wi-Fi access point on the bus. Because the user interest acts as a subscription, any new update on the remote content server will be immediately reflected in the local SECON router. The popularity of the Apple Push Notification Service proves that publish/subscribe system is very important to preserve battery life and maintain performance[13]. The mobile content consumer can also browse popular contents on the bus and view the contents which he is authorized to view. If the mobile content consumer wants some contents which is not pre-loaded on the bus, he can request this content via his own data plan.

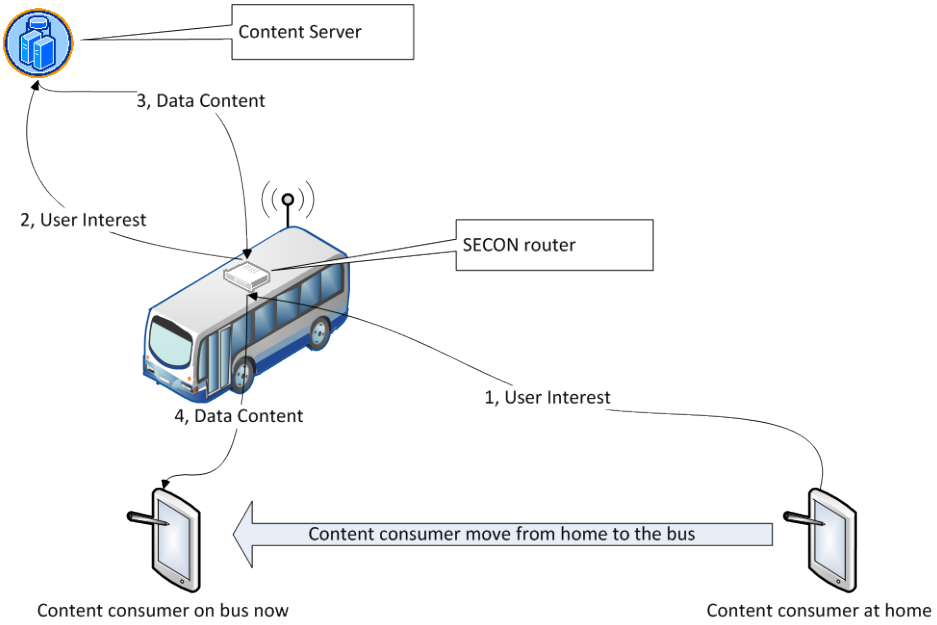

Figure 1 Bus example

Similarly, a mobile node can be a publisher, he can off load the content to the SECON router on the bus and let the bus upload the contents to a high performance content server sometime later. The SECON router on the bus will upload this content

when it has high bandwidth e.g. when it returns back to the bus terminal. Since the content he publishes is encrypted and self verifying, the high performance content server will accept the content forwarded by the bus.

To preserve battery life, our subscription service should be able to differentiate contents and content list. The subscription service also should allow a client to specify a content selector, a maximum number of retrieved data items, and an expiration time. Normally, a user subscribes to a certain content list. Once his device discovers a new content, it then retrieves the new content. In many cases, we are very certain of what we want, and we prefer all new contents which satisfy a certain selector to be pushed directly to our devices. Some examples are: Windows updates service, CCN news, friends' posts on Facebook etc. To accommodate these needs, our system is designed to allow clients to subscribe directly to contents and not just a content list, and also allow clients to specify a content selector to filter out any contents that they do not desire.

To facilitate the content publishing, we introduce a new content publish announcement (CPA) message. Before a SECON node starts to publish contents, it's required to announce what it will publish. The publish announcement message includes a name prefix and a duration limit. The message should be encrypted and self-verifying. The announcement will be propagated among peer routers so that any peer router would know how to route a matching user interest back to the publisher.

The SECON border routers are responsible for CPA aggregation. The current policy is relatively simple: depending on the premium paid by each publisher, the SECON border routers will preserve more or less information about a publisher. Let's look at an example:

secon://wildfire.gov/ca/south/santaana; keywords:wildfire,santaana

secon://wildfire.gov/ca/south/longbeach; keywords: wildfire,longbeach

secon://wildfire.gov/ca/north/sanjose; keywords: wildfire,sanjose

The border router may aggregate these three announcements as follows:

secon://wildfire.gov/ca; keywords:wildfire,santaana,longbeach,sanjose

If the publisher pays lower price, the border router may just forward the announcement as:

secon://wildfire.gov

Our name structure is explicit, the name prefix can easily be hashed for efficient lookup.

3.2 Intentional naming design

Our Intentional naming feature is based on our previous work "*lugrain"*[7] the Lehigh University implementation of the late-binding router which supports intentional name message delivery. The specifications on intentional naming are available in an IETF Internet-Draft [11] and a Milcom 2008 paper by BBN researchers[6]. We will focus on the specifics of lu-grain. The major feature of the Lugrain routing scheme is that our router first routes the message toward the final destination using as few network resources as possible; once the message reaches the target region it will be flooded within that small area.

3.2.1 Name knowledge Base and Dissemination

According to [6], each node maintains a knowledge base (KB) of ontology and attribute information. A KB can contain historical, current, and future information about bindings of intentional name attributes to locally registered application end point identifiers(EIDs) or remote EIDs. The knowledge base can range from a simple look-up table that maps intentional names to canonical EIDs of nodes capable of resolving names in that ontology, to a powerful deductive database engine (such as Prolog) that can infer/derive complex derived attributes by execution of rules on base facts.

In our case, our ontology information has limited forwarding scope (hop-count-bounded) which means the knowledge base only has information about nearby nodes. The benefit of this limitation is that it reduces the size of the knowledge base. In our intentional naming system, the geographic location will be used very frequently, so adding an index for it will reduce the name resolution time. If the network is not very dense, knowledge base can be a list of EIDs and ontology information pairs. Due to our intentional naming feature, routers having only a local view of the network will still work very well in real world routing experiments.

Currently, we didn't implement a deductive database that could enable complex rules. Due to the complexity of managing deductive rules, this feature will be included in future.

3.2.2 Intentional Name Syntax

Names can be canonical such as "secon://lehigh.edu" or intentional such as "secon:intent#(role(student))"

In our design, the intentional name must conform to the following specification:

secon:intent#list-of-predicates|(list-of-predicates)+

where the list of predicates is comma-separated. They are a conjunction (AND) of logical terms. An example of how a predicate looks like is as follows:

secon:intent#location(40.594533,-75.376339,10),!role(CRS)

It means all Content Resolution Servers(CRS) located within the10 miles of Lehigh University mountain top campus.

There would be multiple list-of-predicates. The parentheses which separate multiple list-of-predicates will be evaluated as disjunctions operators, while the commas which separate predicates will be evaluated as conjunction operators.

```
...
expr1     : (term|(LPAREN term RPAREN)+) -> (term)+;
term      : term1 -> ^(TERM term1);
term1     : predicate(COMMA predicate)* -> (predicate)+;
predicate : predicate1 -> ^(PRED predicate1);
predicate1 : placeholder ID LPAREN (|factor(COMMA factor)*) RPAREN -> placeholder ID ^(PARA
(factor)*);
placeholder : (NOT)? -> ^(PLACEHOLDER (NOT)?);
factor    : (NUMBER|ID) ;
...
```

Table 1 intentional name syntax

Table 1 is from our ANTLR definition. ANTLR (ANother Tool for Language Recognition) is a tool that is used in the construction of formal language software tools such as translators, compilers, recognizers and, static/dynamic program analyzers. The ANTLR tools can generate java lexer and parser source code based on the grammar we provide. To better understand Table 1, please refer to ANTLR Cheat Sheet[15]

A predicate is specified as follows:

predicate= placeholder ID (|factor(COMMA factor));*

The placeholder is empty or "!". The exclamation point allows the expression of negation. A factor is a parameter passed to this predicate. It can be a sequence of any numbers or characters.

The intentional name in P. Basu et al [6] paper only supports Horn clause which is a conjunction (AND) of logical terms such as function1(v1),function2(v2),function3(v3). Our design allows disjunction and negation without complex syntax. Our syntax is a superset of the Horn clause and compatible with GRAIN[6] syntax. If we only allow one list-of-predicates and do not use negation, the syntax reduces to a Horn clause syntax.

3.2.3 Intentional Name Convention

The predicates are evaluated with the AND logical operator. The first convention is that the geographic predicate must be specified first. The motivation is that geographic information is easy to process, by specifying a geographic predicate at the beginning can filter out most of the unrelated routing paths. Our intentional name forwarding feature currently supports both geographic forwarding and flooding. At the geographic forwarding phase, an intermediate node only has to check the geographic predicate, so placing geographic predicate at the beginning of a *list-of-predicates* helps to improve system performance.

The second convention is that we currently do not support any negation for the geographic predicates. For efficient routing, we want our geographic predicates as specific as possible. Negation of geographic predicate will not give us a specific location, so we disable it by convention. However, the negation of other predicates is allowed. For example, you can use "!role(CRS)" to filter out content resolution servers.

Currently, we provide two build-in geographic predicates, they are

location(latitude, longitude, radius)

all(max_hop)

For consistency, intentional name will not change during forwarding or flooding. So intermediate routers will not modify the max_hop value. At each hop, the intentional name resolver uses the TTL(time to live) value of a SECON message to detect whether this message has reached the maximum hop or not.

