NDN, Technical Report NDN-0016, 2013. http://named-data.net/techreports.html
Revision 1: December 7, 2013

On the Role of Routing in Named Data Networking

Cheng Yi[†], Jerald Abraham[†], Alexander Afanasyev[‡], Lan Wang[§], Beichuan Zhang[†], Lixia Zhang[‡]
[†]*University of Arizona,* [‡]*UCLA,* [§]*University of Memphis*
{*yic, jeraldabraham, bzhang*}*@cs.arizona.edu,* {*afanasev, lixia*}*@cs.ucla.edu, lanwang@memphis.edu*

***Abstract*—A unique feature of Named Data Networking (NDN) is that its forwarding plane can detect and recover from network faults on its own. Consequently, NDN routers are able to handle network failures locally without relying on global routing convergence. This fundamental change prompts us to rethink the role of routing in NDN networks: does it still need a routing protocol? If so, what impact may an intelligent forwarding plane have on the design and operation of NDN routing protocols? Through analysis and extensive simulations, we show that a routing protocol remains necessary in NDN networks. Routing disseminates initial topology and policy information as well as long-term changes in them, and computes the routing table to guide the forwarding process. However, since the forwarding plane is capable of detecting failures and recovering quickly, routing no longer needs to handle short-term churns in the network. Freeing routing protocols from short-term churns can greatly improve their scalability and stability, enabling NDN to use routing protocols that were previously viewed as unsuitable for real networks.**

I. Introduction

Named Data Networking (NDN) is a new network architecture that changes the basic network service semantics from "delivering packet to a given destination" to "retrieving data with a given name." NDN communication is receiver-driven: a data consumer sends *Interest* packets carrying the names of desired data; any node in the network can return *Data* packets that have matching names to satisfy the Interests. This two-way Interest-Data packet exchange takes the same network path but in opposite directions.

Symmetric Interest-Data exchange and in-network forwarding state enable a unique feature of NDN – *adaptive forwarding* ([1], [2]). More specifically, a node expects a Data packet to come back from the same interface where it forwarded the Interest within a reasonable time period (e.g., round-trip time), otherwise it should receive a NACK packet [2] or get a timeout, which signals a failure of this attempt. Upon detection of a failure, the node can then send the Interest to other interfaces to explore alternate paths. This built-in failure detection and recovery capability works on the forwarding plane, with no intervention from the control plane. Our earlier work [2] shows that NDN's adaptive forwarding can handle link failures, prefix hijacking, and congestion control more effectively than IP networks.

Having an intelligent adaptive forwarding plane raises new research questions. Today's IP networks put all intelligence into routing, which disseminates topology and policy information, computes routes, detects and recovers from failures while the data plane merely forwards packets according to the FIB. When the data plane has its own adaptability, are routing protocols still needed? If so, for what purpose and to what extent? If some of routing's tasks can be offloaded to forwarding, would that bring positive impact on routing protocols' design and operation, e.g., making routing more scalable and stable?

In this paper we investigate the role of routing in NDN networks. Through analysis, design, and extensive simulation, we find that routing is important in bootstrapping the forwarding plane for effective data retrieval, as well as for efficiently probing new links or recovered links. However, *NDN routing does not need to converge fast following network changes*, which can be handled by adaptive forwarding more promptly. This enables one to significantly improve the scalability and stability of the routing system using larger keep-alive timer values that ignore short-term failures. Furthermore, routing algorithms that would not work well in current networks may work fine in NDN due to its reduced role of bootstrapping adaptive forwarding.

The rest of this paper is organized as follows. Section II reviews NDN with a focus on the adaptive forwarding plane. Section III discusses the role of routing in both IP and NDN. The coordination of NDN routing and forwarding is explained in Section IV. Section V evaluates the performance of the coordination. Section VI discusses other possible routing schemes for NDN. Section VII presents related work and Section VIII concludes the paper.

II. NDN Forwarding Overview

Each NDN packet carries a *name* field that uniquely identifies a piece of data, e.g., /ndn/papers/routing.pdf/seg1. NDN routers forward Interests based on the names, and keep forwarding state for each pending Interest. When Data packets arrive, routers use names to match them to corresponding pending Interests and forward them accordingly. Each Interest also carries a *nonce* field that can be used to detect forwarding loops. In this section we briefly review NDN's forwarding process and how it handles link failures.

A. Forwarding Process

There are three key data structures in NDN's node model, i.e., *Forwarding Information Base* (FIB), *Pending Interest Table* (PIT) and *Content Store* (CS). FIB serves as the

forwarding table. It is different from the FIB in IP routers in that it is indexed by name prefixes instead of IP prefixes, and each FIB entry may provide multiple interfaces instead of a single best interface for each name prefix. Unlike FIB, PIT and CS are unique to NDN. Both PIT and CS are indexed by names. A PIT entry records which interface(s) the Interest is received from and has been forwarded to, and is used to guide Data forwarding. CS is a temporary cache of Data packets. If the desired Data is found in the CS, the Interest is satisfied immediately without being further forwarded.

In NDN, only Interest packets are routed. The forwarding process is summarized as follows. When a router receives an Interest, it first checks the Interest name against the CS and returns the Data if there is a match. Otherwise, the router checks the Interest name against the PIT. If a PIT entry already exists, i.e., the Interest has already been forwarded but no Data has been returned yet, the router simply adds the incoming interface of the Interest to the PIT entry. If no PIT entry exists, the router adds a new PIT entry and further looks up the Interest name in the FIB using longest prefix match. If a matching FIB entry is found, the Interest is forwarded by the *forwarding strategy* module. Otherwise, the router cannot satisfy the Interest and may send a NACK back to the incoming interface of the Interest [2]. When a router receives a Data, it checks the Data name against the PIT. If a PIT entry is found, the Data is stored in the CS and further forwarded to the incoming interfaces of the corresponding Interests, which have been recorded in the PIT. Otherwise, the Data is dropped since it is either unrequested or no longer wanted.

