MUCA: New Routing for Named Data Networking

Chavoosh Ghasemi[1], Hamed Yousefi[2], Kang G. Shin[2], and Beichuan Zhang[1]
[1]Department of Computer Science, The University of Arizona
[2]Department of Electrical Engineering and Computer Science, The University of Michigan

***Abstract*—*Named Data Networking* (NDN) is a fundamental paradigm shift from host-centric to data-centric Internet architecture. Among its numerous benefits, in-network caching and multipath forwarding are two prominent features that can significantly improve the performance and resiliency of networks and applications. The current NDN routing protocols, however, still focus on the traditional problem of forwarding content requests to content producers, without explicit or efficient support of in-network caching and multipath forwarding, which will limit NDN's potential and benefits to applications.**

In this paper, we propose a new intra-domain name-based routing protocol to provide simple and scalable support for MUltipath forwarding and in-network CAching (MUCA). While MUCA collects the network topology and computes the shortest paths to content producers in the same fashion as link-state routing protocols, it also learns multiple alternative paths from neighboring routers similar to distance-vector routing protocols. Moreover, by labeling each route update at the entry point into a network, internal routers select the same border router for the same name prefix, which enhances the hit ratio of cached contents. Our in-depth simulations demonstrate MUCA's effectiveness in reducing content retrieval delay and improving network resiliency while lowering the routing protocol overhead.

I. Introduction

Named Data Networking (NDN) is a clean-slate future Internet architecture and also an important representative of Information Centric Networking (ICN) [22], [23]. In NDN, content is identified by a hierarchical name, and both the requests (i.e., *Interests*) and the responses (i.e., *Data*) carry the content name rather than a source/destination address. Among the various benefits of the NDN architecture, in-network caching and multipath forwarding are two major features that can significantly improve network performance and resiliency. Since each network packet carries a unique name that identifies its content, intermediate routers can cache Data in its returning path to the requester(s) to serve future network Interests, i.e., NDN enables native in-network caching. Moreover, as pointed in [21], NDN's forwarding plane can detect routing loops by itself and choose a different next-hop if loop happens. This allows NDN routers to make use of multiple next-hops and adapt the choice based on content retrieval performance.

Full realization of the potential of in-network caching and multipath forwarding needs support from the underlying routing protocol. (Note that the routing plane in NDN is decoupled from the forwarding plane as discussed in Section II.A.) At the heart of NDN, the forwarding engine needs a routing protocol to efficiently compute and install proper forwarding entries in order to forward Interests towards the corresponding content provider(s). While some studies have been done on NDN routing protocols [10], [17] or similar content-centric routing protocols [8], [9], they all focus on the traditional problem of computing the shortest path towards a content producer, without explicit or efficient support of in-network caching and multipath forwarding.

To increase cache hit ratio in network caches, requests for the same content but generated by different consumers should merge as early as possible in the network, i.e., their forwarding paths merge before they reach the content producer. Traditional shortest-path computation does not take this into consideration, thus the result is opportunistic for caching. To alleviate this deficiency, MUCA labels each routing announcement/update in border routers such that all the internal routers will select the same border router for the same name prefix. This guarantees that requests for the same content will always merge before they go out of the network while improving the chance of their merge even before reaching the border router. Not only does this feature increase the efficiency of in-network caching, but also reduces the transport cost incurred by traffic between networks.

To take advantage of the NDN's capability of multipath forwarding, the routing protocol is expected to provide the forwarding plane with multiple next-hops for each name prefix. Traditional routing protocols only compute a single best path. NLSR [17], a name-based link state routing protocol currently used in the NDN testbed, computes a list of ranked next-hops by running Dijkstra algorithm in a router for each of its active interfaces, which can incur significant computational overhead, especially for routers with high connectivity. To address this issue, MUCA—as a link-state routing protocol—borrows a distance-vector mechanism, retrieving routing tables from neighboring routers instead of computing them. From these retrieved routing tables, one can easily figure out the ranked list of possible next-hops and save CPU cycles for the local router.

In addition to explicit support of caching and efficient support for multipath, MUCA employs a simpler mechanism than NLSR to propagate incremental routing updates. NLSR treats the routing update propagation problem as a data synchronization problem, and adopts ChronoSync [25] to synchronize the link state database (LSDB) between neighboring routers. MUCA simply notifies the neighbor router that a routing update is available, and expects the neighbors to retrieve this incremental update. Thus, it effectively reduces the routing overheads.

We have conducted extensive simulations to demonstrate

ISBN 978-3-903176-08-9 ©2018 IFIP

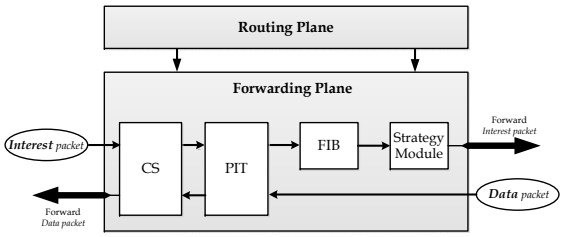


Fig. 1: Forwarding and routing planes in an NDN router

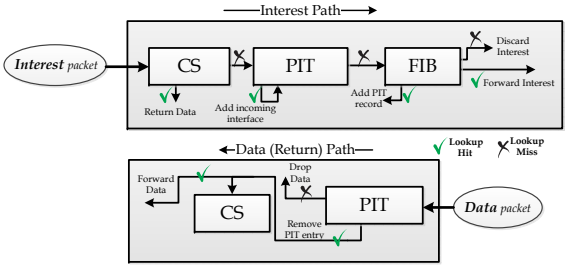


Fig. 2: Interest/Data processing in the forwarding plane

MUCA's benefits for multipath routing, in-network caching, and LSDB synchronization. Compared to the last version of NLSR, as the current *de facto* routing protocol of NDN testbed, MUCA enables multipath routing with 94% less traffic overhead, 26% faster content retrieval (by explicit support of in-network caching), 27% less overall cache space usage, and 22% less engaged routers for caching a specific content. At the same time, MUCA achieves fast reaction to failures due to quick routing update propagation and multipath support.

The remainder of this paper is organized as follows. Section II describes how routing and forwarding planes are decoupled in NDN and motivates our study. The design and operation of MUCA are detailed in Section III. The simulation results are presented in Section IV. Section V discusses the related work, and finally, Section VI concludes the paper.