We also have a non-geographic predicate:

role(node_role)

node_role can be CRS(content resolution server), normal, or other application specific roles. We allow multiple roles, so a node can be a CRS as well as a content publisher.

There is no limit on how to define a new predicate as long as it conforms to our syntax and conventions.

3.2.4 Intentional Name Resolution

Intentional name resolution occurs when a node tries to send a UI message to not yet determined locations. Figure 2 shows how intentional name resolution works.

The convergence layers are the interfaces to the higher protocol layers. They contain adaptation functions both, for data transport and control purposes. Currently, our convergence layers protocol only support TCP and UDP as transport protocols. We may investigate reliable broadcast protocols such as Pragmatic General Multicast (PGM)[16][17] in the near future. While TCP uses ACKs to acknowledge groups of packets sent (something that would be uneconomical for multicast traffic), PGM uses the concept of Negative Acknowledgements (NAKs). A NAK is sent unicast back to the host via a defined network-layer hop-by-hop procedure whenever there is a detection of data loss of a specific sequence.

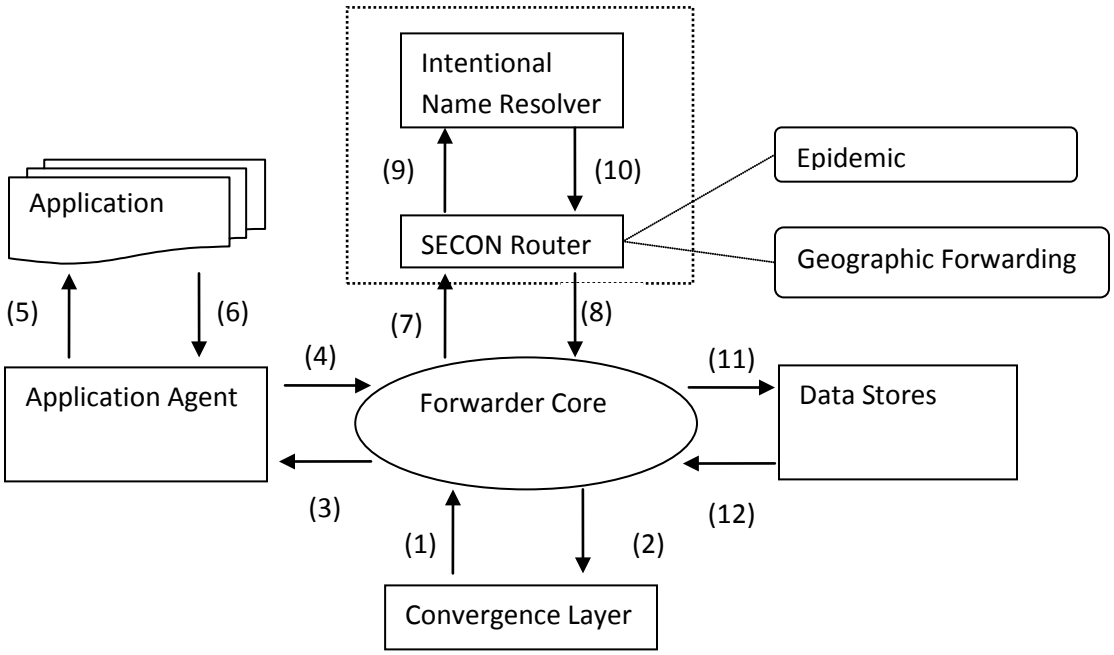

Figure 2 intentional name resolution data flow

The Forwarder Core is the heart of all components. All transmission and reception of data goes through the Forwarder Core. The Application Agent handles the interaction with the applications. The SECON Router will consult the Intentional Name Resolver and provide the route results for Forwarder Core. The routing result could be Epidemic or Geographic Forwarding. The Data Stores caches all incoming and outgoing contents until they expire or when the cache is full.

Figure 2 shows the two data flow paths, one for sending and one for receiving. The sending data flow is started from an Application. The application sends out a message (6) to the Application Agent. The Application Agent handles this message (4) to the Forward Core. The Forward Core uses the SECON Router to resolve the route(7) (8). If the address is intentional, then the SECON Router will consult (9) (10) the Intentional Name Resolver. The receiving data flow begins with the Convergence Layer(1). When the message reaches the Forwarder Core, it will be handled like the a message sent from the Application Agent. When the SECON Router sends the route result back (8) to the Forwarder Core, the Forwarder Core will forward this message either to the Application Agent or to the Convergence Layer. If it's a data content, Forwarder Core may insert it to the Data Stores.

The Intentional Name Resolver will resolve each group of predicate list independently, and merge the resolved result into a final result. For each predicate list, The Intentional Name Resolver will first locate the geographic predicate such as "location" or "all". If it's a "location" predicate and current node is outside the target region, then the INR (Intentional Name Resolver) will resolve to the closest neighbor that can reach that region. If it's a "location" predicate and current node is inside the target

region, then the INR will flood this message to all qualifying neighbors in that region.

If the first predicate is a "all" predicate and the maximum hop is smaller than the time to live (TTL) value, we first initiate our result set to all neighbors and compute a multi-point relays points set. The second step is that for each neighbor if the predicate list evaluate to FALSE and this neighbor is not in the multi-point relays points set, then this neighbor will be removed from the result set.

If a predicate list evaluates to TRUE on the current node, than this message will be forwarded to application layer.

Combining the result sets from each group of predicate list, we obtain the final result set. The convergence layer will forward this message based on the final result set.

3.3 Content Centric Security

In content centric networks, content may be cached in the network, so the security solution should allow data owners to share encrypted published data items with others without knowing a priori who the interested users may be. We note that traditional security solutions [20-22] that require constant access to an Authentication Authorization Accounting server (AAA) server or certificate authorities do not apply in intermittently connected network environments. Thus, effective security solutions that can work in intermittently connected network environments do not exist. We argue that a data centric security solution is desirable for future mobile applications since it allows data owners to have full control over who can view their published data.

In [23,24], the authors have proposed using cipher-text attribute-based encryption

approach to provide such a security feature. Each publisher creates unique attributes for different categories of data items they publish e.g. appliances, clothing, sport-equipment, lawn care etc. Data items that belong to more than one category, e.g. lawn mower belongs to both “lawn care” and “appliances” categories, will be encrypted such that any user who is interested in either “appliances” or “lawn care” will be able to retrieve such data items from the storage nodes. Details of the enhanced CP-ABE design which includes new features such as supporting dynamic attributes, negative attributes and user revocation can be found in [25]. The dynamic attributes will be useful for region-based content distribution e.g. certain encrypted data items (e.g. store coupons) can only be accessed by subscribers that are currently in a certain geographical region. Future enhancement of the design described in [25] include using key delegation approach to make user revocation more scalable.

3.4 SECON Detail Design

3.4.1 SECON Architecture Overview

Content Resolution Servers(CRS) support content caching, intentional name resolution & forwarding. It also supports some additional features: translations of meta-data descriptions from different data schemas, and additional intentional name resolutions such as mapping of some geographically based intentional names into a list of endpoint identifiers. Content Resolution Servers usually have large storage capacity and provide services to broader users.