The forwarding strategy associated with the name space of the requested data determines whether and how an Interest is forwarded. In this paper we adopt the forwarding strategy proposed in [2]. Each interface is assigned a color code depending on its current working status. It is *Green* for a working interface, *Red* if the interface is not working, and *Yellow* if the status is uncertain. The forwarding strategy always prefers Green interfaces over Yellow ones, and never uses Red interfaces to forward Interests. The forwarding strategy takes information such as interface ranking from the routing protocol, interface status, round-trip time (RTT) and congestion level into consideration when making forwarding decisions. Interested readers may refer to [2] for details.

B. Failure Recovery

NDN's two-way symmetric traffic flow enables fast fault detection. Routers can calculate RTT for each Interest-Data exchange, which can be used as a prediction for future Interests. After forwarding an Interest, a router starts a timer based on the average of previous RTT; potential network problems can be detected if no Data is received before the timer expires. With Interest NACK [2], fault detection and notification is even faster. When network problems are detected, routers can explore alternative paths freely without worrying about loops, since they can be detected by checking the nonce field carried in the Interests. Fast fault detection and loop-free forwarding are the two unique features that make NDN's forwarding plane smart and adaptive. As a result, routers are able to handle network faults such as prefix hijacking, failures and congestion locally at the forwarding plane [2].

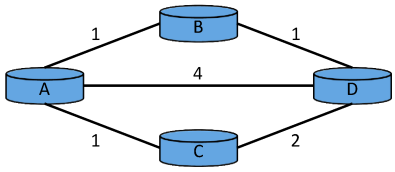


Figure 1. A simple network example.

We use the simple example in Figure 1 to illustrate how NDN routers handle link failures. The cost of the links are marked in the figure; routers rank the interfaces using the cost of their best paths towards the destination. When there is no failure in the network, A uses B as its major next hop for content provided by D. Interface A-B will be marked Green if there is continuous traffic. When link B-D fails, A will keep sending Interests to B at first. However, B cannot satisfy the Interests due to the failure, so it will send NACKs back to A. Upon receiving a NACK, A will mark A-B Yellow and retry the next best interface, in this case A-C. Since there is no failure on this path, Data will flow back through path D-C-A. A will then mark interface A-C Green and start using C as the major next hop.

III. Role of Routing

Since NDN's forwarding model is a strict superset of the IP model, any routing scheme that works well for IP should also work well for NDN [3]. However, today's IP routing protocols suffer from issues such as slow convergence or poor scalability. In addition, NDN has a smart and powerful forwarding plane, which is able to take over part of routing's responsibility in IP. In this section, we first review IP routing, and then rethink the role of routing in NDN.

A. Routing in IP

IP's routing plane is intelligent and adaptive, but its forwarding plane is stateless and strictly follows routing. Therefore the routing plane is also regarded as the control plane. Routing is responsible for building the routing table and maintaining it in face of network changes, including both long-term topology and policy changes as well as short-term churns. When there is a change in the network, routers need to exchange routing updates with each other in order to reach new global consistency. The time period after a change happens and before all routers agree on the new routing state is called the routing convergence period. IP routing protocols need to converge fast in order to reduce packet loss and resume packet delivery after network changes.

However, fast routing convergence is challenging in large operational networks. The fundamental reason is that it

conflicts with other design goals for routing protocols, i.e., routing stability and scalability. Routing stability ensures stable routing paths within the network. It is important for applications that suffer from RTT fluctuation; it also helps routers achieve traffic engineering goals. Routing scalability is essential for supporting a large number of nodes, links and prefixes[1] in the network. For link-state routing, each router knows the entire topology. These protocols can converge fast, but at the cost of poor stability and limited scalability. For distance/path-vector routing, routers do not have a full knowledge of the topology. They are able to achieve better scalability, but the convergence time may be as long as tens of minutes. Below we use link-state routing as an example to explain the issues with today's routing protocols.

The routing convergence period can be divided into four phases, i.e., failure detection, update propagation, route computation and FIB update. In link-state routing, routers periodically exchange HELLO messages to maintain connection: if no HELLO message is received within the DEAD interval, the link is considered down. Previous research ([4], [5]) recommended setting the HELLO interval to be on the order of milliseconds in order to detect failures quickly. However, this not only increases overhead but also affects routing stability, since a temporarily congested link may be mistakenly considered fluctuating down and up. After a link failure is detected, attached routers need to generate routing updates and propagate them to the rest of the network; when a router receives a routing update, it needs to recompute the routing table. To achieve fast routing convergence, all these steps should be done as quickly as possible. However, if the network is unstable (e.g., there is a flapping link), generating routing updates and recomputing routing table frequently will increase bandwidth and computation overhead as well as harm routing stability. At the same time, shortest path first (SPF) computation time increases with the size of the network; FIB update time depends on the number of prefixes. To achieve fast convergence, both the network size and the number of prefixes need to be limited, leading to poor scalability.

There are mechanisms to improve link-state routing stability and scalability. Dynamic timers improve routing stability by limiting the rate of update generation and SPF computation. However, these timers are increased exponentially each time, potentially increasing convergence time significantly when the network is unstable. Therefore, short initial timers have been suggested [5]. *Area* was introduced to improve routing scalability. However, it leads to sub-optimal paths between areas and increases the complexity of configuration. Although inter-area routing can utilize distance-vector or path-vector routing algorithms that may scale better, they converge much slower.

[1]Supporting large number of prefixes is particularly important in NDN since the number of name prefixes will be orders of magnitude larger than the number of IP prefixes in today's Internet.

In summary, it is hard to achieve fast convergence, stability and scalability simultaneously in a routing protocol. If there are other mechanisms to handle failures without global convergence, the requirement on fast convergence can be relaxed, making it possible to achieve routing stability and scalability.

