On the Prefix Granularity Problem in NDN Adaptive Forwarding

Teng Liang, Junxiao Shi, Yi Wang, *Member, IEEE*, and Beichuan Zhang, *Member, IEEE*

***Abstract*— One unique architectural benefit of Named Data Networking (NDN) is adaptive forwarding, i.e., the forwarding plane is able to observe past data retrieval performance and use it to adjust forwarding decisions for future Interests. To be effective, adaptive forwarding assumes that *Interest Routing Locality* is related to Interests' common name prefix, meaning that Interests sharing the same prefix are likely to follow a similar forwarding path within a short period of time. Since Interests can have multiple common prefixes with different lengths, the real challenge is determining which prefix length should be used in adaptive forwarding to record path performance measurements - we refer to this as the *Prefix Granularity Problem*. The longer the common prefix is, the better the Interest Routing Locality, and the larger the forwarding table. Given the limited FIB size, route names are designed to be considerably shorter than Interest names. Existing adaptive forwarding designs use route names to record path performance measurements, which looses forwarding adaptability as it promises in the event of partial network failures. In this work, we propose to dynamically aggregate and de-aggregate name prefixes in the forwarding table in order to use the prefixes that are the most appropriate given current network situation. In addition, to reduce the overhead of adaptive forwarding, we propose mechanisms to minimize the use of the longest prefix matching in Data packet processing. Simulations demonstrate that the proposed techniques can result in better forwarding decisions in the event of partial network failures with significantly reduced overhead.**

***Index Terms*— Information-centric networking (ICN), named-data networking (NDN), adaptive forwarding, prefix granularity problem.**

I. Introduction

IN NAMED Data Networking (NDN), applications produce *Data* packets containing named content with names. To retrieve data, users send *Interest* packets also identified by names. Routers forward Interests based on their name, and Data packets are retrieved though the reverse path of the Interest they satisfy. This stateful Interest-Data exchange pattern enables **adaptive forwarding** in NDN [1], i.e., NDN's forwarding plane is able to observe the data retrieval performance of past Interests and use it to improve the forwarding decisions made for future Interests. For example, an NDN node can measure the round-trip time of Interest-Data exchanges between multiple next hops, and use this information to dynamically pick the best next hop to use for future Interests that share the same name prefix. This unique forwarding adaptability provides better network performance in the event of short-term churns, such as network failures and congestion.

To be effective, NDN adaptive forwarding assumes that ***Interest Routing Locality*** is related to Interests' common name prefix, meaning that Interests sharing the same prefix are likely to take a similar forwarding path within a short period of time. In addition, we assume that Interests sharing a longer name prefix have better Interest routing locality, meaning that they are more likely to take the same forwarding path. This assumption is made from our observations that applications name data based on how it is organized and accessed, because Data packets sharing a longer name prefix have closer connections. For example, `/a/b/seg=1` and `/a/b/seg=2` refer to two segments of the same file, while `/a/b/seg=1` and `/a/c/seg=1` refer to segments in two different files.

Given that Interests can have multiple common prefixes with different lengths, the real challenge is determining which prefix length should be used when performing path performance measurements in adaptive forwarding. Recording path performance measurements on a longer name prefix provides better Interest routing locality, thus helping the forwarding plane to make better forwarding decisions. However, these records will cover fewer Interests, meaning that the forwarding table needs to maintain more records of different name prefixes. We define this problem - which name prefix length is used to record path performance measurements - as the ***Prefix Granularity Problem***[1] in NDN adaptive forwarding. The challenge is to balance the trade-off between Interest routing locality and forwarding table size.

Given the limited FIB size, route names are designed to be considerably shorter than Interest names. More specifically, applications typically register a short common name prefix for their data with routers, and routers may further aggregate route names to reduce their forwarding table sizes. Short route names lead to poor Interest routing locality in the event of

Manuscript received November 17, 2020; revised March 24, 2021; accepted August 2, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Schapira. Date of publication August 27, 2021; date of current version December 17, 2021. This work was supported in part by the National Science Foundation under Grant CNS-1629009, in part by the National Key Research and Development Program of China under Grant 2019YFB1802600, in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2019B121204009, in part by the Peng Cheng Laboratory Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications under Grant LZC0019, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B1515120031. *(Corresponding authors: Teng Liang; Yi Wang.)*

Teng Liang and Yi Wang are with the Network Communication Center, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: liangt@pcl.ac.cn; wy@ieee.org).

Junxiao Shi is with the National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899 USA (e-mail: junxiao.shi@nist.gov).