II. Background and Motivation

A. Routing vs. Forwarding

IP forwarding plane is neither adaptive nor intelligent, as it strictly follows the routing plane [21]. NDN implies a substantial re-engineering of the forwarding plane and changes the role of routing plane from a directive to a consultant [10]. Thus, the routing plane is a second-class citizen in NDN. Actually, the routing plane only computes the route(s) towards each producer and provides the forwarding plane with this information. Instead, a forwarding strategy module is responsible for controlling all forwarding decisions (i.e., whether, where, and when to forward an Interest). Thus, unlike in IP, the forwarding table is not under control of the routing protocol and continuously updated according to forwarding plane performance measurements and administrative policies. This intelligent and adaptive forwarding plane enables exploring more radical and scalable routing schemes that are not possible in IP networks. Fig. 1 shows the forwarding and routing planes in an NDN router. It is worth noting that our main focus in this paper is on the routing plane. The way the paths are used by the strategy module depends on administrative decisions and the adopted forwarding strategies, which is beyond the scope of this paper.

In the NDN's request-driven (pull-based) communication model, a node requests a named content using an *Interest* packet. The intermediate neighbor nodes remember the interface from which this packet was received, and forward it by consulting their forwarding tables. Any node receiving the Interest packet and having the requested content simply responds with the corresponding *Data* packet. Unlike IP-based networks, not all of the packets need to be routed in NDN. Only the Interest packets are routed and the corresponding Data packets are returned based on the state information set up by the Interest packets in intermediate nodes (symmetric Interest-Data exchange).

As shown in Fig. 1, along with the strategy module, NDN has three main tables in its forwarding plane [23]: (1) CS (Content Store), a cache memory, that stores previously retrieved Data packets, (2) PIT (Pending Interest Table) that stores unsatisfied Interests as well as the interfaces through which they have been received, and (3) FIB (Forwarding Information Base) that serves as the forwarding table to direct the Interests towards the potential provider(s) of matching Data. Fig. 2 shows how Interest/Data packets are processed in the NDN tables in both sending and returning paths. Upon arrival of an Interest packet at a router, CS is searched for the requested name. If the desired content is found, then a Data packet is returned else PIT is searched. If the name matches a PIT entry, meaning that an Interest for this data has already been forwarded upstream, we just add the incoming interface to the related entry in PIT and wait for the response Data. Otherwise, after assigning a new entry to the requested name in PIT, the Interest is forwarded to the next hop(s) based on the forwarding strategy looking at FIB. If multiple next-hops exist in a FIB entry, the forwarding strategy determines how to use the multiple routes for forwarding Interests. On the return path, if the desired content arrives after its expiration time, it will be discarded; otherwise, after caching the content in CS, it will be sent through the interfaces listed in the matching PIT entry.

In-network caching is a gift from CS. Thanks to this built-in opportunity, all the NDN routers can act as temporary providers, thus unneeding to traverse the entire network for a desired content. Multipath support is a gift from PIT. Actually, PIT makes the NDN forwarding plane stateful, thus ensuring loop-free forwarding. This brings the opportunity of sending an Interest out through multiple interfaces in each router.

B. Motivation

In-network caching and multipath forwarding support are two prominent features of NDN. However, state-of-the-art studies still focus on the traditional problem of forwarding content requests to content producers, without explicit or efficient support of these two built-in opportunities in NDN,

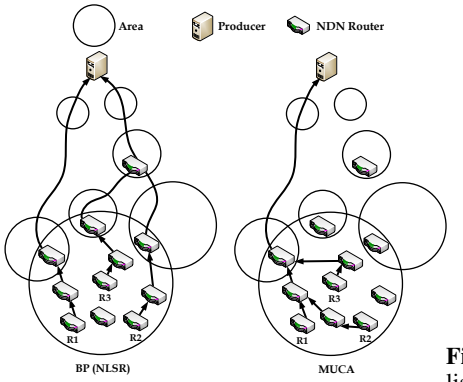


Fig. 3: MUCA merges the forwarding paths for the same content as early as possible (in the worst case, at the same border router). (The areas simply follow the network partitioning in OSPF.)

NLSR	MUCA
BP	MPP
BP_2	BP
BP_3	SBP_1
BP_4	SBP_2
BP_5	SBP_3
..	..
....	....

Fig. 4: MUCA vs. NLSR: a ranked list of routes (BP (Best Path), SBP (Semi Best Path), and MPP (Most Probable Path))

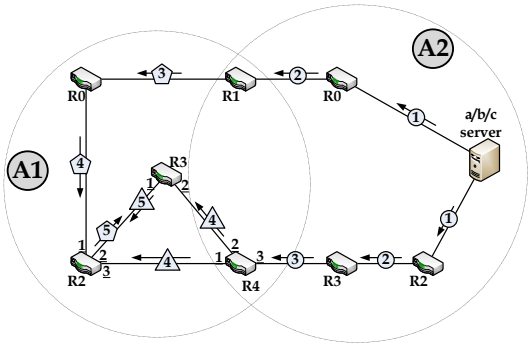


Fig. 5: Update dissemination after adding the `/a/b/c` server

which will limit the NDN's potential and benefits to applications. To alleviate this deficiency, while also improving scalability in terms of both computational and traffic overheads, we propose a new routing protocol which (1) mimics cache-awareness to the routing plane, and (2) computes multiple paths efficiently.

Caching: For any satisfied Interest, the response is cached in all nodes on its returning path to the requester. The story of routing in NDN is deficient without caring the caching capability in the routing plane. Simply enjoying this capability in the forwarding plane without explicitly exploiting in-network caching, as NLSR does, can degrade the performance in terms of both content retrieval delay and traffic (Interest/Data) overheads. Fig. 3 shows a simple example, where three nodes (R1, R2, and R3) request the same content. Ignoring the in-network caching capability in the state-of-the-art—which relies on the shortest paths—can cause forwarding the Interests through late, even never, merging paths (R3&R2 and R1&R2, respectively). Our idea is to label incoming route announcements and updates (advertised in the case of both topology and name prefix changes) at border routers such that all the internal routers will select the same border router for the same name prefix. This not only guarantees that requests for the same content will always merge before going out of the network (see R1&R2&R3's paths), but also improves the chance that they merge even before reaching the border router (see R1&R2's paths). Thus, MUCA equips the forwarding plane with a new path (referred to as *MPP*), explicitly exploiting in-network caching in NDN.