B. Routing in NDN

In NDN, the forwarding plane is the actual control plane since the forwarding strategy module makes forwarding decisions on its own. This fundamental change prompts us to rethink the role of routing. The first question is whether NDN still needs routing protocols. Conventionally, routing protocols are responsible for disseminating topology and policy information, computing routes and handling short-term network changes. For NDN to work without routing, routers need to be able to do the following things efficiently: 1) retrieve Data when the network is stable; 2) handle link failures; and 3) handle link recovery. Can NDN achieve these solely with the forwarding plane?

Another question that arises is if NDN does need routing protocols, how will they be different from today's existing routing protocols? With the intelligent and adaptive forwarding plane, can some of the routing plane's functionality be offloaded to the forwarding plane, and which? In addition, how will the design and operation of routing protocols benefit from this shift of functionality? In the next section we try to give answers to these questions.

IV. Routing and Forwarding Coordination

In this section, we seek answers to the questions raised in III-B. Previous research [2] shows that NDN routers are able to detect and recover from link failures effectively without routing. In this section we focus on the other questions: whether NDN routers can efficiently retrieve Data and handle link recovery without routing. We show that NDN does need routing protocols to help bootstrap the forwarding process and handle link recovery. In addition, we specify how the routing plane coordinates with the forwarding plane, and present a simple method to improve routing stability and scalability in NDN.

A. Interface Ranking

The forwarding plane design presented in [2] assumes interfaces are ranked by routing preference. Can NDN routers retrieve Data efficiently without such interface ranking? The answer is negative. In the extreme case, we can implement a forwarding strategy that floods every Interest to all available interfaces. This way we can always retrieve Data quickly through the best paths. However, it will also incur substantially large overhead. We can also implement a strategy that randomly explore the interfaces one-by-one in a round-robin fashion. Given enough time, routers should be able to find working paths since all possible paths will be explored. One

Pseudocode 1 ProbingDue Algorithm

```
1: function PROBINGDUE(FibEntry, Intf)
2:     if Intf ≠ FibEntry.RoutingPreferredIntf then
3:         if FibEntry.LastProbingTime + M ≤ Now() or
4:             FibEntry.PacketsSinceLastProbing ≥ N then
5:             Return True
6:         end if
7:     end if
8:     Return False
9: end function
```

big issue with this method is that path exploration may take extremely long time as shown in Section V-B.

Consequently, NDN routers need good interface ranking to help bootstrap the forwarding process. The responsibility of providing interface ranking lies in the routing protocols. Existing routing algorithms such as link-state or distance/path-vector routing can be used to rank the interfaces[2]. The details are explained as follows.

1) Link-State Routing: Link-state routing protocols store the entire network topology in the link-state database (LSDB), making it possible to compute optimal interface ranking. Suppose a node N has n interfaces I_1 .. I_n. For Data provided by node M, we rank these interfaces using $C^M_{N,k}$, which is the cost of the best path from N to M through interface I_k. One simple method to compute $C_{N,k}$ for all destinations is to remove all interfaces except I_k from N's LSDB, and run *Dijkstra's algorithm* to compute the shortest paths. This may not be the best method since it will end up calling Dijkstra's algorithm once for every interface. It is just used to illustrate how interface ranking can be done in link-state routing. Optimization of the algorithm is possible but out of the scope of this paper.

2) Distance/Path-Vector Routing: In distance-vector or path-vector routing, routers announce the hop count or complete routing path towards each destination to their neighbors. When router N receives a routing announcement for Data provided by M from interface I_k, it simply records the hop count $H^M_{N,k}$[3]. The interfaces are then ranked by the hop count.

Notice that a router may not receive routing announcement from all interfaces, since these routing protocols often incorporate split-horizon route announcement to prevent routing loops. If router N learns a route towards M through interface I_k, it will not advertise its route to M over I_k. Interfaces that do not receive routing announcement are assigned infinite hop count to ensure they stay at the end of the ranked interface list. They will only be used as the last resort if all higher-ranked interfaces fail to retrieve Data.

These interfaces are useful in many situations. For example, in BGP if a provider P uses a customer C as the next hop, it will not make routing announcement to C. If C's best path fails, it will not have an alternative path until routing converges, in which case P will announce its alternative path to C. RBGP [6] is proposed to address this issue by allowing P to announce its alternative path to C even without failures. NDN, on the other hand, is able to achieve the same effect without changing the protocol.

[2]The case of path-vector routing, i.e., BGP is more complex because it also takes routing policy into consideration. How to accommodate routing policy in interface ranking is part of our future work.

[3]Hop count can be easily extracted from the path in path-vector routing.

Pseudocode 2 Probing Algorithm

```
1: function PROBE(Interest, FibEntry, PitEntry)
2:     interface ← FibEntry.RoutingPreferredIntf
3:     if interface ∉ PitEntry.Outgoing and
4:       interface ∉ PitEntry.Incoming then
5:         if interface.Available then
6:             Interest.Nonce ← GenerateNonce()
7:             Transmit(interface, Interest)
8:             Add interface to PitEntry.Outgoing
9:             FibEntry.LastProbingTime ← Now()
10:            FibEntry.PacketsSinceLastProbing ← 0
11:        end if
12:    end if
13: end function
```

B. Probing

It has been shown that NDN routers can handle link failures locally at the forwarding plane [2]. In this subsection we answer the question of whether the same applies to link recovery. Routers can detect link failures quickly by observing Interest-Data exchanges or Interest NACK. However, there is no explicit signal for link recovery from the forwarding plane. Again let's take Figure 1 as an example. After interface B-D recovers from a failure, interface A-B becomes the best interface for A to retrieve data from D. However, A will continue using interface A-C because the forwarding strategy prefers Green interfaces over Yellow ones. In this case, A needs to probe interface A-B by sending a copy of an Interest to it. If the probing Interest successfully brings Data back, interface A-B will be marked Green and be used to forward subsequent Interests to D.