Beichuan Zhang is with the Department of Computer Science, The University of Arizona, Tucson, AZ 85721 USA (e-mail: bzhang@cs.arizona.edu.com).

Digital Object Identifier 10.1109/TNET.2021.3103187

[1]This work has a previous version published in ACM ICN conference [2].

partial network failures. This problem is inherited in NDN adaptive forwarding. Existing adaptive forwarding designs use **route names** to record path performance measurements from past data retrievals, which is known to encounter difficulties in handling partial network failures. For example, if the route name is `/a`, but Interests within two sub-namespaces `/a/b` and `/a/c` have different best forwarding paths, picking either sub-namespace's best forwarding path as the best path of the route name will make suboptimal or incorrect forwarding decisions for the other sub-namespace.

This work tackles the Prefix Granularity Problem by dynamically expanding and collapsing forwarding table, i.e., disaggregating and aggregating name prefix in forwarding table on the fly. Specifically, once the forwarding plane detects that the traffic under a route name has different optimal forwarding paths, it will disaggregate the route name into several sub-namespaces, expanding forwarding table. Path performance measurements will be recorded on these longer names accordingly, and the traffic under each sub-namespace will be forwarded to the optimal path of that sub-namespace. We propose three different expanding algorithms to determine both when and how to disaggregate a namespace. Because packets may arrive in an unpredictable order, the proposed expanding algorithm may create unnecessary sub-namespaces. To address this issue, we also propose forwarding table collapsing techniques for table optimization.

Another problem with existing adaptive forwarding designs is that they require longest-prefix-match lookups during Data packet processing in order to record performance measurements in the *Forwarding Information Base* (FIB). This results in significantly higher overhead than non-adaptive forwarding, which does not need longest-prefix-match lookups.

To tackle the problem, we start from the observation that if an Interest was only forwarded to the optimal path, the performance measurement on that single path would not help adaptive forwarding to rank multiple next hops, so that it is unnecessary to record this path performance measurement. When an Interest is forwarded to multiple next hops because of either probing or retransmission, adaptive forwarding will collect performance measurements for multiple next hops. However, FIB updates are needed only if the collected route ranking differs from the current route ranking. Therefore, we add these two filters to limit FIB updates, which significantly reduces the overhead of FIB updates during Data processing, with the assumption that a majority of the traffic will be forwarded on the optimal path within a short period of time. Specifically, we store a copy of performance measurements within the Pending Interest Table (PIT) entry, which is already accessed during Data processing. The forwarding plane locates and updates the FIB entry only if the next-hop ranking has changed, after which it triggers FIB expanding algorithms as necessary.

To summarize, the two major contributions of this work are:

- We tackle the Prefix Granularity Problem with dynamic FIB expanding and collapsing (FIB name disaggregation and aggregation) algorithms to improve forwarding decisions with moderate overhead. More specifically, we design, implement and, evaluate FIB expanding techniques.
- We optimize the current data retrieval performance measurement process in adaptive forwarding, which improves Data processing performance by eliminating a majority of the longest-prefix-match name lookups in the Data forwarding pipeline.

Simulations demonstrate that the proposed techniques can make better forwarding decisions during partial network failures with significantly reduced overhead. The rest of paper is organized as follows. Section II introduces related work and elaborates the Prefix Granularity Problem by examining a concrete example, and introduces related work. Then, our design rationale is explained in Section III, which is followed by design details (Section IV). We evaluated our proposed solution in Section VI. Finally, Section VII concludes the paper.

II. Problem Statement

A. NDN Adaptive Forwarding

NDN uses a pull communication model, and it has two major types of network packets, *Interest* and *Data*; both packet types carry a variable-length hierarchical name. The data receiver, called a *consumer*, transmits an Interest packet with the desired Data packet name to the network. Upon receiving an Interest, the NDN router first queries the *Content Store* (CS) for a locally cached Data packet. If no matched Data exists, it then queries the *Pending Interest Table* (PIT) to identify whether the Interest is new, looped, or retransmitted. Finally, the router forwards the Interest according to the *Forwarding Information Base* (FIB), which contains a ranked list of next hops from which forwarding strategies can make forwarding decisions. Forwarded Interests are buffered in the PIT, allowing Data packets to be returned to consumers via the reverse path. Data packets are cached in the CS to satisfy future Interests requesting the same data.