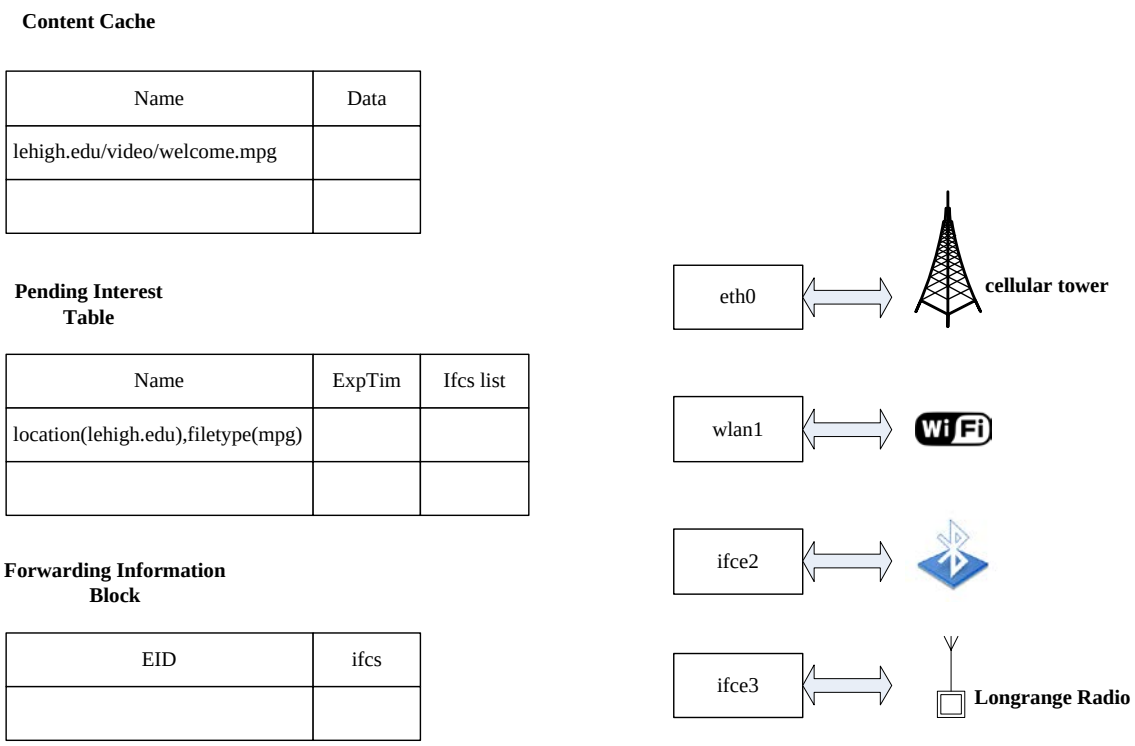

Figure 3 Major tables of a SECON Router

Figure 3 shows the major tables of a SECON router. A SECON node can make use of multiple network interfaces e.g. 3G, WLAN, long range radio or Bluetooth. For efficiency, a SECON node is highly adaptable such that information discovery can be done via 3G interface but content transfer is done via the WLAN interface automatically. A SECON node may enter into a peer-to-peer mode if necessary. In each SECON node, the Pending Interest Table (PIT) maintains all the none expired user interest(UI). The Forwarding Information Block (FIB) maintains forwarding entries which come from the Content Publish Announcements(CPA) of CRS. A Content Cache(CC) is available to each SECON node. In addition, the content resolution server(CRS) has a permanent storage(PS) and may offload the storage request to peer CRS.

To support basic features, SECON use three major types of messages: (a) Content Publish Announcement(CPA) messages, (b) Content Data (CD) messages and (c) User Interest messages. In subsequent paragraphs, we describe these three major types of messages that are exchanged between SECON nodes and what they are used for.

3.4.2 SECON Message Headers

All three types of messages have a common message header as shown in Table 2.

version	The version of SECON protocol
src	Source Node Identifier
dest	Destination Address(Can be specific address or intentional address)
ttl	Time to live value

Table 2 common message header

There are four fields in the common message header, namely(1) version, (2) source endpoint identifier (src), (3) destination end point identifier (dest) and (4) time to live value. The dest field contains information about where the messages are intended to be sent. For example, to send a message to all content resolution servers within 1 mile radius of Lehigh university, the destination field can be written as “secon:intent#location(Lehigh University, 1 mile),role(CRS)".

For intentional names, intermediate routers that support progressive name resolution will forward the messages typically using geographical forwarding first until it reaches the target region. Then, the first router within the target region that receives this

message will then use flooding to disseminate this message to all content resolution servers within the target region.

The TTL field is important especially when our destination field can be a broadcasting address. For example, to limit the scope of broadcast messages, we can set the TTL to 3 which means only multicast routers that are within 3 hops away from the sender will re-broadcast this message. For unicast messages destined to unique identifiers, TTL can be used to limit the number of forwarding hops. This feature is useful in mobile ad hoc networks where the sender basically wants to limit the total number of transmissions in the system by indicating that the messages can be dropped after going through TTL=x hops and still cannot reach the destination.

3.4.3 Content Publish Announcement(CPA) Message

type	Type one, type two or reject
pn	publisher name
uri	URI prefix the publisher intend to publish
cmd	additional attributes associated with the URI prefix.
duration	
signature	
nonce	

Table 3 Content Publish Announcement message

There are two types of content publish announcement (CPA) messages. Type one CPA is sent by any data owner who wishes to publish contents before it starts sending

content data (CD) messages. Type two CPA is sent by CRS to inform nearby routers that new data content can be found at this CRS. The difference between the two types of CPA is that only type two CPA message will be disseminated by neighboring SECON nodes. The CPA contains (a) the Publisher Name, (b) the Uri field is the URI prefix that the publisher intends to publish, (c)Content Meta-Descriptor(CMD) contains additional attributes associated with the URI prefix e.g. politics, weathers, finance. The CPA message is signed for security reason, and a nonce is provided to prevent replay attacks.

When intermediate nodes receive such CPA messages, they create new or update existing entries in their forwarding information block (FIB). They forward these CPA messages to the destinations specified in the messages using the underlying routing protocols.

Depending on CRS's policy and authentication information in the CPA message, CRS can either accept or reject this CPA request. If CRS rejects the CPA request, CRS will not broadcast the CPA to nearby routers and will send back a reject response to the publisher. By default, CRS will accept all CPA requests, and give some low priority publisher temporary ability to publish.

3.4.4 Content Data (CD) Message

type	list or content
uri	This field should be unique
cmd	additional attributes associated with the uri field.
pn	publisher name

duration	
signature	
nonce	

Table 4 Content Data message

After sending a CPA message, the publisher can start publishing content to the desired CRS if no reject response is received. Intermediate nodes can refuse to accept CD messages if the publisher does not send the corresponding CPA message first.

Each data item can be described by a unique resource identifier (URI) which matches a prior Content Publish Announcement message. Each data item is described by some meta-data descriptors (CMDs) e.g. keywords: weather, sport, entertainment, movie, and a URI-like content identifier. The meta-data descriptors can be used for matching with cached user interest messages to be described in next section.

3.4.5 User Interest (UI) Message

Just like CPA, the UI message also contains the publisher name. The UI message can be in plaintext or encrypted. Each UI message is signed and a nonce is included to prevent replay attacks.

type	directory listing/content request/cancel previous UI
uri	Interested prefix or the URI for a unique content data
cmd	additional attributes associated with the uri field.
selector	optional

ttl	optional
duration	
signature	
nonce	

Table 5 User Interest Message

A user can express his interests for certain types of contents by sending a user interest(UI) message. The content meta data(CMD) descriptors include (i) keywords of data items that are of interest to this particular user e.g. movie, weather, sports, music etc. The selector field contains several sub-fields: (a) any treatment preference for the specified keywords in the meta-data descriptors e.g. highest preference first, conjunctions etc, (b) specific publishers that are of interest to this user, (c) the scope of the advertisements i.e. how many hops a user wishes his/her interests to be forwarded by others.

Our user interest(UI) is very different from the interest message in CCNx [5]. The interest message in CCNx is transient, which means either the requested content can be returned within several seconds or this interest message will be discarded. The maximum timeout value for CCNx is 10 seconds for normal network operations. Our user interest message will last for a long time e.g. several days or even several months. Thus, you can think of the interest message in CCNx as a request event while our user interest is a real interest which will exist for a longer duration such that the node that sends this interest message can receive items that are published after the interest message is sent but before this interest expires. The CCNx protocol does not assume that the underlying transport of messages is reliable. To provide reliable delivery, interest messages that are not satisfied

in some reasonable period of time must be retransmitted. On the contrary, our control messages such as user interest messages are sent using reliable transmission mechanisms like TCP.

We support two types of User Interest messages: Type 1 requests an upstream router or CRS to deliver a matched content list (MCL) without sending matching data contents; Type 2 requests upstream router or CRS to deliver matched data contents.

When an intermediate SECON node, let say Node A, receives a UI message from a subscriber, it will search through its local content data storage to see if it carries some data items that match this user interest message. Matching can be done via longest prefix match lookup in the data store or via keywords match. If there are matched data items and the received UI is a Type 1 UI message, then node A will send back a matched content list (MCL). The matched content list will contain descriptors for the top K matched data items. The subscriber can decide whether it wants to retrieve all or some of these data items. A new user interest will be sent out if the subscriber wants more result. A cursor is placed in the selector field of a new user interest message so that Node A will know where to start the matched content list again. Otherwise, node A will send back matching content data items for Type 2 UI messages. Node A will insert this UI message into its Pending Interest Table (PIT) if its duration has not expired, and its TTL has not yet been exceeded.

If the destination address of this UI message is not intentional, node A will look up its FIB table using the longest prefix match, and relay this UI message to the matching

interface. If the destination address of this UI message is intentional, then we use the resolution method discussed in Section 3.1.4.

An example of when an UI message is sent and how it is processed is described as follows: a professor from Lehigh University may visit Montreal to attend a technical conference. She may have a half-day break when she can tour the city. She can send a UI message to find useful information that can help her plan her tour. An example of her UI message is like this:

secon:intent#location(Montreal Hotel B,1000m),role(top-CRS),interest(travel, event)

