NDN, Technical Report NDN-0003, 2012. http://named-data.net/techreports.html

OSPFN: An OSPF Based Routing Protocol for Named Data Networking

Lan Wang[*], A K M Mahmudul Hoque[*], Cheng Yi[†], Adam Alyyan[*], Beichuan Zhang[†]

July 25, 2012

Abstract

Named Data Networking (NDN) is a new data-centric network architecture. In NDN, users send Interest messages to retrieve data by their names. Since the Interest messages do not contain source or destination addresses, routers need to forward them based on the names carried in the messages. In order to provide name-based routing capability in NDN, we have extended OSPF to distribute name prefixes and calculate routes to name prefixes. Our protocol OSPFN is currently deployed in the NDN testbed. This report describes our design, implementation, deployment, and future work.

1 Introduction

The Named Data Networking (NDN) [2, 3] architecture represents a fundamental paradigm shift from the current Internet architecture. IP identifies data using its location (an IP address). In NDN, the data consumer sends out an Interest packet, which identifies the data that the consumer is seeking by *name*. The response to an Interest packet in NDN is called a Data packet, which carries both the full name and the data itself. This departure from where data is, to what data is, defines NDN. As a result, NDN routing needs to provide routes to name prefixes rather than address prefixes. Moreover, when NDN routers forward Interest packets, they set up state information

[*]Lan Wang, A K M Mahmudul Hoque, and Adam Alyyan are with University of Memphis. Their email addresses are {lanwang, ahoque1, aalyyan}@memphis.edu.

[†]Cheng Yi and Beichuan Zhang are with University of Arizona. Their email addresses are yic@email.arizona.edu and bzhang@cs.arizona.edu respectively.

so that returning Data packets can be forwarded back to the consumers. This state enables routers to explore multiple routes for each name prefix without having loops. To fully support this multipath forwarding capability, it would be advantageous for the routing protocol to calculate multiple routes to the same name prefix whenever possible.

Our goal is to develop a dynamic routing protocol for NDN to support the above functionality. For the long term, it is desirable to design the new protocol over the NDN architecture directly, i.e., naming routers and messages without using IP addresses as well as employing Interest/Data messages to exchange routing information. However, because there is an urgent need to support all NDN research areas to use the NDN testbed for prototyping and evaluation, we must quickly deploy a dynamic routing protocol for the testbed. For this reason, we decided to extend an existing routing protocol to provide the necessary functionality. We chose Open Shortest Path First (OSPF) [4] because it is widely used in the Internet and it has high-quality open-source implementations. To reach our eventual goal of designing a routing protocol purely on NDN, we started with version 1.0, which is a name based extension of OSPF that runs on IP and supports only single-path routing. We added the support of configred multipath in version 2.0 and we expect to develop version 3.0, which will run on NDN and support automatic multipath.

Our protocol *OSPF for Named-data* (OSPFN) uses Opaque Link State Advertisements [1] to announce name prefixes while ensuring backward compatibility. It supports name prefix advertisement from multiple sites (for the same or different prefixes). Moreover, since OSPF provides only a single best path to each destination, we added a configured multipath feature to allow users to specify which links to use when the best route fails to bring back data. Even though OSPFN does not support full-fledged dynamic multipath routing capability, the configured multipath feature has already helped us better understand the forwarding behavior of the current CCND implementation. OSPFN is currently deployed at all the ten institutions participating in the NDN testbed.

Section 2 discusses our motivation for the project and provides some background information on OSPF. We describe the OSPFN design in detail in section 3. In sections 4 and 5, we present the implementation and configuration of OSPFN using a sample testbed as an example. Section 6 explains the current deployment of OSPFN and outlines the tools we use to monitor it. Finally, in section 7, we conclude by discussing future work related to OSPFN.

2 Motivation and Background

NDN is a novel proposal that retrieves data based on name instead of location. In NDN, routers forward Interest messages by looking up their Forwarding Information Base (FIB) using the names carried in the messages. Although FIBs can be manually configured, such manipulation is time-consuming and error-prone. For the NDN testbed and, more importantly, future NDN deployment, we need a routing protocol that would calculate routes to name prefixes dynamically based on the network topology. In order to address this issue quickly, we designed our solution as an extension of the current OSPF protocol. Open source code availability of OSPF was also an incentive to extend OSPF.

OSPF [4] is a link state routing protocol that works within an autonomous system (AS). Each router in the system gathers link state information about the network to build a Link State Database (LSDB). This LSDB is updated through the flooding of Link State Advertisements (LSA). All the routers in the network have the same copy of the LSDB. Each router builds a network topology from the LSDB and runs the Shortest Path First algorithm to calculate the path(s) to each destination. Whenever there is any change in the network topology, the routing table is recomputed. OSPF is free of persistent loops due to the flooding of link state information. It supports equal-cost multipath, i.e., it produces multiple paths to the same destination if they have the same lowest cost among all the available paths.