This stateful Interest-Data exchange pattern enables adaptive forwarding in NDN [1]. Specifically, with PIT states, NDN forwarding plane is able to **explore** and **measure** multiple paths without worrying about loops. To utilize multiple paths, the adaptive forwarding plane performs *Interest Retransmission Processing* by forwarding retransmitted Interests to alternative paths that have not been tried, as well as *Interest Probing* by occasionally forwarding an Interest to multiple next hops. Given that each Interest-Data exchange takes path performance measurements (e.g., round-trip time and throughput) on one next hop, these two mechanisms enable the forwarding plane to measure path performance on multiple next hops, and to make a route ranking. With a route ranking, NDN's forwarding plane can make better forwarding decisions, thus improving data retrieval performance on the fly.

One benefit of NDN adaptive forwarding is to better handle link failures. Regarding link failures, if link layers can quickly detect them and inform the network layer, routing protocols can propagate new routing announcements, which will remove failed links from FIB. In other cases where lower-level detection is unavailable, network layers can rely on routing

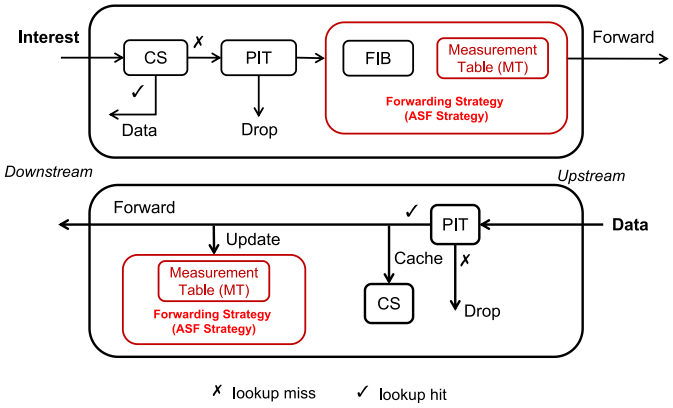


Fig. 1. The adaptive forwarding (i.e., the ASF forwarding strategy) packet processing workflow.

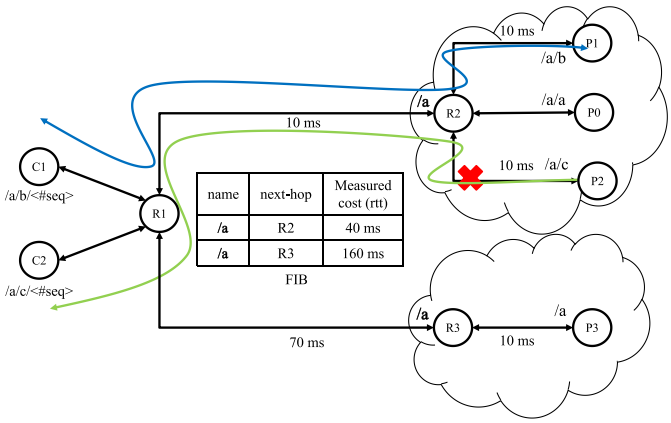


Fig. 2. A simple scenario to demonstrate the FIB prefix granularity problem.

protocols' periodic keep-alive messages to detect failures, which usually takes seconds or even tens of seconds. In NDN, adaptive forwarding can quickly react to link failures without relying on link layer failure detection mechanisms or routing protocols. Because the failed link will not bring back data, it will be marked as the lowest ranked next-hop. In addition, if consumers retransmit Interests that have not retrieved Data, e.g., after a round-trip timeout, the retransmitted Interests will be sent to an alternative path. Therefore, the damage of link failure is minimized with NDN adaptive forwarding.

In the current adaptive forwarding design [1], [3] and implementation, such as Adaptive Smoothed RTT-based Forwarding (ASF) strategy [4], performance measurements are recorded in the *Measurement Table* (MT) (Figure 1). More specifically, during Interest processing, MT lookup occurs after FIB lookup to find the ranking of next hops based on the latest measurements. During Data processing, the MT is updated on each Data packet reception. The overhead of managing MT is analyzed in Section II-C.

B. The Prefix Granularity Problem

In this section, we introduce the Prefix Granularity Problem in NDN adaptive forwarding. First, a simple scenario is used to demonstrate this problem, and the limits of the current design. Then, the root cause of this problem - route name prefix granularity is discussed. Next, we argue that simply using a routing protocol has limits in solving this problem. In addition, a comparison to IP anycast is made for a better understanding of the uniqueness of this problem in NDN. Last, an alternative solution without adaptive forwarding is discussed.