Multipath: Traditional routing protocols only compute a single best path. To implement multipath routing, NLSR provides a ranked list of all possible paths towards each producer. In this line, as a Link State (LS) routing protocol, it has no way except to run the Dijkstra algorithm from every single interface's point of view. This approach incurs high computational overhead, especially for routers with high connectivity. To address this issue, our general idea is to run Dijkstra only once in each router—this surely provides the best path (referred to as *BP*) towards each producer—and then ask for help from neighbors using their precomputed paths. This way, MUCA borrows a Distance Vector (DV) mechanism, realizing a cooperative routing for NDN. Thus, MUCA provides a list of alternative paths (referred to as SBPs) for each BP in much lower complexity in the price of a little more communication overhead.

In conclusion, as shown in Fig. 4, MUCA provides the forwarding plane with a new list of forwarding paths—including MPP, BP, and a dynamic list of SBPs as discussed later—while effectively utilizing caching and multipath in the routing plane.

III. MUCA Design

This section describes our design and its essential parts—multipath routing and LSDB synchronization. We simply follow the network partitioning in OSPF [1] and use a general hierarchical naming model, where each router takes a unique name `/<1-st Area ID>/<2-nd Area ID>/.../<n-th Area ID>/<Router ID>` in ascending order of the areas in which it resides. For example, in Fig. 5, internal router R0 in A1 and border router R4 in both A1 and A2 are simply named `1/0` and `1/2/4`, respectively.

A. Multipath Routing

MUCA comes with three different characterized paths: (1) Best Path (BP), provided by Link State (LS) face; (2) Semi Best Path(s) (SBPs), provided by Distance Vector (DV) face; and (3) Most Probable Path (MPP), provided by effectively exploiting the caching opportunity built in NDN. As the only characteristic in common, all of these three paths route the packets towards the content producers (i.e., original providers). While BP and SBPs ignore the in-network caching capability, MPP tries to meet the network caches as early as possible on its way to the producer.

To describe different paths, Fig. 5 shows an example in which a new `/a/b/c` server announces the content it serves (i.e., `/a/b/c`) by disseminating a routing update in the network. Each sequential step of dissemination is shown as a number next to each link. The shape changes to pentagon and

triangle when the update packet passes border routers `1/2/1` and `1/2/4`, respectively.

1) Scalable Multipath Support:

BP: It is the shortest path between every pair of routers, and used to deliver an Interest packet to its associated producer(s) with a minimum cost. BPs are calculated by the LS face, which runs Dijkstra algorithm in each router, having the global information of its area(s).

After synchronizing LSDBs in an area (see Section III.B), the MUCA's LS face determines BPs and builds the routing table (a.k.a. Routing Information Base (RIB)) for each router to all the others in that area. (For a router, its routing table, RIB, is populated with the costs to reach the other routers within its area.) Fig. 6 shows the RIBs of routers R2, R3, and R4 in area A1 in Fig. 5 after determining BPs, where all link costs are assumed to be 1 for simplicity. To differentiate the interfaces of a router, there is a number next to each link (as also shown in Fig. 5). *BP cost* and *BP interface* at each router are associated with the shortest-path cost and the interface through which the path reaches another router within the same area. For example, *BP interface* and *BP cost* in R3's RIB towards R0 in A1 are 1 and 2, respectively.

SBP: We propose using SBP as a supporting/alternative path for BP. We exploit the potential for cooperation between neighbors offered from the DV face of MUCA. Although DV is unsuitable for large wide-area networks, we borrow its main concept (i.e., querying only the neighbors) to use their "pre-computed" routing tables for finding the Semi Best Paths—called SBPs. It is worth noting that having the complete routing information of neighbors in a router, the number of SBPs towards a provider can be dynamically adjusted according to the administrative policies. Thus, unlike for BP (and MPP), MUCA can provide the forwarding engine with a ranked list of SBPs. However, for simplicity, we only present it as a single path in this paper. SBP can be exploited not only as a backup path for fault-tolerant routing but also for other purposes, such as load-balancing. However, how the strategy module uses this path is beyond the scope of this paper which focuses on the routing plane.

After determining BPs, the DV face resolves SBPs for each router to all the others within the area. To this end, each router sends a query to all of its neighbors and asks for their RIBs. This query is an Interest packet with name `</InterestSBP>`, and each router replies with its RIB (which includes its BP to each destination Y). Then, the SBP to Y passes through the BP of a directly connected neighbor providing the minimum cost. Note that SBP from a router X to a router Y needs to satisfy two conditions to ensure a loop-free routing: (1) SBP cannot go through the same X's interface as BP towards Y, and (2) BP to Y from the neighbor cannot cross X again. Fig. 6 shows the packets exchanged to resolve SBPs for R3. For example, as Figs. 6 and 7 show, R3 chooses R2 as its SBP next-hop to R4, while satisfying two conditions (1) BP and SBP to R4 go through different interfaces, and (2) R3 is not on R4's BP to R2. Moreover, the *SBP interface* and *SBP cost* in R3's RIB towards destination R4 are 1 and 2, respectively, since R2's BP to R4 has cost 1 and R3 reaches R2 through interface 1 just in one hop.

One may argue that as BP and SBP from a source router are not necessarily disjoint paths, our approach is not fault-tolerant enough. However, note that, unlike IP's end-to-end packet delivery model, the NDN forwarding policies are applied in a hop-by-hop manner. Thus, in the case of a link/node failure along the BP, its immediate neighbor quickly replaces the forwarding path by SBP (i.e., a part of BP is replaced by another path towards the producer). From the source router's point of view, the current forwarding path is not the best until it becomes aware of the failure and updates its RIB and FIB. Thus, the synchronization time (called the *convergence time* for fault-tolerant routing) is key to the performance of a multipath routing (see Section III.B).

Finally, note that SBP is equal to the NLSR's second best path (except in case that the second best path goes through the same interface as BP). Thus, it offers almost the same performance by incurring much lower computational overhead.