There is a research question of when to perform probing. An Interest copy is used for probing so that regular Data retrieval will not be affected if probing is unsuccessful. However, this causes extra Interest and Data in the network. There is a trade-off between how fast a link recovery is detected and the amount of overhead caused by probing. In CCNx [7], a prototype implementation of NDN, routers probe alternative interfaces periodically in order to detect better paths. This enables routers to detect link recovery at the forwarding plane. Fast recovery detection is achievable through aggressive probing. However, it will incur significant overhead.

In fact, routing is able to help with the dilemma. If there is a routing protocol, it will be able to detect link recovery

Table I
TOPOLOGIES USED IN THE SIMULATIONS.

Topology	Before Processing		After Processing	
	Node #	Link #	Node #	Link #
Abilene	12	30	11	28
AS1239-PoP	52	168	32	128
AS701-PoP	83	438	47	366
AS1239-Router	284	1882	N/A	N/A

and converge to it. We can take advantage of routing by only probing a Yellow interface if its ranking is higher than the Green interface(s). This way we can keep the probing overhead low, and switch back to the optimal paths as soon as routing converges. Routing convergence time is not a concern because the alternative paths found by the forwarding plane are of good quality [2]. Notice that probing is also useful in failure handling if the alternative paths found by the forwarding plane are not optimal.

We propose a probing algorithm as presented in Pseudocode 1 and 2. After forwarding each Interest, the strategy module calls **ProbingDueue** to check whether probing is needed. Two thresholds are introduced to further limit the probing overhead, For each FIB entry, M is the minimum time interval, and N is the minimum number of packets forwarded between two consecutive probings. The algorithm returns true only if at least M time has elapsed or at least N packets have been forwarded since last probing. Actual numbers of M and N depends on the traffic load as well as the probing overhead network operators are willing tolerate. Pseudocode 2 describes the probing algorithm. It sends a copy of the Interest to the routing preferred interface using a different nonce. The nonce is changed so that routers can distinguish between probing Interests and Interests that looped back.

C. Improving routing stability and scalability

Link-state routing protocols exhibit poor stability and scalability in IP due to the fast routing convergence requirement. However, there is a simple method to address these issues in NDN. Since NDN routers can handle network failures at the forwarding plane, we can actually mask the short-lived failures from the routing protocols. Research shows that the duration of network failures follows a long-tail distribution, and over 50% of failures last less than one minute ([8], [9]). Therefore, the number of routing events can be significantly reduced if routing protocols do not need to react to the short-lived failures. As a result, the bandwidth and CPU cycles consumed by routing updates can be reduced, and there will be less routing fluctuation. In addition, since there is no fast routing convergence requirement, larger networks and more name prefixes become affordable. Therefore, routing overhead can be greatly reduced, both routing stability and scalability can be significantly improved.

For link-state routing, we can implement the idea by increasing the HELLO and DEAD interval. For example, if we set the DEAD interval to be one minute, over 50% of the link failures will not be detected by the routing protocol. Alternatively, we can increase the routing update generation and SPF computation timers to achieve the same effect. Although this idea looks simple, it can be applied to any existing IP routing protocol to improve its stability and scalability. We will evaluate the effectiveness of this method in the next section.

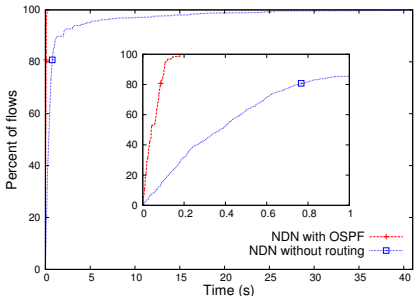


Figure 2. CDF of time to find working paths with and without routing.

V. EVALUATION

In this section we use extensive simulations to evaluate how NDN's routing and forwarding plane benefit from each other. Results show that by masking short-lived failures from routing, its stability and scalability can be significantly improved.

A. Simulation Setup

We simulate both NDN and IP in the QualNet simulator [10], which provides complete implementations of OSPF and RIP. We implemented basic NDN operations and the forwarding strategy presented in [2]. We also make necessary changes to the routing protocols as described in IV-A to support NDN.

We use the Abilene topology [11] and selected Rocketfuel topologies [12] in the evaluation. A summary of the topologies is presented in Table I. We process the first three topologies to remove all single-homed nodes. This is because if links of single-homed nodes fail, these nodes will be disconnected from the network and therefore cannot provide any useful result. For OSPF, we use propagation delay as the cost of the links. Unless otherwise specified, we report results from the AS1239-PoP topology. Results for other topologies are similar and lead to the same conclusions. The AS1239-Router topology is only used to show the improvement of routing scalability.

For each topology, we generate random link failures as follows. We use a shifted Pareto distribution to generate time-to-failure and time-to-recover distributions for each link independently [13]. We use 120 seconds as the mean-time-to-recover, and 1000 seconds as the mean-time-to-fail. We also tune the parameters so that 50% of the failures last less than one minute [8]. When a link fails, both directions of the link stop working. With this model, multiple network events (failures and recovery) can happen concurrently.

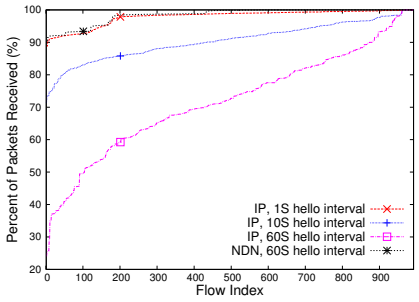


Figure 3. Packet delivery performance in IP under different HELLO interval.

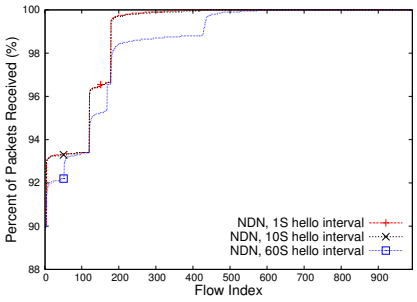


Figure 4. Packet delivery performance in NDN under different HELLO interval.