1) A Simple Scenario: The Prefix Granularity Problem is elaborated using an example shown in Figure 2, in which two consumers $C1$ and $C2$ are connected to three producers (i.e., applications that serve data) $P0$, $P1$ and $P2$, and a data repository $P3$, through routers $R1$, $R2$ and $R3$.

In this example, $P0$, $P1$ and $P2$ are producers that serve data under name prefixes `/a/a`, `/a/b` and `/a/c` respectively, and they are connected to the site router $R2$. $R2$ announces the common name prefix `/a` to its neighbors. $P3$ is a data repository that serves data under name prefix `/a`, and it is connected to the site router $R3$. Through a routing protocol, router $R1$ learns a FIB entry `/a` with the initial next-hop ranking such that $R2$ is preferred to $R3$, because of a shorter delay.

Consumers $C1$ and $C2$ start sending Interests to retrieve data under name prefixes `/a/b/<#seq>` and `/a/c/<#seq>` respectively. Initially, routers forward Interests from $C1$ to $P1$ (the blue flow), and Interests from $C2$ to $P2$ (the green flow), because $R1$ picks $R2$ as the best next hop. Then, the network link between $R2$ and $P2$ fails.

Because of NDN adaptive forwarding, when the network link fails, ideally $R1$ should start forwarding Interests from $C2$ to $P3$, given that the timed-out Interest resulted from the link failure would move $R2$ to a lower ranking in adaptive forwarding. However, the current design records next-hop performance measurements at FIB entry level, i.e. `/a`, which associates the `/a` prefix with either the existing next-hop ranking ($R2$, $R3$) or the new ranking ($R3$, $R2$). In the former case, $C2$ is unable to retrieve data, even if there exists a working path to $P3$. In the latter case, both consumers can retrieve data, but $C1$ is retrieving data from $R3$, which has a higher round-trip time than retrieving data from $R2$.

This scenario is simulated with ASF strategy in Section VI-B. The simulation results match the first aforementioned case, that $R1$ continues to forward Interests from $C2$ to $R2$ after the link failure. This is because the Interest-Data exchange between $C1$ and $P1$ (the blue flow) is still working after the link failure, and each Data reception at $R1$ triggers adaptive forwarding to update round trip time at FIB name `/a`. Although the Interest-Data exchange between $C2$ and $P2$ (the green flow) breaks after the link failure, and the sent Interests of the green flow after the link failure are eventually timed out, which mark $R2$ with a lower ranking at $R1$. However, because the updates from the blue flow are more frequent than the updates from the green flow, hence $R1$ still considers $R2$ to be a better choice than $R3$ for the name prefix `/a`.

2) Route Name Prefix Granularity: In an NDN network, routes not only can be location names, but also can be data name prefixes, referring to paths toward specific collections of data. Given that each segment of a file has its own unique name, and the number of file segments in a network can be gigantic, it is unscalable to use file segment name as route

name in an NDN network. To reduce the size of FIB table, a producer normally registers a short name prefix to a router, and routers may further aggregate route names to a shorter common name prefix. Therefore, there exists a difference between route name granularity and data name granularity, and this name difference leads to insufficient Interest routing locality in the situation of partial routing failure (e.g., a file is deleted on one site).

This problem is inherited in NDN adaptive forwarding, but is not addressed by existing designs, which use static route name to record path performance measurement. More specifically, path performance measurement collected from one Interest-Data exchange only indicates the network situation for a specific data name, not necessarily reflecting the network situation for the whole route name, hence recording performance measurement at route name may make suboptimal or wrong forwarding decisions (e.g., Fig 2). As a result, adaptive forwarding becomes ineffective in this scenario. This work intends to address this issue by dynamically expanding route names to record path performance measurement.

3) Why Does a Routing Protocol Not Help: One common argument is that the problem demonstrated in the simple scenario (Fig 2) can be solved with the help of a routing protocol, e.g., if $R2$ announces $/a/b$ through a routing protocol after the link fails, then $R1$ will forward traffic under $/a/c$ to the only available next-hop $R3$.

There are three issues in this solution. First, as analyzed in the previous section, to reduce FIB size, route names are designed to be short, and $R2$ may be configured to announce route name only at level-one granularity, i.e., $/a$. Second, even if $R2$ is able to announce a longer route name prefix, it may not be able to detect the unreachability of $/a/c$, which can be one of many file names in the producer. Last, relying on a routing protocol, $R2$ not only needs to announce $/a/b$, but also to announce $/a/a$; there can be more name prefixes to be announced at level-two, and they should be announced as well. As a result, the size of routing announcement, RIB, and FIB may be significantly increased at each router in the network.