2) In-network Caching Support:

MPP: As mentioned earlier, MUCA also equips the forwarding plane with another path—called *Most Probable Path* (MPP)—through which an Interest will likely meet the desired content before reaching the producer(s). By using MPPs, we try to forward similar traffic (Interests) via the same (even partly) path to maximize utilizing built-in caching in NDN. To this end, we send all the requests which target the same name prefix through the same border router. We consider the case where routing announcements/updates are received in an area through multiple border routers. Then, the entry points (i.e., border routers) simply update a field *Modified Time* in receiving announcements/updates—this is referred to as *labeling process*. Finally, the internal routers choose the border router informed of a new name prefix before the others (i.e., that with the least *Modified Time*) and use their BPs (and SBPs) towards this border router to forward their similar requests. The same scenario applies sequentially in other areas. Finally, from a given router's point of view, there is a path (maybe longer than BP) which eventually reaches the producer(s), but with a higher probability of satisfying its request by an intermediate router.

For example, from the vantage point of router R3 in A1 in Fig. 5, its MPP to `/a/b/c` server goes through router `1/2/1` (R1). This is because the advertisement of this name prefix has been received by router `1/2/1` at the second hop, and propagated into the network (follow pentagon-shaped updates) before router `1/2/4` (follow triangle-shaped update) receiving the update at the third hop. Finally, BP and MPP towards `/a/b/c` server are [`1/3-1/2/4-2/3-2/2`] and [`1/3-1/2-1/0-1/2/1-2/0`], respectively. Note that the resolved MPP in a router towards `/a/b/c` server may be the same as BP or SBP (e.g., for routers `1/0` and `1/2`), or different (as described for router `1/3`). Thus, although MPPs may take longer paths than BPs, they can effectively reduce content retrieval delay. This is because MPPs from all internal routers for `/a/b/c` are directed to the same border router in

RIB of R2

Node name	BP cost	BP Interface	SBP cost	SBP Interface
1/0	1	1	-	-
1/3	1	2	2	3
1/2/4	1	3	2	2
1/2/1	2	1	-	-

RIB of R3

Node name	BP cost	BP Interface	SBP cost	SBP Interface
1/0	2	1	3	2
1/2	1	1	2	2
1/2/4	1	2	2	1
1/2/1	3	1	4	2

RIB of R4

Node name	BP cost	BP Interface	SBP cost	SBP Interface
1/0	2	1	3	2
1/2	1	1	2	2
1/3	1	2	2	1
1/2/1	3	1	4	2

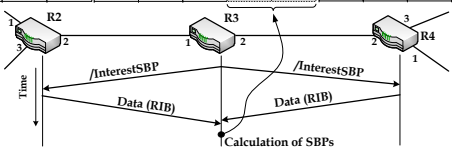


Fig. 6: RIB (routing table) update and SBP calculation by exploiting the distance-vector face of MUCA

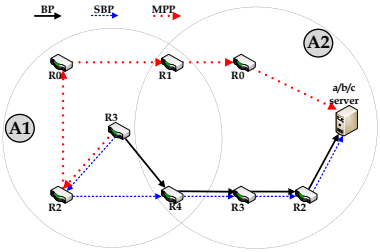


Fig. 7: BP, SBP, and MPP from R3 to retrieve `/a/b/c`

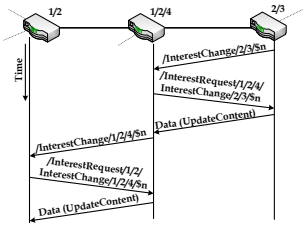


Fig. 8: Notifying the neighbor routers to synchronize their LSDBs

an area, which not only guarantees that the paths merge before going out of the area, but also improves their chance to merge even before reaching the border router.

To realize MPP, we define MUCA header, including three fields *Area ID*, *Border Router*, and *Modified Time* augmented with NDN Data header. These fields experience no change in the return path while passing internal routers. However, upon arrival at a border router, they are updated according to the current area, border router, and arrival time.

By presenting MPP, we prevent scattered caching and forwarding Interests through improper paths or towards deprecated copies of contents. Moreover, by avoiding sending several similar Interests throughout the network, MPPs reinforce the role of PIT, as one of the NDN main design principles. This way, we save more network bandwidth and decrease the possibility of congesting the intermediate links/routers, and reduce the transport cost incurred by traffic between networks. By receiving fewer Interest packets at the producers, the servers' load will also be reduced.

Note that MUCA can support multipath routing not only for a single content producer by leveraging BP, SBP, and MPP (Fig. 7 depicts BP, SBP, and MPP from router `1/3` to `/a/b/c` server), but also for multiple producers (in case there are more `/a/b/c` servers in the network).

B. LSDB Synchronization

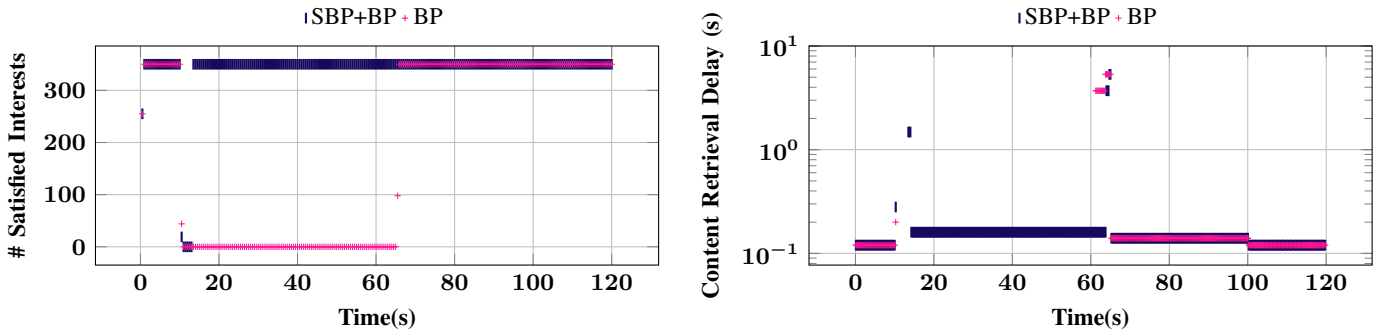
As part of any LS routing protocol, to synchronize all the LSDBs (Link State Databases), each router needs to detect a new update in the case of both topology and name prefix changes and disseminate it throughout the network. (The LSDB at each router contains information on reachability to both routers and name prefixes.) In this line, the latest version of NLSR [17] uses ChronoSync [25]. Considering the routing update propagation problem as a synchronization problem, two neighbors need to periodically inform each other about the state of their LSDBs (even if there is no changes in the network). Although there are some benefits with ChronoSync, especially in highly dynamic and unreliable scenarios, it is not a perfect fit to NDN routing protocol synchronization, so incurring high message overhead. To address this problem, MUCA suggests that each router simply notifies its neighbors that a routing update is available, and expects the neighbors to retrieve this incremental update. This mimics a push-based notification of the exact changes in heart of the NDN pull-based communication model, which effectively reduces not only the convergence time but also many unnecessary periodic control packets to detect the updates and difference between LSDBs, if any.