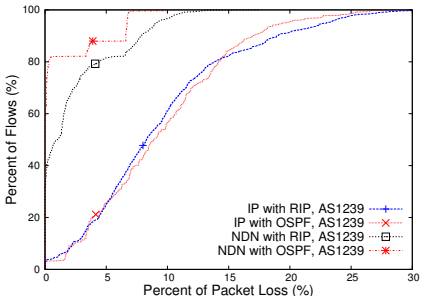
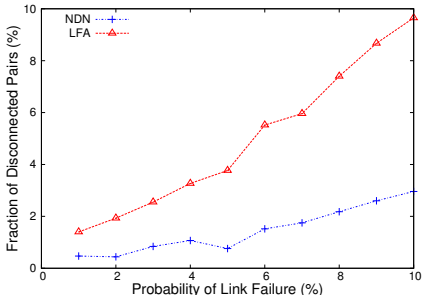
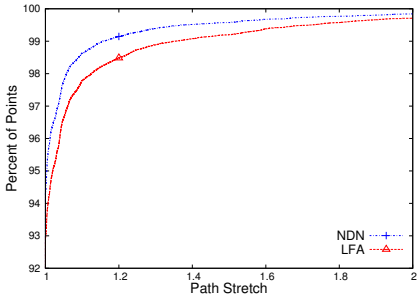


Figure 5. CDF of packet loss rate under different routing protocols.



(a) Fraction of disconnected pairs.



(b) CDF of path stretch.

Figure 6. Comparison between NDN and IPFRR.

B. NDN without Routing

In this experiment we show how NDN forwarding works without routing. Since routers have no idea how to rank the interfaces without input from routing, we implement a forwarding strategy that prefers Green interfaces over Yellow ones, but randomly picks a Yellow interface if no Green interface exists. If Data is brought back from an interface, the interface will be marked Green and used to forward subsequent Interests.

In each run of the experiment we pick one node as the consumer and another as the content provider. Assuming the consumer keeps retransmitting Interests until Data is received, we measure how long it takes. We enumerate all combinations of consumers and providers and draw the CDF in Figure 2. In 89% of the cases, the consumer retrieves Data within one second. However, it can take up to 40 seconds to find a working path in some rare cases. The situation can get worse as the network becomes larger. In contrast, Data retrieval always follows the best paths when routing protocol can provide interface ranking. Therefore, although NDN's forwarding plane can find working paths on its own, it benefits from a routing protocol to provide interface ranking to make the local search more effective.

C. Handling Link Failures

In this experiment, we study the impact of HELLO interval on packet delivery performance. We inject random link failures into the network as described in Section V-A. In order to measure packet delivery performance in NDN and IP, we run simple applications among all pairs of nodes in the network. For NDN, each node announces a distinct name prefix and provides content under this prefix. Each node also acts as a consumer requesting data from all other nodes. A consumer sends one Interest towards each name prefix every second. If Data is not received, a consumer will retransmit the Interest every second up to twice. Different consumers request different pieces of Data from the same name prefix so that they do not affect each other. Caching is also disabled so that we can focus on routing and forwarding behaviors[4]. For IP, each node acts as both client and server. Each client sends one UDP request to each server every second[5]. The server responds with UDP packet carrying the content. Similar to NDN consumers, these clients also retransmit requests if replies are not received. The sizes of the UDP packets are the same as those in NDN.

Figure 3 and 4 present the packet delivery rate for each node pair in IP and NDN under different HELLO interval settings. Figure 3 shows that HELLO interval has a huge impact on the packet delivery performance in IP. The shorter HELLO interval, the faster packet delivery can be resumed. The median packet delivery rate of IP is 99%, 91% and 72% when the HELLO interval is 1S, 10S and 60S respectively.

[4]If consumers request the same content and caching is enabled, NDN would perform even better.

[5]The packet rate is much lower than real Internet traffic due to performance limitation of the simulator. In fact, the IP packet delivery performance will be worse if the packet rate is higher.

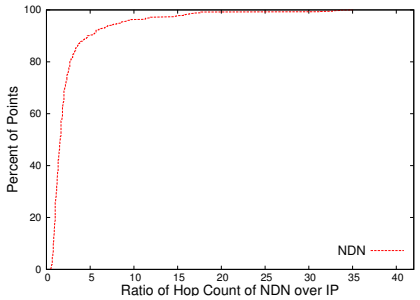


Figure 7. CDF of ratio of hop count of NDN over IP when prefix is unreachable.

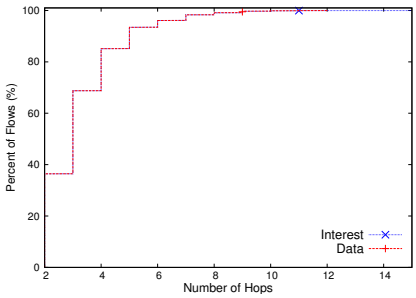


Figure 8. CDF of number of hops that probing Interests and Data traverse.

Figure 3 also shows that NDN with 60S HELLO interval even works slightly better than IP with 1S HELLO interval.

Figure 4 shows the impact of HELLO interval on the packet delivery rate of NDN. When the HELLO interval increases from 1S to 10S, the performance degradation is negligible. When the HELLO interval increases from 10S to 60S, the packet delivery rate decreases slightly. This is because only two consumer retransmissions are allowed. The packet delivery performance can be further improved by allowing more consumer retransmissions. Overall, the HELLO interval has little impact on the packet delivery performance in NDN.