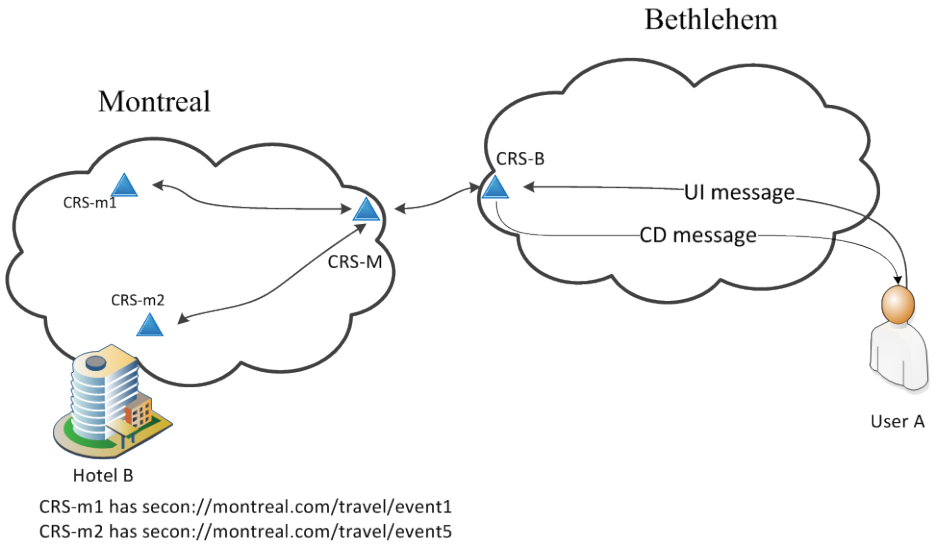

Figure 4 User Interest message with Intentional Named Destination

This message will be sent by intermediate SECON nodes towards a top-level CRS in Montreal that takes care of the travel category. The top-level CRS caches this UI message, and then disseminates this message to all content resolution servers that are

located within 1000m of that Montreal hotel B. The content resolution servers in that target region each generates a content response. This top-level CRS then consolidates all responses and sends an aggregated response to that professor. In Figure 4, we show how the UI and the CPA messages are exchanged between the various nodes. The device from Professor A sends the UI message (UI type=list) as shown. This message is forwarded using geographical forwarding until it reaches CRS-M which is in Montreal. CRS-M disseminates this UI to CRS-m1 and CRS-m2 that are within the target region. CRS-m1 and CRS-m2 each responds with its own CD message (CD type=list). CRS-M aggregates these two messages and sends a merged CD message (CD type=list) to Professor A. Professor A can then send a new UI message that specifically asks for the data item related to event1. Note that to minimize the round trip retrieval time, an additional field can be provided in the UI message to retrieve all contents if the total matched items are less than K such that the professor needs not send another UI message to retrieve a specific item.

Chapter 4

Prototype & Experiment

In this chapter, we first describe our prototype. After that, we describe the experiment we conducted on Emulab[29] and ORBIT[30,31].

4.1 SECON Prototype

Traditionally, testbeds are expensive to build. They are difficult to reconfigure and share, and have limited flexibility. In addition, some networking phenomena such as wireless radio interference can be very difficult to reproduce experimentally, thus making it difficult to compare or evaluate protocol designs. As a result, simulation oriented network research is prevalent in recent years. However, simulation is not perfect, and it will miss some important details that can be captured in real testbeds and real networks.

Thanks to the recent developments in future internet researches and funding from National Science Foundation & DARPA, several easy-to-use and freely available testbeds are available e.g., ORBIT[30,31], Emulab[29]. Thus, we decide to develop a prototype instead of simulation for our research, and test it on real network testbeds such as Orbit and Emulab.

We choose Java as our prototype language. We have a very clear server client separation. Only one server daemon can run on one machine, but multiple clients (GUI application) can run simultaneously without any problem. The sample GUIs are shown in Figure 5 and Figure 6. The CPA announcement, UI message, Data content and the

neighbors of the local daemon will be shown in four independent tabs in the top area(as shown in Figure 5). The bottom area is used for a user to choose what he wants to send, e.g. CPA/UI/CD messages. To check the received data content, the users just have to double click on that received data item entry, another sample GUI (as shown in Figure 6) that displays that data item will show up.

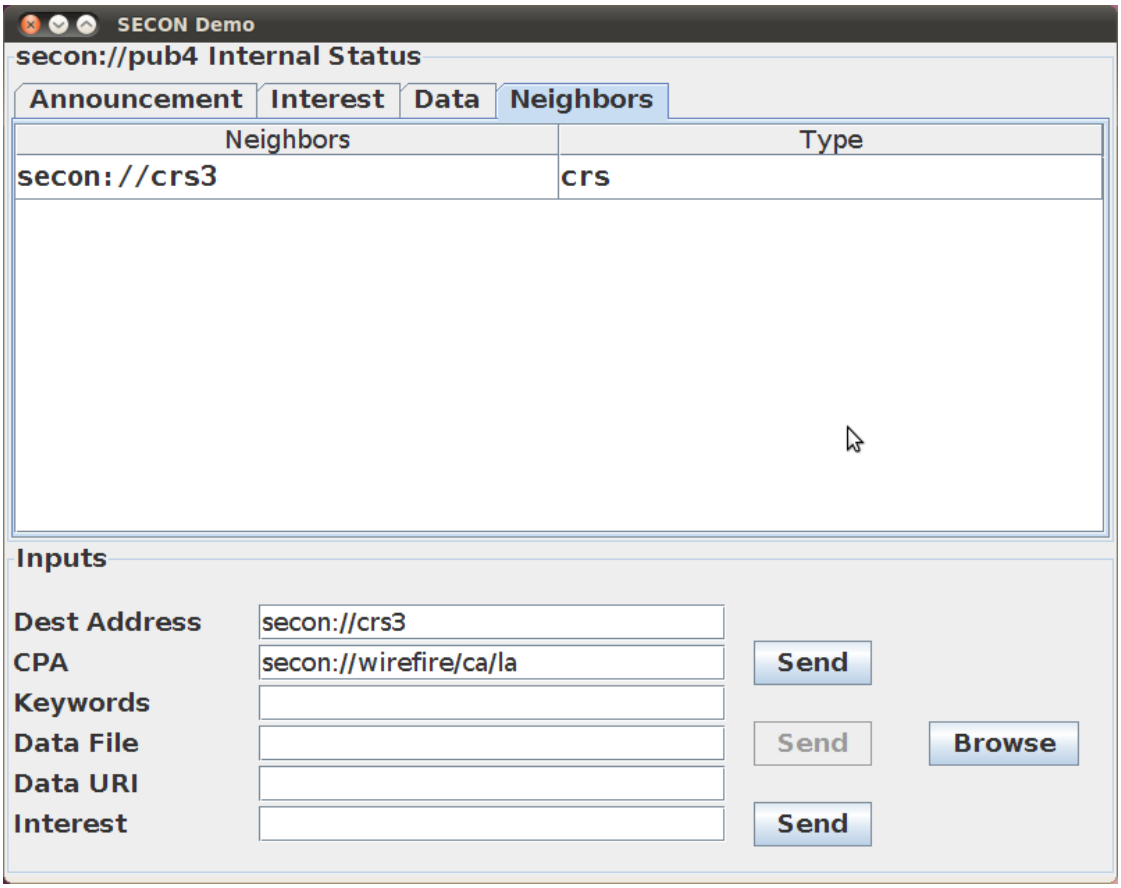

Figure 5 user interface (Neighbor view)

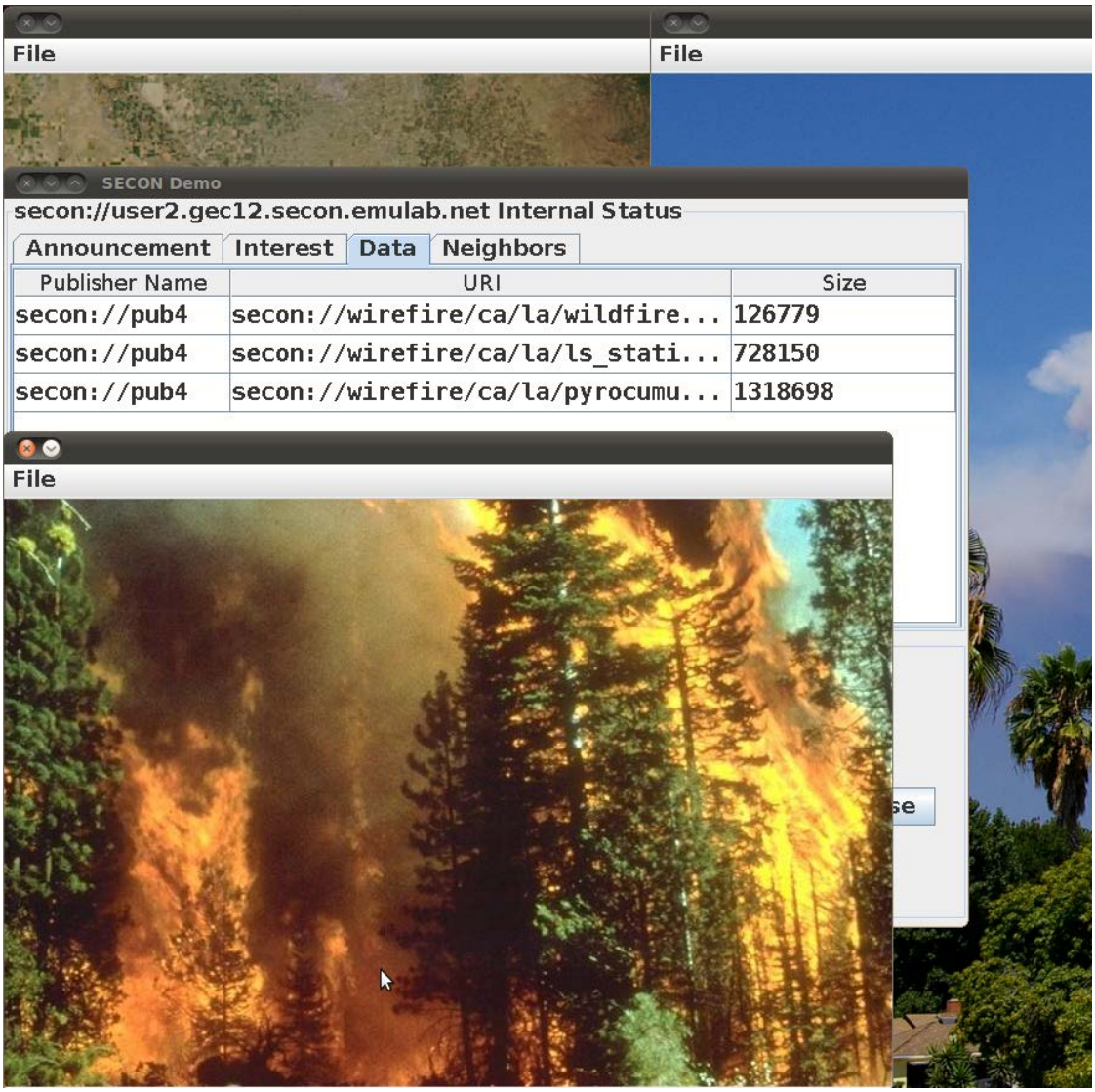

Figure 6 user interface (Data Content view)