Implementing LSDB Synchronization: The currently adopted update model in NDN is request-driven, requiring all the routers to pull the updates. To implement our approach, in the case of any update in a node, it sends an *InterestChange* to its neighbors and they respond by returning *InterestRequest*s to pull new changes (as a Data packet *UpdateContent*). Upon receiving the changes at a neighbor, it updates its LSDB and notifies its neighbors. This procedure is repeated hop-by-hop until all the routers in the associated area are informed of any update. Names `/InterestChange/<origin router name>/<nonce>` and `/InterestRequest/<sender router name>/InterestChange/<origin router name>/<nonce>` are used for InterestChange and InterestRequest packets, respectively, where the suffix of InterestRequest is identical to its associated InterestChange. Here, *origin router name* and *sender router name* refer to the routers from which InterestChange and InterestRequest are transmitted, respectively, and *Nonce* is a random integer. Fig. 8 shows LSDB synchronization process for routers `2/3`, `1/2/4`, and `1/2` in Fig. 5, where $\$n$ is a *Nonce*. After receiving the routing update at router `1/2/4` from router `2/3`, it updates its LSDB and then sends InterestChange to its neighbor (i.e., router `1/2`). Upon receiving this packet, router `1/2` requests an update by returning an InterestRequest. Finally, router `1/2/4` generates an UpdateContent by which router `1/2` can update its LSDB and inform its neighbor of this new change.

IV. Evaluation

In this section, we evaluate the performance of MUCA via extensive simulations using ndnSIM, which is the de facto simulator of NDN. The results are compared to the last version of NLSR [17], as the current de facto routing protocol of NDN testbed, to demonstrate MUCA's benefits for multipath routing, LSDB synchronization, and in-network caching utilization. The simulation parameters are set to their default values in ndnSIM [3]. The topology generator aSHIIP v.3 [2] and the GLP model [4] are also employed to generate random networks. The results are the average of 10 runs.

Scattered caching may waste and improperly leverage cache capacity.



(a) (b)

Fig. 9: The benefits of multipath routing in a node failure scenario: (a) number of satisfied Interests and (b) content retrieval delay

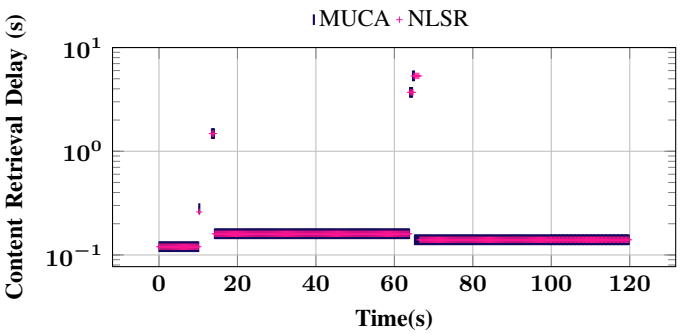


Fig. 10: Semi Best Path (SBP) in MUCA achieves the performance of second Best Path in NLSR with much lower routing overheads

A. Multipath Support

We answer three questions on the performance of our proposed multipath routing protocol exploiting LS and DV faces: (1) what are the benefits of employing multipath? (2) how MUCA and NLSR outperform each other during a failure scenario?, and (3) what is the cost of resolving SBPs in terms of traffic overhead?

To answer the first question, Fig. 9 compares two modes, BPs alone and BPs & SBPs together, in terms of number of satisfied Interests and consumer-perceived response time (a.k.a. content retrieval delay) in case of node failure. The retrieval delay is the time duration between sending an Interest and receiving its corresponding Data, including the time for retransmissions. We capture network events during a period of 120 seconds, where two randomly selected nodes act as the producer and consumer residing in two sides of a 100-node network partitioned into four areas. After computing and populating the LSDBs in all the routers, at the 10-th second, we brought down a node on the consumer's BP towards the producer. In MUCA with only a single path, the node failure triggers the Dijkstra algorithm to calculate the new BP and resolve the desired content. As shown in Fig. 9(a) and (b), it takes about 60 seconds to detect the node failure and then to converge (thus, using only BPs, no Data returns during this period). This figure clears the importance of using multipath, as using a single path (i.e., BP) drastically increases packet drop rate (see Fig. 9(a)) and content retrieval delay (see Fig. 9(b)), during convergence time. However, providing the forwarding plane with SBPs lets it forward Interests on another path almost immediately from the failed node's neighbor, until the network converges and the consumer updates its BP (around the 70-th second). Finally, at the 100-th second, when the failed node is recovered, the traffic is switched back on the old path. Fig. 9 thus illustrates the benefit of employing SBPs along with BP, and its vital effect on overall performance of the network.

To answer the second question, Fig. 10 compares MUCA with NLSR in the same scenario except that the failed node is not brought back up. Both MUCA and NLSR enable the forwarding plane to quickly switch to the alternative paths (SBP and the second best path, respectively) and continue transmission of packets with a larger retrieval delay. After detecting the node failure (around the 70-th second as mentioned earlier), both MUCA and NLSR daemons start propagating this change throughout the network and updating LSDB of the routers consequently. As evident from the figure, both protocols perform almost the same, though MUCA could converge a little faster due to its update propagation mechanism. After convergence time, both protocols switch to a new BP with a relatively longer delay than the initial one. In conclusion, using SBP, MUCA can achieve the performance of NLSR only by incurring little traffic overhead to the network, while drastically reducing the computational overhead of NLSR.

Finally, to answer the third question, Fig. 11 shows a complete analysis of traffic overhead of resolving SBPs. To give a broader view, we consider different sizes of the network, i.e., 50, 80, 100, and 150 nodes. (We attempted to cover medium to large networks (as the AT&T core network topology has 154 nodes [9].)) We also change parameter p in the GLP model to

This is long enough for warming up and testing the behavior of network in terms of convergence time.