We also evaluate the packet delivery performance under different routing protocols. Figure 5 shows the CDF of packet loss rate of NDN and IP when OSPF and RIP are used. Although RIP is generally considered to have poor routing convergence properties, it performs quite well with NDN. NDN with RIP performs much better than IP with OSPF or RIP. The performance difference between OSPF and RIP in NDN is due to the difference in interface ranking. Recall that RIP may not provide hop count for all interfaces, thus OSPF is able to provide better interface ranking.

D. Comparison with IPFRR

In the previous section we evaluate the packet deliver performance of plain IP, which totally relies on routing to handle network failures. However, today's ISP networks often adopt solutions that handle network failures without routing convergence, e.g., IPFRR. In this experiment, we compare NDN against Loop-Free Alternate (LFA) [14], the only commercially available IPFRR solution. We implement LFA in a custom simulator, and repeat the link failure experiment in [2]. In each run of the experiment, we associate each link with a probability of failure, and randomly generate link failures. We run each experiment 1000 times and report the average result.

Figure 6(a) shows the fraction of disconnected pairs under different failure probability. It shows that NDN is always able to recover much more failure scenarios than LFA. Figure 6(b) shows the CDF of stretch of alternative paths found by NDN and LFA. The 98-percentile of path stretch for NDN and LFA is 1.06 and 1.13 respectively. In conclusion, NDN is able to cover more failure scenarios and find better alternative paths than LFA.

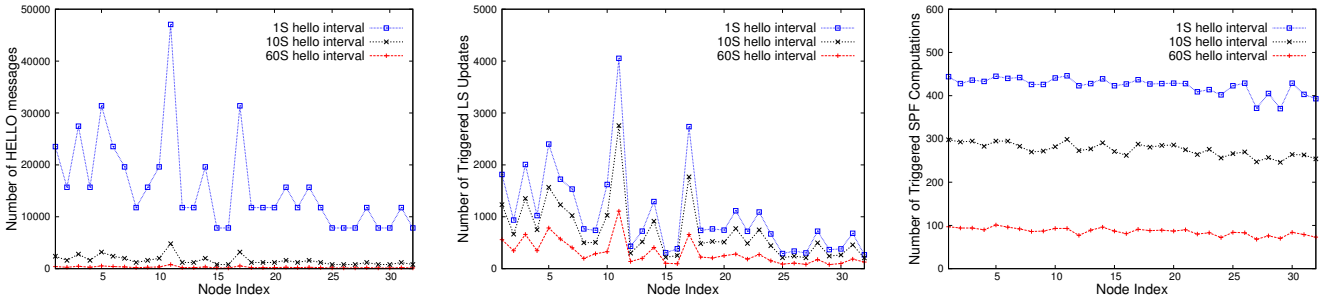
E. Prefix Unreachable

Previous experiments show that NDN performs well in handling link failures. When a node fails, however, the name prefix served by the node may become unreachable. In such cases, path exploration may lead to extra Interests all over the network. In this experiment we evaluate NDN's exploration overhead when a name prefix becomes unreachable. In each run of the experiment we fail one node and let all other nodes request content from this node before routing convergence[6]. Both NDN and IP applications will retransmit the same request twice. For each flow, we count the number of hops that each packet traverses in both NDN and IP, and compute the ratio of hop count of NDN over IP. We run the experiment for every node failure scenario and present the CDF of the ratio in Figure 7.

In IP, retransmitted requests will be sent to the same paths, whereas in NDN, retransmitted Interests may trigger path exploration, leading to large overhead. Surprisingly, NDN incurs less overhead than IP in 26% of the cases. This is because retransmitted Interests do not always trigger path exploration in NDN. If a node has already explored all its interfaces, a further retransmission will only get a NACK back to the application without being further forwarded. In contrast, IP routers will always forward the packets before routing convergence. The ratio is smaller than 5 in 93% of the cases. Only in some rare cases does NDN cause excessively high exploration overhead.

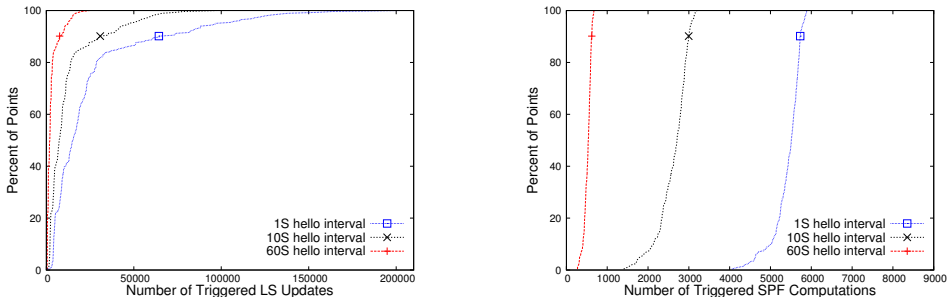
The exploration overhead becomes significant when popular content becomes unreachable, as many consumers will be requesting the content and their Interests will trigger many attempts by routers to find working paths. But on the other hand, popular content is usually hosted and served by multiple servers placed at different locations. In addition, popular content is more likely to be cached by routers. Thus its chance of becoming unreachable is slim. The overall impact in large scale networks needs further investigation.

[6]After routing converges, routers will learn about the failure and stop forwarding the requests.



(a) Number of HELLO messages. (b) Number of triggered LS updates. (c) Number of triggered SPF Calculations.

Figure 9. Routing overhead under different HELLO intervals in AS1239 PoP-level topology.



(a) CDF of number of triggered LS updates. (b) CDF of number of triggered SPF computations.

Figure 10. Routing overhead under different HELLO intervals in AS1239 router-level topology.