Because of our publish/subscribe design in content centric networks, we decide not to build our prototype on top of existing software such as IETF DTN implementation, and CCNx. The IETF DTN implementation is known to be slow while preliminary implementation of CCNx is as inefficient. Thus, we develop our prototype from scratch. To support iterative and incremental development, we choose a limited set of core features to develop for our preliminary prototype. These features include

publish/subscribe feature, publish announcement aggregation feature, content keyword feature and intentional name feature. Our experiments conducted in Emulab evaluates all these features. Because the we can specify the experiment resources in ORBIT testbed, the experiment is consistent and repeatable. So we use the ORBIT testbed to measure our prototype performance in wireless ad-hoc network.

4.2 Testbed

In the early stage of Internet, researchers can test and play with their routers without worrying too much about the unforeseen side effects. However, today's Internet is not just used only for university research experiments. Today's Internet is closely knit to our daily life operations, and any misconfiguration would cause significant economic losses. Thus, it is extremely difficult to introduce new technologies and protocols without enough scientific validation via large-scale testing. According to S. Paul et al.[26], PlanetLab [27, 28] was the first effort to produce a testbed that would allow researchers to perform very large scale experiments. It utilizes thousands of Internet nodes offered by research, educational and industrial organizations, spread over multiple countries. These Internet nodes run Linux virtual server software to provide isolated resource allocation to multiple concurrently active experiments using a common control framework. However, the deterministic behaviors cannot be verified due to the PlanetLab's poor repeatability. In contrast, a simulated testing environment is deterministic and repeatable but not able to take into account the scale and diversity of real world. Emulab[29] is an effort to leverage the best of PlanetLab and simulated testing environment. Emulab provides an "emulation" environment for network experimentation. Emulab has the ability to offer simulated links

and nodes within PlanetLab slices. Emulab allows researchers access to realistic experiments without sacrifice fine grained control and repeatability.

ORBIT[30,31] is a two-tier wireless testbed designed to achieve reproducibility of experimentation, while also supporting evaluation of newly designed protocols and applications in real-world settings. The laboratory-based wireless network emulator includes a large two-dimensional grid of 400 802.11 radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models. Once the basic protocol or application concepts have been validated on the lab emulator platform, users can migrate their experiments to the field trial network which provides a configurable mix of both WiMAX and 802.11 wireless access in a real-world settings.

Although Emulab also provides 802.11 radio nodes, Emulab's 802.11a/b/g testbed is deployed on multiple floors of an office building. ORBIT provides a laboratory based wireless network emulator for an initial, reproducible testing environment, and Real-world testbed environment of wireless nodes (a mixture of WiMAX and 802.11 wireless access) for field validation of experiments.

4.3 Experiment in Emulab

The topology of the experiment conducted in Emulab is shown in Figure 7. In this experiment, we deployed five SECON nodes in Emulab as well as five SECON nodes deployed in our WiNS Lab. The topology forms an inter-domain scenario which can be used to test many of our features.

To create this topology, we have to do three things: 1, setup the network in Emulab. 2 setup the network in our WiNS Lab. 3, connect the two networks. Setting up the network in Emulab just needs a NS file[35]. Emulab will create several network interfaces according to the topology defined in the NS file(each link corresponds to two virtual network interfaces). To set up the network in WiNS Lab, we create five Linux virtual machines running on a host workstation. Each virtual machine has a public routable IP and a private IPv4 network address such as 192.168.1.12. The public routable IP address lets us control the node remotely. We use "iptables" command to block certain IP addresses to form the topology. To connect the two networks, we use OpenVPN to create a tunnel link between crs2 and crs4.

In this experiment setup, we use a prophet[34] module to maintain neighbor information. We assume each CRS periodically readvertises aggregated CPAs newly received from publishers to its neighboring CRSes.

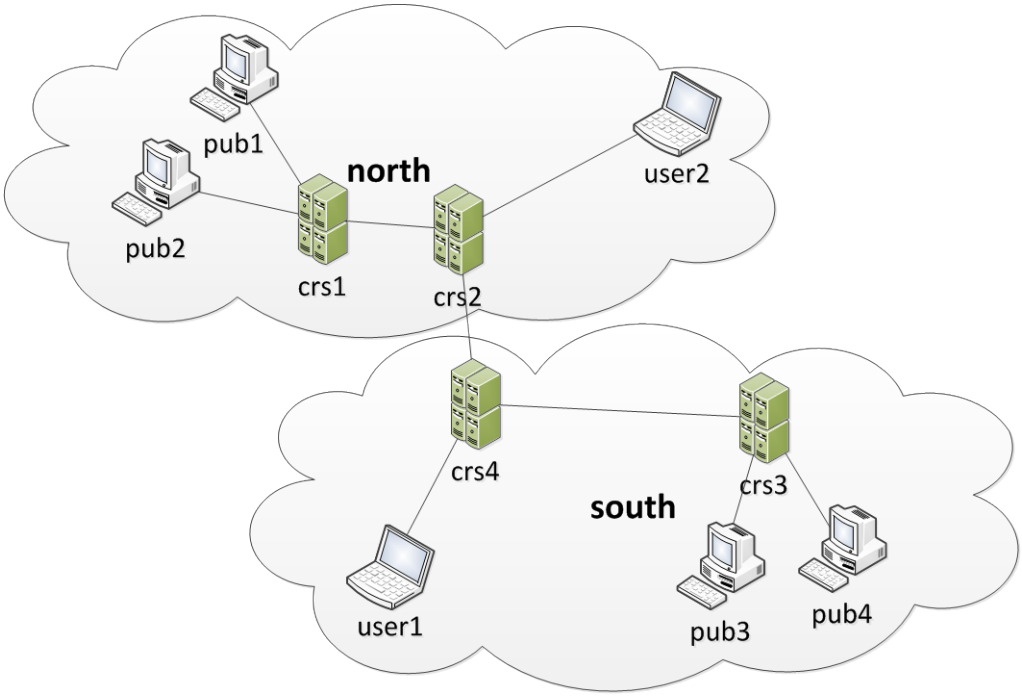

Figure 7 Emulab Inter-domain experiments topology

4.3.1 Keyword feature experiment

In this experiment, we have four publishers. Each of them publishes wildfire information in different areas (See Table 6). We observe that crs1 aggregates the CPA message sent from pub1 and pub2 as " secon://emergency/wildfire/ca/north/" and forwards the new CPA to crs2. crs2 then forwards this CPA message to crs4. crs4 eventually sends this CPA to crs3.

Publisher	CPA
pub1	secon://emergency/wildfire/ca/north/pinecrest
pub2	secon://emergency/wildfire/ca/north/greenville
pub3	secon://emergency/wildfire/ca/south/barstow
pub4	secon://emergency/wildfire/ca/south/goleta

Table 6 Content Publish Announcement list

Each data content they publish will have a metadata descriptor to indicate the fire danger level. The Keetch-Byram Drought Index (KBDI)[33] is used to equate the effects of drought with the potential for wildfires. It measures the dryness of the soil and duff layer. The higher the index the higher the risk of wildfire. Table 7 shows the five possible fire danger levels.

Keywords	KBDI
Low Fire Danger	0-150
Moderate Fire Danger	140-300
High Fire Danger	300-500
Very High Fire Danger	500-700
Extreme Fire Danger	700-800

Table 7 fire danger level

The user1 uses "secon://emergency/wildfire/ca/" as its prefix and "Very High Fire Danger, Extreme Fire Danger" as its keywords. Currently, we only support exact keyword match. We leave semantic-aware and fuzzy keyword search as future work.

We randomly select a danger level and create a data content for each publisher. Our experiment shows that user1 receives data content from all four publishers when the metadata descriptor of the data content has either the keyword "Very High Fire Danger" or

"Extreme Fire Danger". The data content messages without these two keywords will not be sent to user1.