As in OSPF, tracking the network connectivity is handled by sending Hello packets to neighbor nodes. Usually after hearing nothing from a neighbor within three continuous Hello packet interval, the neighbor is determined as dead.

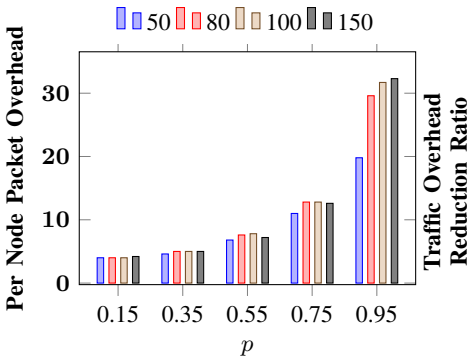


Fig. 11: Average "per-node" packet overhead to resolve SBPs for different network sizes (50, 80, 100, and 150 nodes) and node degrees (p)

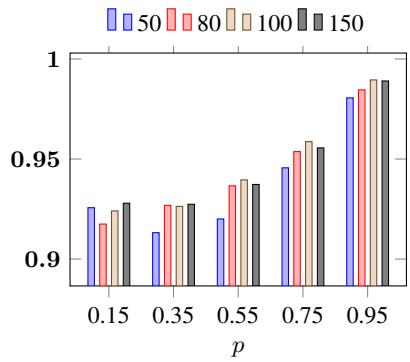


Fig. 12: MUCA vs. NLSR: MUCA effectively reduces traffic overhead by utilizing a new LSDB synchronization mechanism

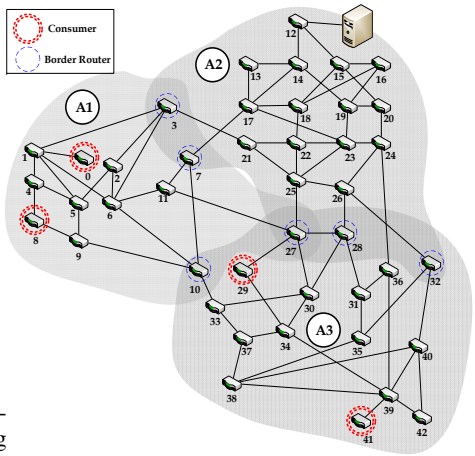


Fig. 13: Network topology to evaluate the MPP performance

create different node degrees, ranging from very sparse to very dense deployments, where p $(1-p)$ specifies the probability of adding a predefined number of new links (a new node) to the network at each time-step. The results show that this overhead is negligible. For example, when there are 80 routers in the network and the density is normal (i.e., $p = 0.5$, which means the network is neither fully mesh nor sparse), less than 10 packets per node is needed to resolve SBPs.

B. LSDB Synchronization

We now evaluate the performance of MUCA against NLSR in terms of synchronization overhead. In each scenario, a random router is informed of a new name prefix and updates its LSDB. Then, the number of transmitted packets (including all periodic and non-periodic ones) are measured. Fig. 12 shows the traffic overhead reduction ratio provided by MUCA to synchronize all the LSDBs in the network. MUCA is shown to significantly outperform NLSR, especially in larger networks. Moreover, by increasing p, the number of links grows, so more periodic packets will be exchanged by NLSR. This figure casts doubt on using ChronoSync (or similar algorithms) as it performs poorly compared with a straightforward approach. Instead, by blocking many unnecessary packets (e.g., those exchanged for finding differences between LSDBs), our approach reduces the traffic overhead on average by 94% in the network.

C. In-Network Caching Support

As mentioned earlier and also shown in Fig. 4, MUCA provides the forwarding plane with a new ranked list of the candidate paths, giving priority to MPPs. In this section, we compare the NDN forwarding based on MPP versus BP (NLSR approach) to see whether MUCA's ranking rubric outperforms NLSR. Moreover, to fully demonstrate the pros and cons of MPP-based forwarding, we also compare it with "flooding" (as the simplest de facto forwarding strategy in NDN). All three schemes follow regular caching of Data packets at intermediate nodes and use LFU as their replacement policy [14]. The performance of in-network caching is

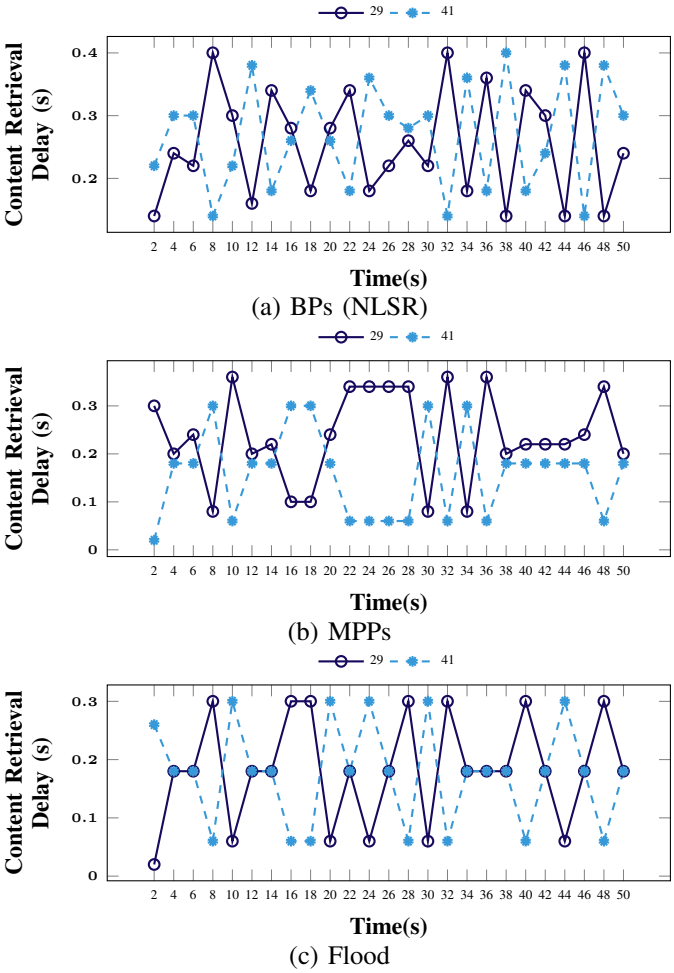


Fig. 14: Content retrieval delay in nodes 29 and 41 using different forwarding schemes

evaluated in terms of: (i) content retrieval delay, (ii) overall cache memory usage, and (iii) number of nodes engaged in caching.