OSPF supports Opaque LSA (OLSA) [1] to provide extensibility for future use. OLSA consists of a standard LSA header followed by an application specific data field, which can be used by any external application to extend OSPF. OLSAs are distributed by OSPF throughout the network. This distribution of OLSAs in the network is limited by the flooding scope, which is determined by the Opaque Type field in OLSA header (see Section 3 for more information).

3 Design

Routing in NDN is different from traditional IP routing in two ways: (a) routing on names: data producers register name prefixes, not address prefixes; (b) multipath: routing protocol is expected to provide multiple paths to each name prefix (if such paths exist). Our current protocol "OSPFN" supports routing on names and configured multipath routing.

We use OSPF's Opaque LSAs or OLSA [1] to announce name prefixes.

OLSA allows for application specific information to be advertised in the network. Legacy nodes will not use these LSAs to build their topology, but will still forward them. This ensures backward compatibility with legacy nodes and allows new functions to be implemented among the upgraded nodes. Moreover, open source routing suites, such as Quagga OSPF, provide an API that allows for easy injection and retrieval of OLSAs through OSPF. This flexibility and ease of use makes OLSA the perfect candidate for advertising name prefixes.

OSPFN produces routes to name prefixes and installs them into CCND (Content Centric Network Daemon) [5], which handles the forwarding of Interest and Data messages. More specifically, each NDN router runs CCND, OSPFN and OSPFD (OSPF daemon) in parallel (Figure 1). OSPFN builds Name OLSAs and injects them into the local OSPFD, which floods the OLSAs to the entire network. When the OSPFD at a node receives an OLSA, it delivers the OLSA to its local OSPFN. Since each LSA carries the ID of the router that originated the LSA, OSPFN can obtain the router ID of a Name OLSA and query OSPFD to retrieve the nexthop to reach the router (note that OSPFD still floods its regular LSAs and computes the shortest path tree based on the overlay topology). OSPFN then installs the name prefix and its associated nexthop into the CCND FIB. In the remainder of this section, we describe the format of name OLSA messages, the process for route calculation, and the detailed message exchange process.

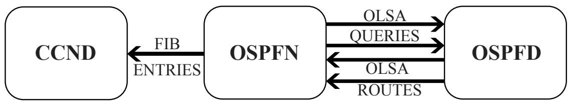


Figure 1: Relationship between CCND, OSPFN, and OSPFD

3.1 Name OLSA Messages

A Name OLSA message carries a single name prefix. Its format is shown in Figure 2. The majority of the fields are assigned by OSPFD in the advertising router. OLSA has three scopes for flooding the network. We use the LS Type “10” for area scope flooding, which means the OSLA will be flooded only within the local area. The Opaque Type field can contain a value within the range 127-255 for application specific use; we use 236 for Name OLSA. The Opaque ID is a unique value assigned by the user to identify this name prefix. The Opaque Information field carries a 32-bit field for the name prefix size, an 8-bit field for the name prefix format, as well as the actual name prefix. The name prefix format can be URI (0) or CCNB (1). Other formats can be defined in the future.

0	8	16	24 31
LS Age		Options	LS Type
Opaque Type	Opaque ID		
Advertising Router			
LS Sequence Number			
LS Checksum		Length	
Opaque Information (variable size)			

LS Age	The age of the LSA in seconds
Options	Optional capabilities associated with the LSA
LS Type	The link-state type of the Opaque LSA that identifies the LSAs range of topological distribution. The three types used are as follows: • 9 - link-local scope; Opaque LSAs are not flooded beyond the local (sub)network • 10 - area-local scope; Opaque LSAs are not flooded beyond the local area • 11 - AS-wide scope; Opaque LSAs are flooded throughout the AS
Opaque Type	Specifies the application specific type of the Opaque LSA
Opaque ID	Type-specific ID
Advertising Router	Router ID of the LSAs originator
LS Sequence Number	Used to detect old and duplicate LSAs
LS Checksum	Checksum of the complete header except the LS Age field
Length	Total header length
Opaque Information	Application-specific data

Name LSA Opaque Information
Size in Bytes of Name Prefix (32 bits)
Name Type (8 bits)
Name Prefix (variable size)

Figure 2: Name OLSA Message Format

3.2 Route Calculation

OSPFN does not perform shortest path calculation – it queries OSPFD for the nexthop to the origin router of a name prefix. When OSPFN receives the query's result, it adds a FIB entry containing the name prefix and the returned nexthop. In the case of a name prefix advertised by multiple routers, OSPFN sends a query for each of the origin routers and inserts a FIB entry containing the name prefix and each returned nexthop.