F. Probing Overhead

We evaluate probing overhead in this experiment. In each run of the experiment, we fail one link and run applications to let routers find working paths. Then we bring the link back up again, and run applications after routing convergence to measure the number of hops that probing Interests and Data traverse. Interest NACKs are counted as probing Interests. Applications are only run between node pairs whose traffic is affected by the failure. We run the experiment on all link failure scenarios and report the CDF in Figure 8. In 36% of the cases, probing Interests and Data only traverse 2 hops; they traverse no more than 6 hops in 94% of the cases. Probing Interests traverse more hops than Data in some rare cases, because a probing Interest does not necessarily bring Data back, and some of them may loop back to previously visited nodes and trigger NACKs. This experiment shows that by taking advantage of routing, probing only incurs very small overhead.

G. Routing Overhead

In this experiment, we evaluate the routing overhead of OSPF under different HELLO and DEAD interval settings. Specifically, we measure the number of HELLO messages, link-state (LS) updates and SPF computations for each node. HELLO and LS update messages constitute the majority of routing messages triggered by failures and recovery. We set the HELLO interval to be 1S, 10S and 60S; the DEAD interval is always four times the HELLO interval. Random link failures are injected into the network as described in Section V-A, and each experiment is run for 3000 seconds. Only LS updates and SPF computations triggered by failures and recovery are counted[7]. The numbers obtained in this experiment are the same for both NDN and IP.

Figure 9(a) shows the number of HELLO messages sent by each node under different HELLO interval settings in AS1239-PoP topology. As the HELLO interval increases from 1 second to 60 seconds, the number of HELLO messages sent by each node is decreased by 98% as one would expect. Figure 9(b) and 9(c) present the number of triggered LS updates and SPF computations for each node. As the HELLO intervals increase, less failure events will be detected by OSPF. No routing update will be generated and propagated for the undetected failures, and thus no SPF computation will be performed. If we increase the HELLO interval from 1 second to 60 seconds, the number of LS updates is decreased by 52% to 80%, and the number of SPF computations is decreased by 77% to 82%. Therefore, we can effectively reduce the overhead caused by HELLO messages, LS updates and SPF computation by increasing the HELLO interval.

We run the same experiments in AS1239 router-level topology to illustrate how the method works in large ISP networks. The CDF of number of triggered LS updates and SPF computations are presented in Figure 10. The median numbers of LS updates and SPF computations are decreased

[7]Notice that OSPF also floods refresh link-state announcements periodically even in the absence of network event. These refresh updates are not counted since they are not affected by routing convergence behaviors.

by 87% and 90% when HELLO interval increases from 1 second to 60 seconds. In conclusion, routing overhead can be significantly reduced by masking short-lived failures from the routing protocol. Since less LS updates are generated and propagated and less SPF computations are performed, routing becomes more stable and scalable.

VI. Discussion

Routing is a necessary subsystem for any large scale network. Like IP, NDN itself does not dictate what kinds of routing algorithms or protocols to use. However one can take advantage of NDN's adaptive forwarding plane to improve the stability and scalability of existing routing protocols, as well as enable routing protocols that deem difficult to adopt in IP networks.

Traditional Routing Protocols: As we have discussed in this paper, traditional routing protocols such as OSPF, RIP, and BGP can benefit greatly from NDN's adaptive forwarding plane. Since fast routing convergence is no longer a requirement, these routing protocols can be tuned for synchronizing among routers long-term topology and policy information without handling short-term churns. Routing assumes a supporting role to forwarding. It provides a reasonable starting point for forwarding which can then effectively explore different choices. Its job becomes more of disseminating topology and policy information than distributed computation of best paths. This new division of labor between routing and forwarding makes routing protocols simpler and more scalable.

Centralized Routing: Routing protocols have been designed to operate in a distributed manner to avoid single point of failure [15]. However with the increasing complexity in network management, Software-Defined Networking (SDN) has emerged to enable centralized management and control of networks, including logically centralized routing scheme. It is much easier to change the routing configurations on a central controller than on all participating routers, and to implement sophisticated traffic engineering schemes at the controller than individual routers. Routing overhead can also be greatly reduced, since routing updates only need to be sent to the controller instead of being flooded to the entire network, and only the controller needs to perform SPF computations.

However, a centralized routing scheme is also associated with several disadvantages, e.g., single point of failure and potentially longer convergence delay. One can mitigate single point of failure by physical replication of the central controller, which adds both the cost and complexity. A biggest concern is potentially prolonged convergence delay, which includes failure detection at local router, report to the controller, route recompilation at the controller, and dissemination of new routes to individual routers. NDN's adaptive forwarding removes the demands on convergence delay. As we have shown, NDN routers can adapt to network changes without waiting for routing to converge, making centralized routing feasible.

Coordinate-based Routing: In coordinate-based routing, instead of disseminate the network topology to routers, the coordinates of nodes are disseminated. The main characteristics of the network topology are embedded in the coordinates. Routers do greedy routing based on coordinates, i.e., forward packets to the neighbor whose distance (computed using coordinates) to the destination is the shortest among all neighbors. One example of such routing scheme is hyperbolic routing [16]. The advantages of this routing scheme include smaller routing tables (i.e., only need to know the destination's coordinates and neighbor routers' coordinates) and minimal routing updates (i.e., link failures and recovery do not affect a node's coordinates). However, in IP networks, this routing scheme is not guaranteed to be able to deliver packets. It is possible that the forwarding process runs into a local minimal, where all neighbors are farther to the destination than the current router. Path stretch may also get large. NDN's adaptive forwarding can fix these problems and make this routing scheme a possibility.

VII. Related Work

Francois et al. show that sub-second link-state routing convergence in large intra-domain networks is achievable by tuning various timers [5]. But their method incurs extra routing overhead and may cause routing instability. Fast reroute (FRR) mechanisms handle link failures by pre-computing alternative paths. MPLS FRR mechanisms are proposed to provide backup paths in MPLS-enabled networks to protect specific link failures. Similarly, IPFRR mechanisms ([14], [17], [18], [19], [13]) provide temporary alternative paths before routing convergence in pure IP networks. However, it is hard for the FRR mechanisms to cover all possible failure scenarios. In addition, they cannot handle multiple link failures well. Therefore, the FRR mechanisms still require fast routing convergence.