Fig. 13 illustrates a network of 44 nodes partitioned into

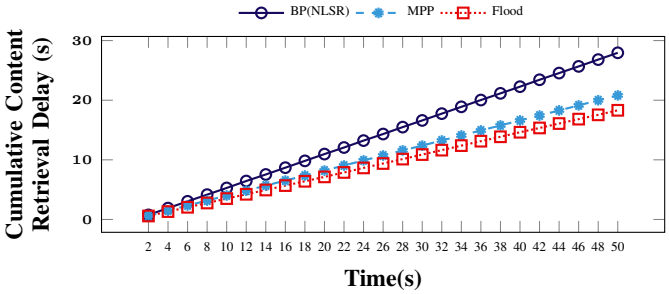


Fig. 15: Cumulative content retrieval delay using different forwarding schemes

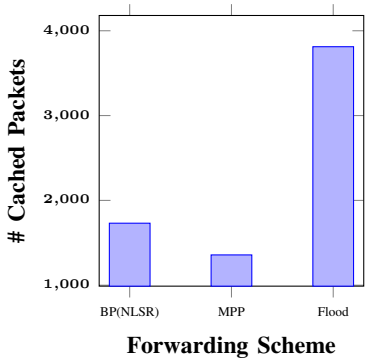


Fig. 16: Number of packets cached in the network using different forwarding schemes

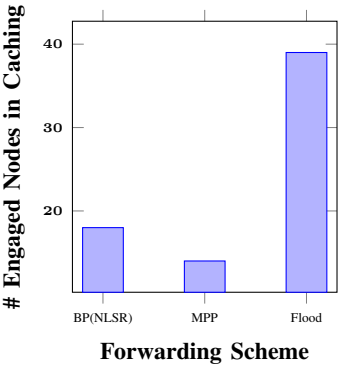


Fig. 17: Number of nodes engaged in caching using different forwarding schemes

three areas, where four consumers request 20 name prefixes served by one server (at the top). The location of each consumer is also denoted by double-dotted circles. In order to test the network operation, each time step is randomly chosen between 0 and 4 seconds, in which each of four consumers requests one of the prefixes. At the end of simulation that lasts for 50 seconds, each node has requested at least a half of the existing name prefixes. Thus, it is highly possible that a content is repeatedly requested by different consumers (time-locality principle). Although each content in CS is valid only for a short while in reality, we enjoy the caching opportunity built in NDN.

Fig. 14(a)-(c) shows the content retrieval delay for nodes 29 and 41 (two of four consumers) in A3 for all three forwarding schemes. This delay for the nodes under the MPP-based scheme is shown to be higher than that under the flooding scheme, while smaller than that under the BP-based one. Indeed, although leveraging only BPs can result in the same or even better performance under a special condition (when the content requisition does not follow the time locality principle), MPPs can, in general, reduce the retrieval delay. As evident from the figure, nodes 29 and 41 have relatively opposite retrieval delay trends (i.e., high vs. low). This verifies that by leveraging in-network caching, if a node needs a content which has already been requested, it will meet a cached version with high probability. For example, as evident from the 22-nd till the 28-th second in Fig. 14(b), node 41 retrieves a desired content very quickly as it has been already consumed by node 29. For the majority of time, the content can be retrieved in less than 0.2 second in Figs. 14(b) and (c), while this is reversed in Fig. 14(a).

Fig. 15 illustrates the cumulative retrieval delay for all consumers using different forwarding schemes. As evident from the figure, the MPP-based forwarding scheme performs close to flooding which is our lower bound—flooding minimizes the delay in the cost of incurring the maximum traffic overheads to the network. Actually, the MPP-based scheme outperforms the BP-based one on average by 26% in this scenario, while flooding reduces the retrieval delay on average by 34% and 12% compared to BP- and MPP-based schemes, respectively.

Fig. 16 shows the cumulative cache memory usage by each scheme in terms of the number of cached packets at the network nodes. As evident from the figure, the MPP-based scheme decreases the overall CS space usage on average by 27% and 64% compared with BP-based and flooding schemes, respectively. Besides, as shown in Fig. 17, forwarding the Interests over MPPs can reduce the number of nodes engaged in caching a specific content on average by 22% and 64% compared to BP-based and flooding schemes. Obviously, fewer engaged nodes mean less scattered caching, while also providing a lower retrieval delay by using MPP compared to the BP-based scheme. In general, scattering the content between more network nodes (as flooding does) results in a greater opportunity to reduce the retrieval delay. However, we did not follow this to avoid caching redundant contents. Finally, the MPP-based scheme consumes CS space very efficiently while also performing very close to flooding. Based on these observations, we can make a trade-off between the content retrieval delay and the CS space usage by choosing MPPs or flooding.

Knowing the benefits of using MPP over BP in different aspects, we can reject adopting the assumption of several studies (like NLSR) which imply using path cost as a metric to rank the available paths. Instead, we believe that the paths with higher probability to meet the cached content have higher priority over the shortest paths towards the provider(s) in NDN. That is why MUCA gives MPP a higher priority than BP and SBPs in its ranked list of paths as shown in Fig. 4.

V. Related Work

There exists several studies on routing in NDN. OSPFN [18], as an extended version of OSPF [1], is the first NDN routing protocol for rapid prototyping of name-based forwarding in the NDN testbed. However, it suffers from several critical drawbacks such as IP dependency, employing GRE tunnels, and disregarding multipath forwarding. The two-layer routing protocol in [6] uses OSPF to resolve topology and calculate the shortest-spanning trees. However, it relies on flooding for update dissemination and does not yet support multipath routing towards a single producer. To mitigate these problems and allow for new topology-discovery methods, a named-data link state routing protocol (NLSR) was proposed in [10], [17]. However, it—as the current de facto routing protocol of NDN

testbed—suffers from high computational and traffic overheads. A controller-based routing scheme (CRoS) for NDN was also proposed in [15], which uses multiple controllers to achieve scalability. However, it incurs high traffic overhead due to flooding of Interests to search for controllers. LSCR [9] proposed a name-based link-state routing protocol which aims to provide forwarding plane with permanent loop-free paths. DCR [8] is the first name-based content routing which does not require any information about physical topology and works solely based on distance information. However, both LSCR and DCR refuse to provide the network with information of all available providers, while none of them explicitly employ in-network caching capability, as well. Bloom Filters (BF) are used in [5], [11], [12], [20] to digest FIB and exchange information about content availability, but they incur high signaling overheads. The stateful BF is also used in [16], but it only leverages the passive mode of prefix announcements, thus flooding the network multiple times. Although BF can reduce the space complexity, it suffers from false positives (collisions) which, in turn, degrade performance. Besides, Wang *et al.* [19] and Zhang *et al.* [24] attempted to utilize in-network caching by adding new data structures or modifying the existing ones. However, they preserve the relationship with IP-based routing, which does not meet the NDN's goal of departing from IP [10].