Because the OSPF protocol provides only a single nexthop for each destination, except when there are equal-cost shortest paths, OSPFN by default generates one route for each single-origin name prefix. However, one of the unique features of NDN is its forwarding strategy, which can explore multiple paths to retrieve a piece of named data. Ideally, we should either modify

OSPF to calculate multiple best paths to each destination or let OSPFN manage the topology information and perform multipath calculation. However, these designs would take a long time to implement. As a poorman's solution to multipath routing, OSPFN allows operators to specify a ranked list of nexthops and it will insert the corresponding routes into CCND's FIB, so that CCND will try them when the best path fails to bring back data. Each interface is associated with a preference; the more preferred interfaces will be tried first. We call this feature "configured multipath routing". To ease the burden on operators, the multipath configuration is specified for each node, not for each name prefix. Note that this is simply an initial order for CCND to explore; CCND's forwarding strategy will choose the best path based on which path retrieves the data fastest.

When configured multipath is used, OSPFN generates a list of FIB entries for each name prefix: the most preferred nexthops are those for the origin routers, ranked by their associated path costs, followed by the configured multipath nexthops by descending order of preference. OSPFN then inserts the FIB entries in the reverse order, as CCND tries the last inserted FIB entry first.

3.3 Messages Exchange Process

When a router boots up, it reads the configuration file and creates a Name OLSA for each name prefix that it wants to advertise to the network. It then sends the Name OLSAs to the local OSPFD to be flooded through the area. The router may also learn the name prefixes it should originate through other means, e.g., another protocol.

OSPFD informs OSPFN whenever there is an update in its LSDB along with the content of the LSA. After receiving an LSA from OSPFD, OSPFN first checks whether the LSA is an OLSA or not. If it is not an OLSA, OSPFN simply discards that LSA. Otherwise, the router checks whether the OLSA is originated by itself. If not, the OLSA is processed. OSPFN reads the name prefix from the Opaque LSA field and creates an entry in its own Name Prefix table containing the name prefix and the origin router. After that, OSPFN sends a query to OSPFD for the nexthop(s) to reach the origin router(s) of each name prefix.

When OSPFD receives a query message from OSPFN, it looks up its routing table for the nexthop list and associated path costs, includes them in a single message and then sends the message back to OSPFN. When OSPFN receives this message from OSPFD, it will use the nexthop list and path costs to update the Name Prefix Table for all name prefixes that have this router as their origin router. Then OSPFN will create FIB entries for each name prefix and insert them into CCND. One FIB entry is created for each next hop for a name prefix.

When OSPFN receives any messages from OSPFD about deleting a Name OLSA, it will delete entries for the name prefix in its own Name Prefix table. OSPFN will then send messages to CCND to delete the corresponding FIB entries from CCND. Figure 3 shows the sequences of messages exchanged between OSPFD, OSPFN and CCND.

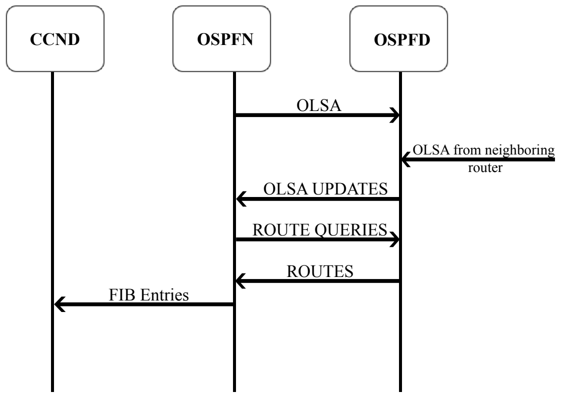


Figure 3: Sequences of Messages exchanges among OSPFN, OSPFD and CCND

4 Implementation

OSPFN is developed based on Quagga 0.99.17 [6] and tested on Ubuntu 10.04, 10.10, 11.04, 12.04, Fedora 9, and FreeBSD 9.0. It is implemented and distributed by the University of Memphis and the University of Arizona. We released the first version on Oct. 18, 2011 and the second version on May 15, 2012. Our code is open source and available at github.com. OSPFN2.0 can be downloaded from `https://github.com/NDN-Routing/OSPFN2.0`. Below we describe several important implementation details.

4.1 Configuration

Table 1 shows the configuration commands supported in the current OSPFN implementation.