Instead of relying on a routing protocol to propagate finer route name prefixes, this work intends to address this problem in adaptive forwarding, which relies on local path performance measurement and make local forwarding decisions accordingly.

4) Comparing to IP Anycast: The given scenario can be implemented in an IP network via IP anycast [5], which allows multiple hosts to have and announce the same unicast IP addresses to the network. IP anycast is widely used for DNS and CDN. In practice, IP anycast can be announced as a subnet of the fixed size $/24$ to BGP, then an IP router will forward packets to the closest matched IP anycast net. Given that IP forwarding has no adaptability, if the destination machine or the link to a subnet fails, an IP router is unable to detect the failure or forward traffic to an alternate IP anycast location. In NDN, although the fixed-length address becomes a variable-length data name, the prefix granularity problem of "anycast" still exists. NDN adaptive forwarding treats one route name prefix as a unit for path performance measurements and path selection, which is similar to the fixed size $/24$ address in IP anycast. However, the network may have a failure that affects a finer name prefix granularity, but the current adaptive forwarding is unable to detect or handle it. This work is intended to tackle this problem.

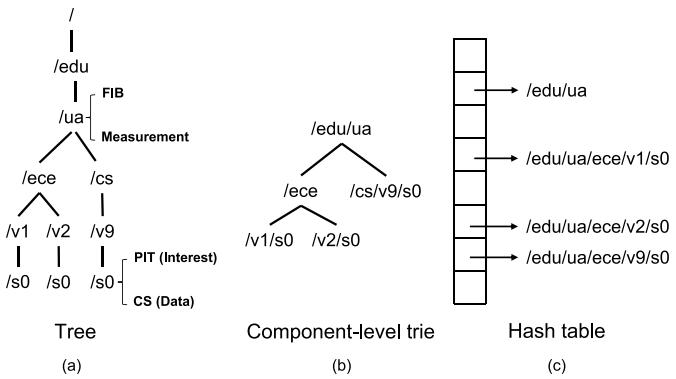


Fig. 3. Three data structures to store name-based entries.

5) An Alternative Solution Without Adaptive Forwarding: Last, we discuss an alternative solution to this problem. Instead of using data name prefixes as routes, the alternative solution uses location names as routes. For example in Fig. 2, $R1$ has two routes with names $/R2$ and $/R3$, pointing to sites $R2$ and $R3$ respectively. Within each site, $R2$ and $R3$ have their data-name-based routes. In addition, consumers require a global DNS-like lookup system to find the locations for a given data name, such as NDNS [6].

In the alternative solution, $C2$ first queries NDNS, and finds that both $R2$ and $R3$ sites have data start with name $/a$; then $C2$ sends an Interest with name $/a/c$ with a forwarding hint $/R2$; the Interest containing the forwarding hint is sent to router $R1$, and is forwarded based on the forwarding hint, thus it will be forwarded to $R2$ to retrieve data. Once there is a link failure and data cannot be retrieved from site $R2$, $C1$ will try to retrieve data from site $R3$ by choosing its site name as the forwarding hint.

The alternative solution requires consumers to support the forwarding adaptability logics, inter-sites routers to support forwarding-hint based forwarding mechanism, and a global DNS-like lookup system. In contrast, this works intends to solve the problem by only improving NDN adaptive forwarding.

C. Table Lookup in Adaptive Forwarding

The second problem of the current adaptive forwarding design is its performance overhead introduced to the Data processing pipelines. To understand the problem, this section analyzes how NDN adaptive forwarding stores various states, their relationships, and mechanisms to improve their processing performance.

NDN adaptive forwarding stores various states in CS, PIT, FIB and MT. All these table are indexed by names. One example of their relationships is in Figure 3(a). Specifically, Data packets are identified by their complete names (e.g. */edu/ua/cs/v9/s0*). Most Interests have the same name as Data, but NDN also allows an Interest name to have a shorter prefix of the Data name, which enables in-network name

discovery [7]. FIB entry names are prefixes of Interest names, and are considerably shorter than typical Interest names, because an application usually registers a smaller number of name prefixes to cover all the data it serves, and routers can further aggregate prefix registrations to fewer routing announcements. Finally, existing adaptive forwarding designs record path performance measurements along with FIB name.

After briefly summarizing the relationships among table and packet names, we list detailed packet processing pipelines for each table, to better understand the overhead of each table lookup and the bottleneck.