Path splicing [20] is an end-to-end multipath solution that provides link recovery controlled by end hosts. Each router provides multiple routing tables and let end hosts specify which one to use at each router. Path splicing may take long time to find alternative paths, and sometimes may not be able to find them even if they exist. Therefore fast routing convergence is still required. Multiple routing configuration (MRC) [21] also uses multiple routing configurations to handle network failures. Different from path splicing, MRC lets routers switch configuration when failures are detected. However, MRC may not handle multiple concurrent failures well. Since each router needs to maintain multiple routing tables, the computation overhead during routing convergence increases with the number of routing tables.

There are also solutions that carry routing/forwarding information in packet headers. Failure carrying packets (FCP) [22] puts failure information into the packet headers,

and let routers recompute the routing tables on-the-fly upon receipt of FCP. However, the method increases computation overhead, and the sizes of FCP headers may become arbitrarily large. Packet Re-cycling (PR) [23] reroutes packets along pre-computed backup paths in case of failures. In addition to the ordinary routing table, each router also creates a cycle following table, the generation of which is an NP-hard problem. When failures are detected, PR bits are set in packet headers to guide packet forwarding. Liu et al. propose Data-Driven Connectivity (DDC) [24] to ensure forwarding connectivity at the data plane. DDC organizes the network as a destination-oriented directed acyclic graph (DAG) to avoid loops, and uses two bits in the packet header to notify link reversal. DDC has its own control plane algorithm, therefore cannot make use of existing routing protocols.

VIII. Conclusion

In this paper we study the role of routing in NDN. NDN's adaptive forwarding plane leads to a new division of labor between routing and forwarding planes. While the latter can detect and recover from link failures quickly independent from the former, the former helps bootstrap adaptive forwarding and handle link recovery. We specify how NDN routing coordinates with forwarding through interface ranking and probing mechanisms. Our analysis and extensive simulations show that NDN routing protocols can benefit from the forwarding plane due to the relaxed requirement on timely detection of failures and convergence delay. Consequently NDN routing stability and scalability can be greatly improved. Moreover, the adaptive forwarding plane also enables new routing schemes that may not work well in IP to be used in NDN.

References

[1] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive Forwarding in Named Data Networking," *ACM SIGCOMM CCR*, vol. 42, no. 3, 2012.

[2] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A Case for Stateful Forwarding Plane," *Computer Communications: ICN Special Issue*, 2013.

[3] 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.

[4] C. Alaettinoglu, V. Jacobson, and H. Yu, "Towards Milli-Second IGP Convergence," Internet Draft draft-alaettinoglu-isis-convergence-00.txt, IETF, November 2000.

[5] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving Sub-Second IGP Convergence in Large IP Networks," *ACM SIGCOMM CCR*, vol. 35, no. 3, 2005.

[6] N. Kushman, S. Kandula, D. Katabi, and B. Maggs, "R-BGP: Staying Connected in a Connected World," in *Proceedings of NSDI*, 2007.

[7] "CCNx," http://www.ccnx.org/.

[8] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of Failures in an Operational IP Backbone Network," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, 2008.

[9] D. Turner, K. Levchenko, S. Savage, and A. C. Snoeren, "A Comparison of Syslog and IS-IS for Network Failure Analysis," in *Proceedings of IMC*, 2013.

[10] "QualNet Network Simulator," http://web.scalable-networks.com/content/qualnet/.

[11] "Yin Zhang's Abilene TM." [Online]. Available: http://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[12] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 2–16, 2004.

[13] S. Lee, Y. Yu, S. Nelakuditi, Z. li Zhang, and C. nee Chuah, "Proactive vs Reactive Approaches to Failure Resilient Routing," in *Proceedings of IEEE INFOCOM*, 2004.

[14] A. Atlas and A. Zinin, "RFC 5286: Basic Specification for IP Fast Reroute: Loop-Free Alternates," 2008. [Online]. Available: www.ietf.org/rfc/rfc5286.txt

[15] P. Baran, "On Distributed Communications Networks," *IEEE Transactions on Communications Systems*, March 1964.

[16] F. Papadopoulos, D. Krioukov, M. Bogua, and A. Vahdat, "Greedy Forwarding in Dynamic Scale-Free Networks Embedded in Hyperbolic Metric Spaces," in *Proceedings of IEEE INFOCOM*, 2010.

[17] A. Atlas, "U-turn Alternates for IP/LDP Fast-Reroute," draft-atlas-ip-local-protect-uturn-03, 2006. [Online]. Available: http://tools.ietf.org/html/draft-atlas-ip-local-protect-uturn-03

[18] S. P. S. Bryant, C. Filsfils and M. Shand, "IP Fast Reroute using Tunnels," draft-bryant-ipfrr-tunnels-02, 2005. [Online]. Available: http://tools.ietf.org/html/draft-bryant-ipfrr-tunnels-02

[19] S. P. S. Bryant and M. Shand, "A Framework for IP and MPLS Fast Reroute Using Not-via Addresses," draft-ietf-rtgwg-ipfrr-notvia-addresses-11. [Online]. Available: http://tools.ietf.org/html/draft-ietf-rtgwg-ipfrr-notvia-addresses-11

[20] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path Splicing," in *Proceedings of ACM SIGCOMM*, 2008.

[21] A. Kvalbein, A. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP Network Recovery Using Multiple Routing Configurations," in *Proceedings of INFOCOM*, 2006.

[22] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving Convergence-Free Routing using Failure-Carrying Packets," in *Proceedings of ACM SIGCOMM*, 2007.

[23] S. S. Lor, R. Landa, and M. Rio, "Packet Re-cycling: Eliminating Packet Losses Due to Network Failures," in *Proceedings of HotNets*, 2010.

[24] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring Connectivity via Data Plane Mechanisms," in *Proceedings of NSDI*, 2013.