`ccnnametype type`	
Function	specify the format of the subsequent name prefixes
Parameter	`type`: name prefix format, 0 (URI, default) or 1 (CCNB).
`ccnname name_prefix op_id`	
Function	specify a name prefix to be originated
Parameters	`name_prefix`: name prefix to be originated `op_id`: a unique ID for this name LSA. The op_id must be unique among the name prefixes advertised by the same router.
`multipath-order a.b.c.d pref_order`	
Function	specify the interfaces to be explored by CCND and their associated preferences
Parameters	`a.b.c.d`: next hop address `pref_order`: preference of next hop
`logdir dir`	
Function	specify the directory of the log files
Parameter	`dir`: directory for logging

Table 1: OSPFN Configuration Commands

4.2 LSA Origination

OSPFN originates OLSAs after processing its configuration file. For each `ccnname` command in the configuration file, OSPFN creates a Name OLSA with LS Type 10, Opaque Type 236, and the configured Opaque ID. The Opaque Data field is set with the size of the name prefix, name type from the `ccnnametype` command, and the name prefix. OSPFN then injects this OLSA into OSPFD.

4.3 Multipath Ordering

When OSPFN injects routes into the CCND FIB, the best path will be injected last so that CCND will try it first. The other interfaces will be inserted based on their preference specified by the operator. For example, suppose a router has three interfaces with the following preferences:

```
multipath-order 10.0.1.2 3
multipath-order 10.0.2.1 2
multipath-order 10.0.8.2 1
```

If 10.0.2.1 is the best nexthop to the name prefix P, then OSPFN will construct the FIB entries such that CCND will first use 10.0.2.1 to forward Interest packets under P. If this interface fails to bring back data, then it will try 10.0.1.2, followed by 10.0.8.2. When the best nexthop changes, OSPFN will adjust the FIB entries accordingly. For example, if 10.0.8.2 becomes the new best nexthop, then the order becomes 10.0.8.2, 10.0.1.2 and 10.0.2.1.

5 Example

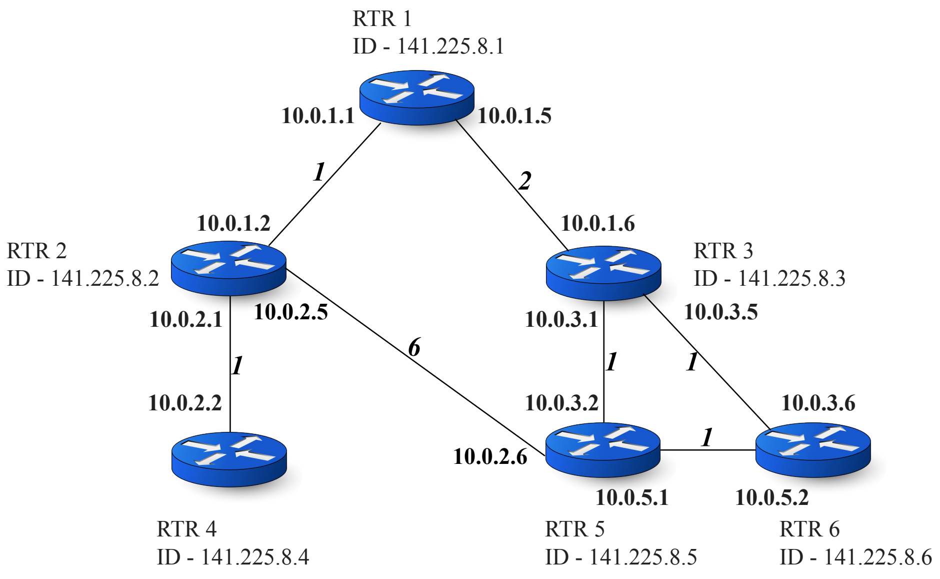


Figure 4: Network Topology

To demonstrate the design operation, the sample network topology in Figure 4 will be used. The sample network consists of 6 NDN routers. Each router's public IP is displayed in the figure, as well as the tunnel address for each router. The cost of each link is also shown. Each NDN router has direct access to the name prefixes shown in Table 2, which also displays the origin router of each name prefix. Each router is initially configured with the name prefixes they want to advertise using the textbfccnname command. Optionally, a router can specify back up paths using the **multipath-order** command and the logging directory using the **logdir** command in its config-

Router	Name Prefixes
RTR1	/ndn/flying/delta /ndn/flying/aa /ndn/flying/ticketprices
RTR2	/ndn/sports/soccer /ndn/sports/bb
RTR3	/ndn/lifestyle/home /ndn/lifestyle/cooking
RTR4	/ndn/leisure/park /ndn/leisure/theater
RTR5	/ndn/travel/Michigan /ndn/travel/Illinois
RTR6	/ndn/airport/Detroit /ndn/airport/Chicago /ndn/flying/ticketprices