4.3.2 Intentional name feature experiment

To test the intentional name feature, we first assign SECON nodes a set of coordinates:

node id	location	coordinate
pub1	Pinecrest	39.15°N 120.97°W
pub2	Greenville	40.14°N 120.96°W
pub3	Barstow	34.85°N 116.77°
pub4	Goleta	34.44°N 119.82°W
crs1	San Jose	37.33°N 121.9°W
crs2	San Francisco	37.77°N 122.41°W
crs3	Santa Clarita	34.4°N 118.54°W
crs4	Los Angeles	34.05°N 118.25°W
user1	Long Beach	33.76°N 118.19°W
user2	Napa	38.29°N 122.29°W

Table 8 coordinates

In this experiment, we do not let pub1 to pub4 send out CPA messages so the routes of UI messages are only determined by intentional name resolution. The four publishers publish data content to the nearest content resolution server. The user1 sends out an intentional user interest message with address "secon:intent#location(34.24, -118.32,200), role(CRS)" and interest "secon://emergency/wildfire/ca/north"

In this experiment, this UI message reaches both crs1 and crs2. The intermediate CRS(CRS4) will cache this UI message, so that when a content data message flows back, intermediate content resolution servers will not reject this CD message. Our experiment successfully delivers all content data published by pub1 and pub2 to user1.

In conclusion, our experiment on Emulab shows that our prototype is very good at finding a route either by announcing it from the publisher or by sending an intentional user interest message from the content seeker.

4.4 Experiment in ORBIT

4.4.1 publish/subscribe based content delivery performance

This experiment we conducted on ORBIT testbed has the linear topology as shown in Figure 8. Node1, node2 ... node8 correspond to the node1-1, node1-2 ... node1-8 of ORBIT testbed sandbox 4. Each node has an Intel® Core™ i7-875K Processor (8M Cache, 2.93 GHz), an Intel® Centrino® Advanced-N + WiMAX 6250 chip and a 8GiB 1333MHz memory kit.

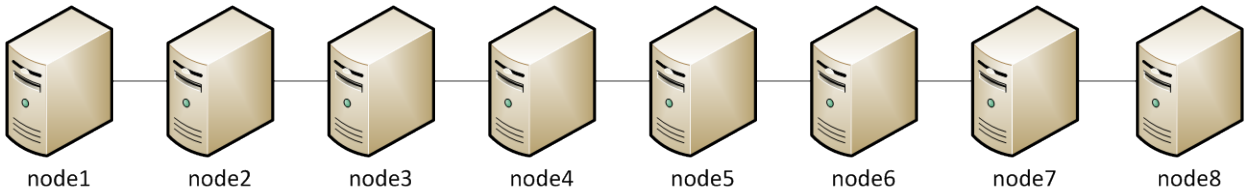

Figure 8 Linear Topology for Testing CPA/UI Messages

The node1 is the publisher, node2 to node8 are subscribers.

The node1 announces CPA message: "secon://node1-1.sb4.orbit-lab.org/clothing/"

This CPA message will be propagated from node1 to node2, from node2 to node3, ..., finally from node6 to node7. This CPA information will be used to update the FIB table. So when each node receives a matching UI message they will know how to forward the UI message back to the publisher. In this experiment setup, we use a prophet[34] module to maintain neighbor information. We use "iptables" command to block certain IP addresses to form the topology.

When node8 receives an UI message "secon://*/clothing/*" from the application, node8 will look up its data storage to see if there is any match. If the UI is not expired, , node8 will forward this UI message to node7. In this way, this UI message will be eventually forwarded back to the publisher node1. When node1 receive this UI message, it will look up its content cache and send the content from "secon://node1-1.sb4.orbit-

lab.org/clothing/1" to "secon://node1-1.sb4.orbit-lab.org/clothing/100" to node2.

When we conduct this experiment, we set the wireless interface to ad-hoc mode and use TCP as the transport protocol. We send three different sizes of data contents (1KB, 16KB, 256KB). For each data content size, 100 packets will be sent out to obtain an accurate average delay and lost measurements. Since we use TCP as our transport protocol, we expect a zero packet lost rate. The packet inter arrival time is set to 10 milliseconds, 100 milliseconds and 1000 milliseconds to observe the network behavior under different load.

Figure 9 is the hop by hop delay measurement for 1KB content data. Node1 is the publisher so the delay is zero. As we can see, the delay increases smoothly hop by hop in this experiment. The average 7 hop delay for the contents delivered from node1 to node8 is about 193.95 ms if the contents are generated at a interval of 10 ms. Recall that to form this topology, we use the "iptables" tool to drop IP packets received from certain IP addresses. However, all 8 nodes are actually sharing the same wireless channel and can interfere with one another. When the packet generation interval is 10 ms, there is much interference among nodes as they attempt to relay packets they generate or receive. The average 7 hop delay is 96.25 ms if the interval is 100 ms while it is 69.78 ms when the interval is 1000 ms. Thus, we see that as the packet generation interval is beyond 100 ms, there is very little interference among the nodes. However, continuously increasing the interval will not decrease the average delay further because the contention among nodes is already removed once the interval is beyond 1000 ms. The maximum throughput for this experiment is 1024*8*7/0.06978=821 Kbps. We only use a single thread to send TCP traffic, so this result is acceptable.

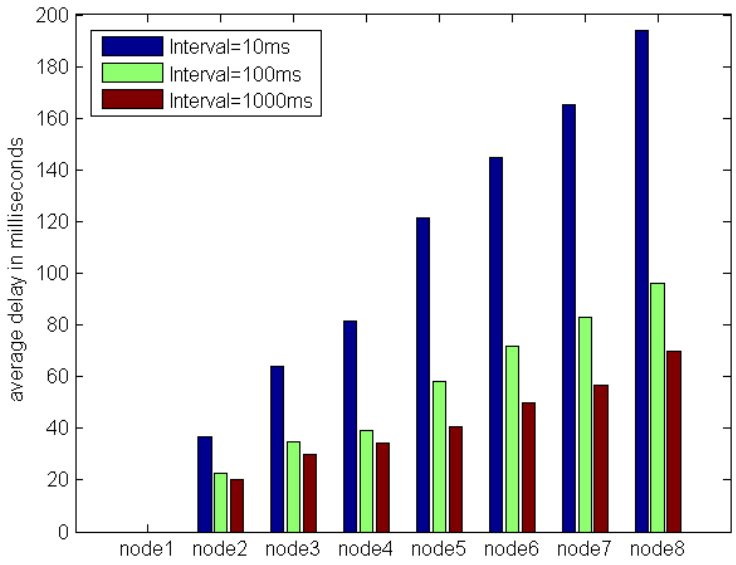

Figure 9 hop by hop delay for 1KB data content

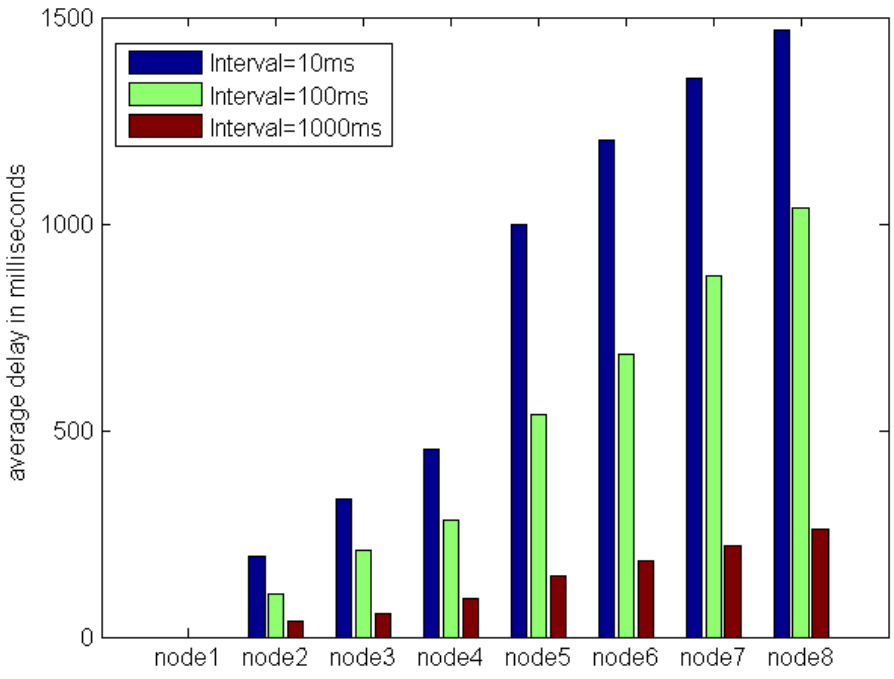

Figure 10 hop by hop delay for 16KB data content

Figure 10 shows the hop by hop delay for 16KB data content, when the interval is 10ms, the 7 hop delay is 1468.71ms. When the interval is 100ms, the 7 hop delay is 1039.46ms. When the interval is 1000ms, the 7 hop delay is 262.66ms. If the channel is not congested, it takes about 32ms (assuming 4Mbps channel) to transmit a 16Kbytes data content so with 7 hops, we will have an end-to-end delay of about 224ms which is quite close to what we observed for the case where the packet generation interval is 1000ms. The maximum throughput for this experiment for the 1000 ms case is 1024*16*8*7/0. 26266= 3.5 Mbps.