There have also been other efforts [7], [13] focusing on "inter-domain" routing and leveraging the concept of BGP which are beyond the scope of this paper.

VI. Conclusion

Explicit support of in-network caching and multipath forwarding from routing protocol is key to realize NDN. This paper proposed MUCA as a stand-alone intra-domain routing protocol for NDN and highlighted its important features. It makes three main contributions: (1) a combination of link-state and distance-vector routing protocol classes to efficiently support multipath routing, (2) a new path, different from the regular forwarding paths, to effectively exploit built-in caching opportunity in NDN, and (3) a new mechanism for LSDB synchronization, where the incremental routing updates are simply notified to the neighbor routers. Finally, MUCA equips the forwarding plane with a new ranked list of forwarding paths. Our in-depth evaluation demonstrates the benefits of MUCA and its superiority over NLSR, the current de facto routing protocol of NDN testbed.

We expect that MUCA will play a key role in NDN, as what OSPF has done in IP-based networks. MUCA can easily accommodate new ideas/solutions thanks to its flexible design. MUCA is, therefore, a good starting point towards a comprehensive routing solution for NDN.

Acknowledgments

This work was supported in part by NSF under Grants CNS-1345142 and CNS-1629009.

NDN can work in two name-announcement modes (i) active mode (where the prefixes are announced by producers), and (ii) passive mode (where the prefixes are solicited by consumers).

References

[1] "OSPF Version 2," https://www.ietf.org/rfc/rfc2328.txt, [Online].
[2] "Supelec," http://wwwdi.supelec.fr/software-orig/ashiip/, [Online].
[3] A. Afanasyev *et al.*, "ndnSIM: NDN simulator for NS-3," Tech. Rep. NDN-0005, 2012.
[4] T. Bu and D. Towsley, "On distinguishing between internet power law topology generators," in *IEEE INFOCOM'02*, 2002, pp. 638–647.
[5] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, and G. Rossini, "INFORM: A dynamic interest forwarding mechanism for information centric networking," in *ICN'13*, 2013, pp. 9–14.
[6] H. Dai, J. Lu, Y. Wang, and B. Liu, "A two-layer intra-domain routing scheme for named data networking," in *IEEE GLOBECOM'12*, 2012, pp. 2815–2820.
[7] S. DiBenedetto, C. Papadopoulos, and D. Massey, "Routing policies in named data networking," in *ACM SIGCOMM workshop on Information-centric networking (ICN'11)*, 2011, pp. 38–43.
[8] J. Garcia-Luna-Aceves, "Name-based content routing in information centric networks using distance information," in *ACM Conference on Information-Centric Networking (ICN'14)*, 2014, pp. 7–16.
[9] E. Hemmati and J. Garcia-Luna-Aceves, "A new approach to name-based link-state routing for information-centric networks," in *ACM Conference on Information-Centric Networking (ICN'15)*, 2015, pp. 29–38.
[10] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," in *ACM SIGCOMM workshop on Information-centric networking (ICN'13)*, 2013, pp. 15–20.
[11] M. Lee, K. Cho, K. Park, T. Kwon, and Y. Choi, "SCAN: Scalable content routing for content-aware networking," in *IEEE ICC'11*, 2011, pp. 1–5.
[12] H. Liu, X. De Foy, and D. Zhang, "A multi-level DHT routing framework with aggregation," in *ACM SIGCOMM workshop on Information-centric networking (ICN'12)*, 2012, pp. 43–48.
[13] J. Rajahalme, M. Särelä, P. Nikander, and S. Tarkoma, "Incentive-compatible caching and peering in data-oriented networks," in *ACM CoNEXT'08*, 2008, pp. 1–6.
[14] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Cooperative routing protocol for content-centric networking," in *IEEE LCN'13*, 2013, pp. 699–702.
[15] J. V. Torres, L. H. G. Ferraz, and O. C. M. B. Duarte, "Controller-based routing scheme for named data network," Electrical Engineering Program, COPPE/UFRJ, Tech. Rep., 2012.
[16] M. Tortelli, L. A. Grieco, G. Boggia, and K. Pentikousis, "COBRA: Lean intra-domain routing in NDN," in *IEEE CCNC'14*, 2014.
[17] Y. Y. L. W. B. Z. V. Lehman, A. M. Hoque and L. Zhang, "A secure link state routing protocol for NDN," Tech. Rep. NDN-0037, Jan. 2016.
[18] L. Wang, A. K. M. M. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for named data networking," University of Memphis and University of Arizona, Tech. Rep., 2012.
[19] S. Wang, J. Bi, and J. Wu, "Collaborative caching based on hash-routing for information-centric networking," in *SIGCOMM'13*, 2013, pp. 535–536.
[20] Y. Wang, K. Lee, B. Venkataraman, R. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *IEEE INFOCOM'12*, 2012, pp. 286–291.
[21] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *ACM Conference on Information-Centric Networking (ICN'14)*, 2014, pp. 27–36.
[22] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.
[23] L. Zhang *et al.*, "Named Data Networking (NDN) Project," Tech. Rep. NDN-0001, 2010.
[24] X. Zhang, T. Niu, F. Lao, and Z. Guo, "Topology-aware content-centric networking," in *SIGCOMM'13*, 2013, pp. 559–560.
[25] Z. Zhu and A. Afanasyev, "Let's ChronoSync: Decentralized dataset state synchronization in named data networking," in *IEEE ICNP'13*, 2013, pp. 1–10.