Table 2: Name Prefixes Originated by Each Router

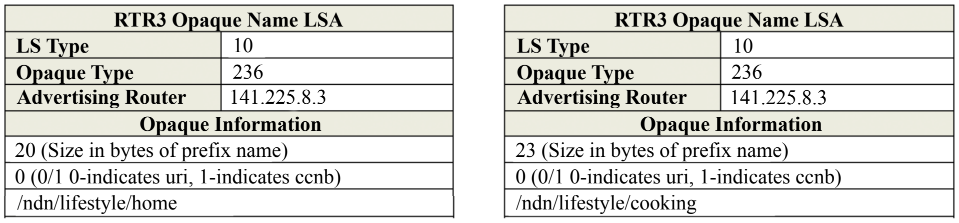
RTR1 Configuration File
ccnname /ndn/flying/delta 1 ccnname /ndn/flying/aa 2 ccnname /ndn/flying/ticketprices 3 multipath-order 10.0.1.2 10 multipath-order 10.0.1.6 20
RTR3 Configuration File
ccnname /ndn/lifestyle/home 1 ccnname /ndn/lifestyle/cooking 2 multipath-order 10.0.1.5 10 multipath-order 10.0.3.6 15 multipath-order 10.0.3.2 20
RTR6 Configuration File
ccnname /ndn/airport/Detroit 1 ccnname /ndn/airport/Chicago 2 ccnname /ndn/flying/ticketprices 3

Table 3: Router Configuration Files

uration file. The configuration files for RTR1, RTR3, and RTR6 are shown below in Table 3. Each configuration file consists of the name prefix that the router is originating, and in the case of RTRs 1 and 3, the configured multipath settings.

The OSPFN on each node will build a Name OLSA for each name prefix in its configuration file to advertise to the network. It will inject the Name OLSAs and handle the arrival/deletion/update of Name OLSAs from other routers through the API provided by OSPFD. Each node will then be able to build a Name Prefix Table containing the prefix names of all content advertised in the network. The Name OLSAs for RTR3 are shown in Figure 5, which contains important information about the Name OLSAs such as the type, size of the name prefix, as well as the name prefix itself. RTR3's Name Prefix Table is shown in Figure 6; for each name prefix, the corresponding origin router is shown.

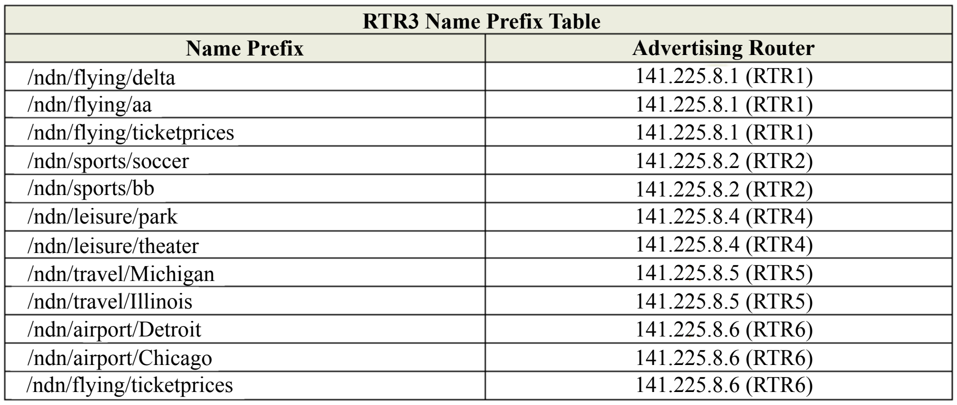
Each NDN node will then construct FIB entries for each unique name prefix in its Name Prefix Table. In the following example a FIB entry for RTR3 will be constructed for the name prefix `/ndn/flying/ticketprices`. First the origin router (or routers) for the name prefix is looked up in the Name Prefix Table, then, the next hops to the origin routers are obtained from the OSPFD Routing Table (Figure 7).



RTR3 Opaque Name LSA	
LS Type	10
Opaque Type	236
Advertising Router	141.225.8.3
Opaque Information	
20 (Size in bytes of prefix name)	
0 (0/1 0-indicates uri, 1-indicates ccnb)	
/ndn/lifestyle/home	

RTR3 Opaque Name LSA	
LS Type	10
Opaque Type	236
Advertising Router	141.225.8.3
Opaque Information	
23 (Size in bytes of prefix name)	
0 (0/1 0-indicates uri, 1-indicates ccnb)	
/ndn/lifestyle/cooking	

Figure 5: Name OLSA for RTR3



RTR3 Name Prefix Table	
Name Prefix	**Advertising Router**
/ndn/flying/delta	141.225.8.1 (RTR1)
/ndn/flying/aa	141.225.8.1 (RTR1)
/ndn/flying/ticketprices	141.225.8.1 (RTR1)
/ndn/sports/soccer	141.225.8.2 (RTR2)
/ndn/sports/bb	141.225.8.2 (RTR2)
/ndn/leisure/park	141.225.8.4 (RTR4)
/ndn/leisure/theater	141.225.8.4 (RTR4)
/ndn/travel/Michigan	141.225.8.5 (RTR5)
/ndn/travel/Illinois	141.225.8.5 (RTR5)
/ndn/airport/Detroit	141.225.8.6 (RTR6)
/ndn/airport/Chicago	141.225.8.6 (RTR6)
/ndn/flying/ticketprices	141.225.8.6 (RTR6)