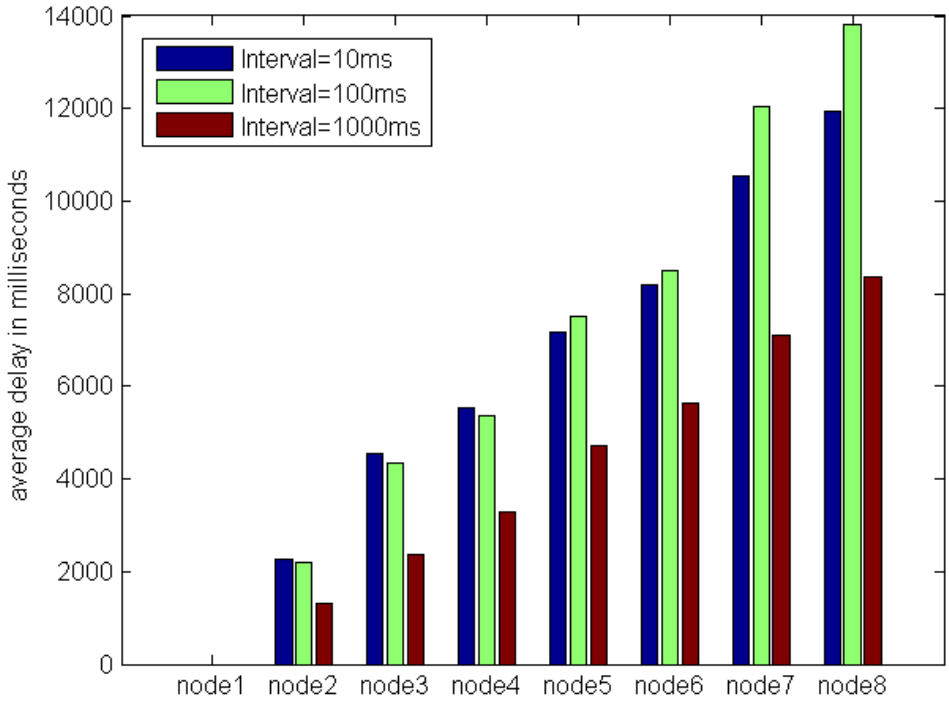

Figure 11 hop by hop delay for 256KB data content

Figure 11 shows the hop by hop delay for 256KB data content, when the interval is 10ms, the 7 hop delay is 11953.47ms. When the interval is 100ms, the 7 hop delay is

13823.52ms. When the interval is 1000ms, the 7 hop delay is 8348.61ms. To reduce the interference, we probably have to increase the interval to 2500ms or more. The maximum throughput for this experiment is 1024*256*8*7/8.348= 1.758 Mbps.

In this experiment, we analyze the effectiveness of publish/subscribe based content delivery performance. The results shows that our prototype forwards the data content smoothly.

4.4.2 Publish announcement aggregation and its performance

Figure 12 is a 8-node testbed we deployed in ORBIT recently. Publisher 1 & 2 send content data packets to CRS1. Publishers 3 & 4 send such packets to CRS2. Subscriber S1 sends a UI to CRS3 which forwards it to both CRS1 & CRS2. Figure 13 plots the average latency of content data delivered to S1.

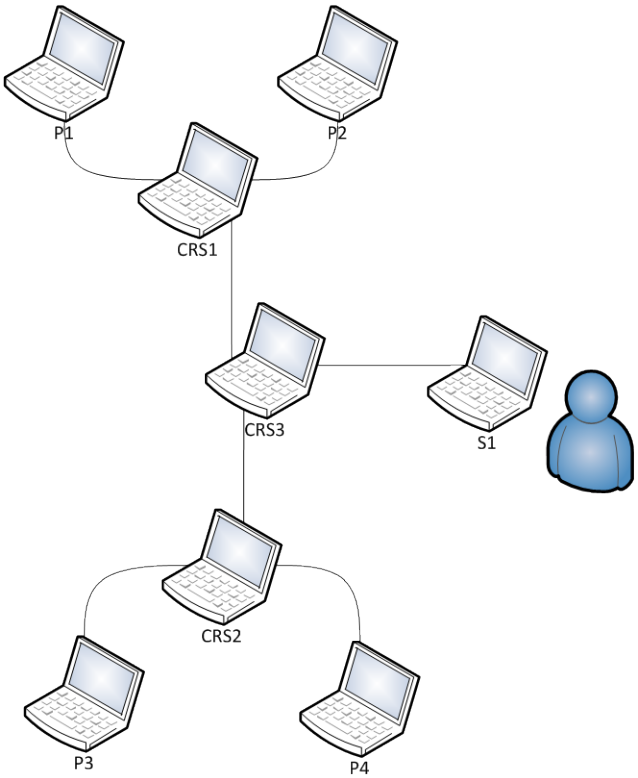

Figure 12 CPA Announcement Experimental Setup

In this experiment, p1 p2 p3 and p4 publish contents according to Table 6. S1 subscribes to "secon://emergency/wildfire/ca/north". So the data content will be forwarded through CRS1, CRS3 to S1.

Figure 13 shows the delay on CRS1, CRS3 and S1. We can see that the delivery delay from P1 at CRS1 is lower than the delay from P2. That is because P1 starts sending content first. When P2 starts sending contents, the wireless link is already congested. The delay at CRS3 and S1 is significantly larger than the delay at CRS1. That is because CRS1 is the first hop which has less wireless interference.

In our experiment, the packet loss rate is 0%. Only contents published by P1&P2 are transferred to S1. The publish announcement aggregation feature works very well.

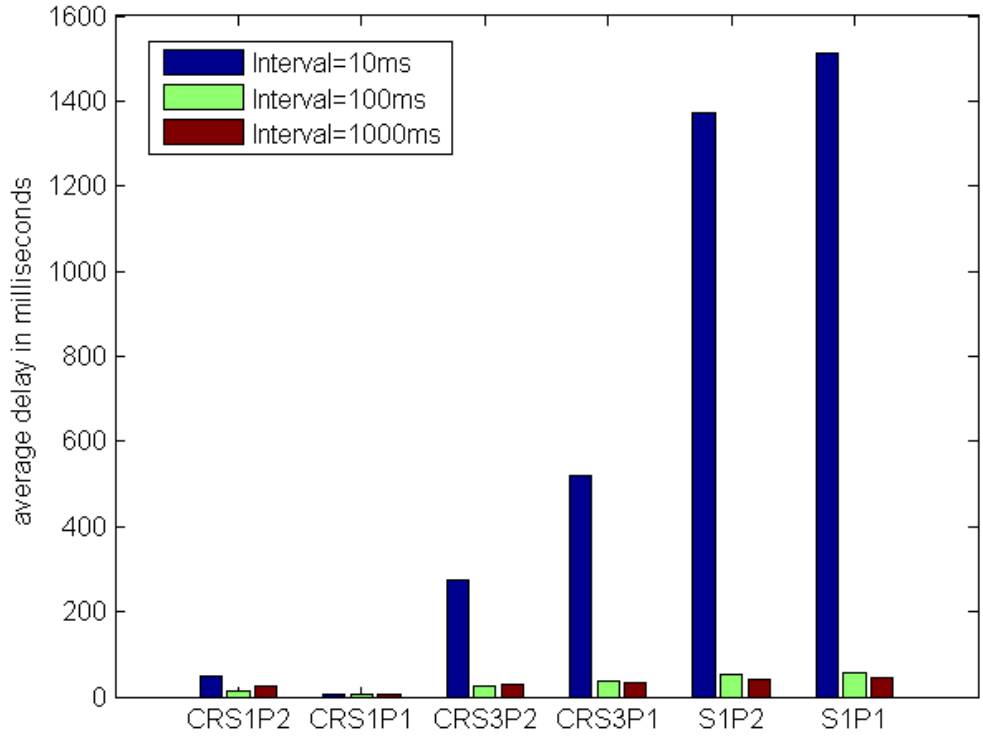

Figure 13 delay on CRS1 CRS3 and S1

4.4.3 Impact of Mobility

One good feature of content centric network is that when a packet is lost, you don't have to request it from the original publisher, intermediate nodes have a copy of that packet. So the lost packets can be retransmitted very quickly.

This experiment we use the topology of Figure 8. The difference is that we only use 1KB content and use 200ms, 400ms as the content generation intervals. We let the link between node7 & node8 has a loss rate of 40%. We observe that even at such a high packet loss rate, the average delay to node8 is not affected much. The reason is that each content centric network node has a content cache. All contents are cached if the storage is not full. So when node8 lost the packet, it can request it from node7 instead of node1.