Figure 6: RTR3's Name Prefix Table

For a name prefix advertised by a single router, a FIB entry is constructed containing the name prefix and the next hop to the origin router. Then the constructed FIB entry is inserted into the CCND FIB. If a name prefix is advertised by multiple routers, FIB entries are constructed for each next hop to the origin routers. Then, these FIB entries are added in CCND FIB in decreasing order of path cost to the origin routers. Higher preference is given to lower cost path.

Now if a router has been configured for multipath, it will construct and insert FIB entries for each configured next hop for a name prefix by increasing order of preference, followed by any next hops for the origin routers of that name prefix by decreasing path cost. Since CCND uses the last inserted FIB entry first, OSPFN makes sure that the origin routers' nexthops get higher preference than other configured nexthops. Partial FIB of RTR3 will look like Figure 8, where the next hops are ordered from lowest to highest preference. To find RTR3's next hops for `/ndn/flying/ticketprices`, firstly the origin router(s) of the prefix, i.e., RTR's 1 and 6, are looked up

RTR3 OSPF Routing Table	
Destination	**Next Hop**
RTR1	10.0.1.5 (RTR1)
RTR2	10.0.1.5 (RTR1)
RTR4	10.0.1.5 (RTR1)
RTR5	10.0.3.2 (RTR5)
RTR6	10.0.3.6 (RTR6)

Figure 7: RTR3's Routing Table

in the OSPF routing table. This yields the next hop associated with each router, 10.0.1.5 (RTR1) and 10.0.3.6 (RTR6), respectively. The path cost is also looked up and the lower value wins. RTR6 has a path cost of 1 from RTR3, while RTR1 has a path cost of 2 from RTR3, thus RTR6 becomes the best next hop, followed by RTR1. This gives us the first 2 next hops for the prefix. In our example, RTR3 has a backup path due to its multipath configuration, therefore the backup link is used as the next hop, which is RTR5. A similar process is performed to obtain the next hops for /ndn/sports/soccer.

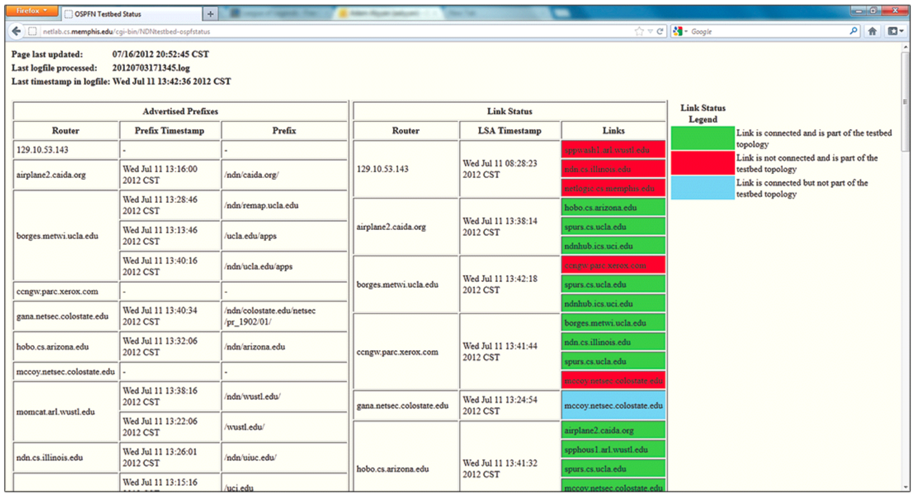
RTR3 FIB	
Name Prefix	**Next Hop**
/ndn/flying/ticketprices	10.0.5.1 (RTR5)
/ndn/flying/ticketprices	10.0.1.5 (RTR1)
/ndn/flying/ticketprices	10.0.3.6 (RTR6)
...	...
/ndn/sports/soccer	10.0.3.6 (RTR6)
/ndn/sports/soccer	10.0.5.1 (RTR5)
/ndn/sports/soccer	10.0.1.5 (RTR1)
...	...

Figure 8: RTR3's FIB Table

6 Deployment and Testbed Status Monitoring

We deployed the first version of OSPFN in Oct. 2011 and the second version in May 2012 over the NDN testbed. All the 10 sites on the NDN testbed have deployed OSPFN 2.0 as of June 28 2012, including UCLA Remap, PARC, University of Arizona, Colorado State University, Washington University (including the Supercharged PlanetLab Platform or SPP nodes), University of Memphis, CAIDA, UCLA, Northeastern University and UIUC. From the deployment experience, we noticed that setting up and configuring GRE tunnels in different OSes accounted for the majority of our problems. Managing private IPs is also an issue in network management. We hope to get rid of these problems when the next version is released, which will be a pure name-based design using NDN messages.

We have developed web-based tools to monitor which prefixes are advertised by OSPFN and which neighbors are advertised by OSPFD at each NDN hub (Figure 9). The current state of each link is also displayed along with a timestamp of when the corresponding LSA was received. The timestamp at the top of the page indicates the time the page was last updated. Another monitoring tool we use is the CCND status web page (Figure 10), which helps us verify whether OSPFN installs routes correctly into CCND.



Page last updated: 07/16/2012 20:52:45 CST
Last logfile processed: 20120703171345.log
Last timestamp in logfile: Wed Jul 11 13:42:36 2012 CST

Advertised Prefixes		
Router	**Prefix Timestamp**	**Prefix**
129.10.53.143	-	-
airplane2.caida.org	Wed Jul 11 13:16:00 2012 CST	/ndn/caida.org/
borges.metwi.ucla.edu	Wed Jul 11 13:28:46 2012 CST	/ndn/remap.ucla.edu
	Wed Jul 11 13:13:46 2012 CST	/ucla.edu/apps
	Wed Jul 11 13:40:16 2012 CST	/ndn/ucla.edu/apps
ccngw.parc.xerox.com	-	-
gana.netsec.colostate.edu	Wed Jul 11 13:40:34 2012 CST	/ndn/colostate.edu/netsec/pr_1902/01/
hobo.cs.arizona.edu	Wed Jul 11 13:32:06 2012 CST	/ndn/arizona.edu
mccoy.netsec.colostate.edu	-	-
momcat.arl.wustl.edu	Wed Jul 11 13:38:16 2012 CST	/ndn/wustl.edu/
	Wed Jul 11 13:22:06 2012 CST	/wustl.edu/
ndn.cs.illinois.edu	Wed Jul 11 13:26:01 2012 CST	/ndn/uiuc.edu/
	Wed Jul 11 13:15:16	/uci.edu

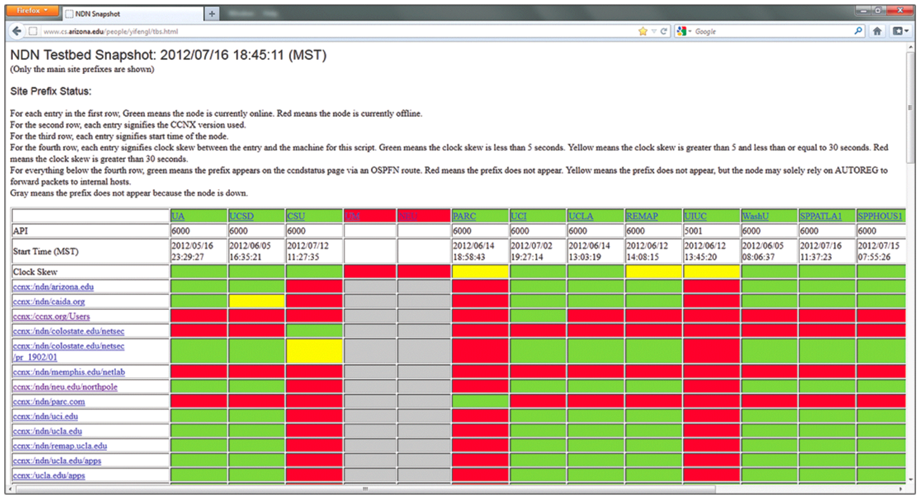
Link Status		
Router	**LSA Timestamp**	**Links**
129.10.53.143	Wed Jul 11 08:28:23 2012 CST	🟥 sppwash1.arl.wustl.edu
		🟥 ndn.cs.illinois.edu
		🟥 netlogic.cs.memphis.edu
airplane2.caida.org	Wed Jul 11 13:38:14 2012 CST	🟩 hobo.cs.arizona.edu
		🟩 spurs.cs.ucla.edu
		🟩 ndnhub.ics.uci.edu
borges.metwi.ucla.edu	Wed Jul 11 13:42:18 2012 CST	🟥 ccngw.parc.xerox.com
		🟩 spurs.cs.ucla.edu
		🟩 ndnhub.ics.uci.edu
ccngw.parc.xerox.com	Wed Jul 11 13:41:44 2012 CST	🟩 borges.metwi.ucla.edu
		🟩 ndn.cs.illinois.edu
		🟩 spurs.cs.ucla.edu
		🟥 mccoy.netsec.colostate.edu
gana.netsec.colostate.edu	Wed Jul 11 13:24:54 2012 CST	🟦 mccoy.netsec.colostate.edu
hobo.cs.arizona.edu	Wed Jul 11 13:41:32 2012 CST	🟩 airplane2.caida.org
		🟩 spphous1.arl.wustl.edu
		🟩 spurs.cs.ucla.edu
		🟩 mccoy.netsec.colostate.edu