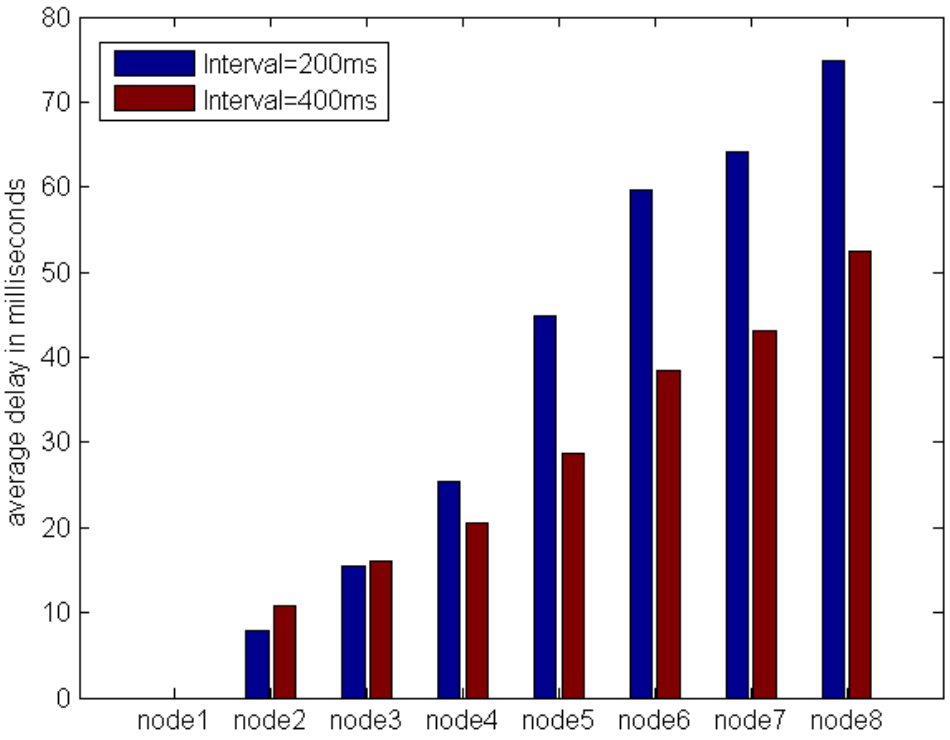

Figure 14 the link between node7 & node8 has a lost rate of 40%

Chapter 5

Conclusion

In this thesis, we developed our content centric mobile network which features publish announcement aggregation, intentional name, publish/subscribe based content delivery, and content keywords search. Some of these features have been discussed by other researchers in different contexts. We argue that these features are closely related and serve the same goal: to provide flexibility to access contents in highly dynamic environments. Due to time constraint, we have implemented only the basic functions of each feature. However, even the basic functions demonstrate the potentials of our content centric mobile network design. In SECON, each route can be determined either by following the path taken by content publish announcements or by having the user send an intentional name based user interest message, which means our design will work even when the network does not provide good connectivity and content publish announcements cannot be propagated through the network successfully. Our publish/subscribe based content delivery not only delivers new contents fast, it also allows mobile devices to preload contents to a specific server in order to preserve battery life and achieve better delivery performance. Content keywords search feature is only one example of what our content metadata descriptor can provide. Future enhancements on content metadata descriptors will give us richer expressiveness, so that content discovery will be much more accurate and efficient.

We conduct our experiments on two real testbeds: Emulab and ORBIT. The Emulab testbed is very suitable for testing our various features, while ORBIT provides

dedicated wireless testbed resources for more repeatable experiments. Our experiences show that creating an artificial wireless network topology may introduce undesirable interferences.

During our prototyping process, we discover some enhancements which could potentially improve the efficiency of our design. We currently run our protocol on top of a TCP stack. That means we do not make use of the broadcasting nature of wireless network. The Pragmatic General Multicast (PGM)[16][17] should be adopted as an alternative transport protocol when a message needs to reach many neighboring receivers. PGM uses the concept of Negative Acknowledgements (NAKs) to acknowledge groups of packets. A unicast NAK is sent back to the host whenever there is a data loss detection. Another possible enhancement would be multi-point relays (MPR) [18][19]. Multipoint relaying is a technique to reduce the number of redundant re-transmissions while broadcasting. The key idea is that each node selects a subset of its neighbors (relay points) which connect to all two-hop neighbors. MPR has been implemented successfully in the Optimized Link State Routing(OLSR) protocol[19]. Since our intentional name resolution includes a geographic forwarding phase and an epidemic flooding phase, adopting MPR should considerably improve the flooding efficiency.

Reference

---

[1] M. Chuah, X. Xiong, “Secure Content Centric Mobile Network”, to appear at IEEE Globecom 2011

[2] K. Fall, “A delay tolerant network architecture for challenged networks”, Proceedings of ACM Sigcomm, 2003.

[3] V. Cerf et al, “Delay Tolerant Networking Architecture”, RFC4838, April, 2007.

[4] https://www.ccnx.org/releases/latest/doc/technical/NameConventions.html

[5] V. Jacobson et al, “Networking Named Content”, Proceedings of ACM CONEXT, Dec, 2009.

[6] P. Basu et al, “Persistent Delivery with Deferred Binding to Descriptively Named Destinations”, Proceedings of IEEE Milcom, 2008

[7] I. Stoev, M. Chuah, B. Herbst, "Lehigh Intentional-Naming Router Implementation Report"

Lehigh University, April 30th, 2010

[8] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and Beyond) Network Architecture. In SIGCOMM, 2007.

[9] V. Jacobson, “Content Centric Networking,” Presentation at DARPA Assurable

Global Networking, January 30, 2007.

[10] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., and J. Lilley, “The design and implementation of an intentional naming system,” Proc. ACM SOSP, December 1999.

[11] http://tools.ietf.org/html/draft-pbasu-dtnrg-naming-00

[12] SUNY Stony Brook, “XSB Logic Programming and Deductive Data Base System,” URL – http://xsb.sourceforge.net

[13] http://en.wikipedia.org/wiki/Apple_Push_Notification_Service

[14] http://tools.ietf.org/html/rfc3305

[15] http://www.antlr.org/wiki/display/ANTLR3/ANTLR+Cheat+Sheet

[16] PGM Reliable Transport Protocol Specification: http://tools.ietf.org/html/rfc3208

[17] http://code.google.com/p/openpgm/

[18] A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks”, Proceeding of HICSS, pages 3866 – 3875, 2002

[19] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlehaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocal(OLSR)," RFC 3626, Oct. 2003, Network Working Group.

[20] B. Anton, B. Bullock, J. Short, “Best Current Practices for Wireless Internet Service Provider (WISP) Roaming”, Wi-Fi Alliance, Feb, 2003

[21] J. Leu et al, “Running Cellular/PWLAN services: practical considerations for Cellular/PWLAN architecture supporting interoperator roaming”, IEEE Communications Magazine, Vol 44, No 2, July pp 73-84

[22] M. Shi et al, “A service agent based roaming architecture for WLAN/Cellular Integrated Networks”, IEEE Transactions on Vehicular Technology, Vol 56, No 4, July, 2007 pp 3168-3181

[23] V. Goyal et al, “Attribute-Based Encryption for fine-grained access control of encrypted data”, Proceedings of ACM CCS, pp 89-98, 2006

[24] J. Bethencourt, A. Sahai, B. Waters, “Ciphertext-policy attribute-based encryption”, Proceedings of 28th IEEE Symposium on Security and Privacy, 2007

[25] M. Chuah, S. Roy, I. Stoev, “Secure Descriptive Message Dissemination in Disruption Tolerant Networks”, Proceedings of ACM MobiOpp, Feb 2010.

[26] Subharthi Paul , Jianli Pan , Raj Jain, Architectures for the future networks and the next generation Internet: A survey, Computer Communications, v.34 n.1, p.2-42, January, 2011

[27] L. Peterson, A. Bavier, M. E. Fiuczynski, et al., “Experiences building planetlab,” in Proceedings of the 7th symposium on Operating systems design and implementation (OSDI 2006), pp. 351-366, Berkeley, CA, 2006.

[28] PlanetLab,http://www.planet-lab.org

[29] University of Utah, the Emulab Project, http://www.emulab.net

[30] ORBIT, http://groups.geni.net/geni/wiki/ORBIT

[31] ORBIT-Lab, http://www.orbit-lab.org

[32] OMF: control and Management Framework, http://omf.mytestbed.net

[33] http://www.fs.fed.us/r8/fireprevention/currentDanger.php

[34] Anders Lindgren , Avri Doria , Olov Schelén, Probabilistic routing in intermittently connected networks, ACM SIGMOBILE Mobile Computing and Communications Review, v.7 n.3, July 2003.

[35] http://isi.edu/nsnam/ns/

Vita

Xiong Xiong

xix209@lehigh.edu

PROFILE:

- Birthday: April 29, 1984
- Bith Place: Jingmen City, Hubei Province, China
- Mother: Liane Gao
- Father: Zhongyin Xiong

Education:

Lehigh University, MS in Computer Science, 2009~2011

Beijing University of Posts and Telecommunications, ME in Computer Science, 2006~2009.

Beijing University of Posts and Telecommunications, BE in Computer Science, 2002~2006

- Industry Scholarship of BUPT 2004~2005 (top 5%).
- First prize winner of the Undergraduate Electronic Design Contest, Beijing, 2004.

Research Experience:

Jun. 2010 ~ Dec, 2011 **WiNS Lab, Lehigh University**

Position: **Graduate Research Assistance**

May 2007 ~ Mar 2008 **IBM China Research Laboratory**

Position: **Research Intern**

Oct 2006 ~ Apr 2007 **Beijing University of Posts and Telecommunications**

Position: **Research Assistant**

Teaching Experience:

Sep. 2009 ~ May. 2010 **CSE Department, Lehigh University**

Position: **Teaching Assistant**

Sep 2007 ~ Dec 2007 **University for Science and Technology Beijing**

Position: **Lecturer**

Sep 2006 ~ Jun 2007 **Beijing University of Posts and Telecommunications**

Position: **Teaching Assistant**