Link Status Legend	
🟩	Link is connected and is part of the testbed topology
🟥	Link is not connected and is part of the testbed topology
🟦	Link is connected but not part of the testbed topology

Figure 9: OSPFN Status on NDN Testbed

7 Future Work

We plan to continue running and maintaining OSPFN 2.0 on the NDN testbed. At the same time, we would also like to ensure the long-term viability of NDN routing. Therefore, we have recently started developing an OSPF protocol over NDN. It will not rely on IP addresses, but rather, each router and interface are named. It will also automatically calculate multiple forwarding choices without user configuration.


NDN Snapshot

www.cs.arizona.edu/people/yifengl/tbs.html

NDN Testbed Snapshot: 2012/07/16 18:45:11 (MST)
(Only the main site prefixes are shown)

Site Prefix Status:

For each entry in the first row, Green means the node is currently online. Red means the node is currently offline.
For the second row, each entry signifies the CCNX version used.
For the third row, each entry signifies start time of the node.
For the fourth row, each entry signifies clock skew between the entry and the machine for this script. Green means the clock skew is less than 5 seconds. Yellow means the clock skew is greater than 5 and less than or equal to 30 seconds. Red means the clock skew is greater than 30 seconds.
For everything below the fourth row, green means the prefix appears on the ccndstatus page via an OSPFN route. Red means the prefix does not appear. Yellow means the prefix does not appear, but the node may solely rely on AUTOREG to forward packets to internal hosts.
Gray means the prefix does not appear because the node is down.

	🟩 UA	🟩 UCSD	🟩 CSU	🟥 UM	🟥 NEU	🟩 PARC	🟩 UCI	🟩 UCLA	🟩 REMAP	🟩 UIUC	🟩 WashU	🟩 SPPATLA1	🟩 SPPHOUS1
API	6000	6000	6000			6000	6000	6000	6000	5001	6000	6000	6000
Start Time (MST)	2012/05/16 23:29:27	2012/06/05 16:35:21	2012/07/12 11:27:35			2012/06/14 18:58:43	2012/07/02 19:27:14	2012/06/14 13:03:19	2012/06/12 14:08:15	2012/06/12 13:45:20	2012/06/05 08:06:37	2012/07/16 11:37:23	2012/07/15 07:55:26
Clock Skew	🟩	🟩	🟩	🟥	🟥	🟨	🟩	🟩	🟨	🟨	🟩	🟩	🟩
ccnx:/ndn/arizona.edu	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/caida.org	🟩	🟨	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ccnx.org/Users	🟥	🟥	🟥	⬜	⬜	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥
ccnx:/ndn/colostate.edu/netsec	🟥	🟥	🟩	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
ccnx:/ndn/colostate.edu/netsec/pr_1902/01	🟩	🟩	🟨	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/memphis.edu/netlab	🟥	🟥	🟥	⬜	⬜	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
ccnx:/ndn/neu.edu/northpole	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/parc.com	🟥	🟥	🟥	⬜	⬜	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥
ccnx:/ndn/uci.edu	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/ucla.edu	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/remap.ucla.edu	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ndn/ucla.edu/apps	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩
ccnx:/ucla.edu/apps	🟩	🟩	🟥	⬜	⬜	🟥	🟩	🟩	🟩	🟥	🟩	🟩	🟩

Figure 10: OSPFN routes installed into CCND

8 Acknowledgments

Yaoqing Liu participated in the implementation of OSPFN 1.0. Gus Sanders developed the web-based OSPFN status monitoring tool. Yifeng Li developed the CCND status webpage. We would also like to thank Lixia Zhang, Van Jacobson, James Thornton, Michael Plass and Syed Obaid Amin for their valuable feedback. Finally, we thank all the NDN testbed operators for deploying OSPFN at their sites.

References

[1] L. Berger, I. Brystkin, A. Zinin, and R. Coltun. The OSPF opaque LSA option. *RFC 5250*, July 2008.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of ACM CoNEXT*, 2009.

[3] L. Zhang et al. Named data networking (NDN) project. Technical Report NDN-0001, PARC, October 2010.

[4] J. Moy. OSPF version 2. *RFC 2328*, Apr. 1998.

[5] PARC. CCNx open srouce platform. `http://www.ccnx.org`.

[6] Quagga routing software suite. `http://www.quagga.net`.