In Interest processing:

- *CS Lookup:* conducts *Exact Match* against Interest name. NDN allows Interest to retrieve Data with a longer name. Supporting such name lookup in CS provides one use case, i.e., CS can provide a matched Data with any version to an Interest without version number. However, this use case is needed only in limited scenarios. Therefore, CS lookup can get rid of such support to reduce processing overhead, since exact match is considerably more efficient than finding Data with a longer name. In addition, this allows CS and PIT to be stored on the same data structures, and merge the two table lookups into one name lookup [8].
- *PIT Lookup:* conducts *Exact Match* against Interest name.
- *FIB Lookup:* conducts the *Longest Prefix Match (LPM)* against Interest name, because LPM can provide the most accurate routes. LPM is considered as the bottleneck of name lookup performance, and various data structures have been proposed to make it more efficient.
- *MT Lookup:* conducts LPM against Interest name, because existing designs record path performance measurements with FIB name. Therefore, FIB and MT lookups can be merged into one LPM name lookup.

In Data processing:

- *PIT Lookup:* expects a Data packet to satisfy any Interest whose name is a prefix of Data name, e.g., Data with name */a/1* should satisfy Interests with either name */a* or */a/1*. For major cases, Interest name and Data name are the same. For minor cases, Interest name is a prefix of Data name. These two types of cases require processing with significantly different overhead, the former one only needs exact match, while the latter one requires to look up all prefixes of data name in PIT. To distinguish these two types, *CanBePrefix* flag is tagged on Interest if its name can be a prefix of data name [9]. To further improve this lookup performance, PIT token [10] is introduced as an index carried by both Interest and Data. This work does not rely on the usage of PIT token.
- *CS Lookup:* conducts *Exact Match* against Data name.
- *MT Lookup:* conducts LPM against Data name to find a FIB name, because path performance measurements are bound with FIB name in existing designs. This lookup becomes the bottleneck in Data processing.

To summarize, without sacrificing the major forwarding semantics, table lookups use simpler matching rules to reduce memory access. More specifically, PIT and CS use exact match (PIT token is used in Data processing), while FIB and MT use LPM which is the bottleneck of packet processing. In addition, the same matching rules allow them to share the same data structure and merge table lookups into one name lookup. This work proposes a different technique to manage MT, in order to reduce LPM in Data processing (Section III-C).

Another major research topic is to design data structures to improve name lookup performance, such as to use trie [11], hash table [8] and Bloom filter [12]. Figure 3 gives an example of three different data structures. Regarding a tree data structure, LPM requires tree traversal from root to leaf. To reduce the traversal distance, trie-based data structures are proposed to merge nodes for parents that have one child. However, the worse case still incurs massive memory access to visit a leaf node in the tree, proportional to the height of a name trie. Another idea is to use hash table to store names, which changes leaf lookup in tree-like structures to constant time. The challenge is that LPM can be inefficient as the name prefix of descending length has to be checked. *2-stage LPM algorithm* [8] is proposed to reduce the number of prefix to be checked. This work is not arguing which data structure is better. Although a tree structure is used to explain the proposed algorithms, the algorithms are independent of any data structure.

III. Design Rationale

In this section, we consider the design rationale of the proposed FIB expanding and collapsing techniques.

A. Dynamic FIB Expanding

As elaborated in Section II-C, existing adaptive forwarding designs use static FIB name to address the Prefix Granularity Problem, which has been proved to be ineffective in handling partial network failures. This problem can be solved by disaggregating FIB name into longer names to record the observed path performance measurements. In other words, the measurements can be recorded at a finer grained-prefix that better matches network conditions. Figure 4 demonstrates how disaggregating FIB names solves the Prefix Granularity Problem in the given scenario; after the link failure, $R1$ records the measured path performance on a longer name prefix `/a/c`, so that the green flow for `/a/c` will be forwarded to an alternative working path, while the blue flow for `/a/b` sticks to the best path.

We refer to this solution as *Dynamic FIB Expanding*. The challenge is how to quickly disaggregate FIB name to the accurate name prefix granularity, so that it matches the current network situation, while the introduced overhead is minimized. Figure 5 gives three examples of FIB update after observing the past path performance measurements. Example (1) updates the next-hop ranking on FIB name directly, which is how existing adaptive forwarding designs work. Examples (2) and (3) update the next-hop ranking on disaggregated names that are finer grained than the original FIB name.

One question is how long should the FIB name be disaggregated to. Example (2) expands the original FIB name with one more name component, while example (3) expands it with

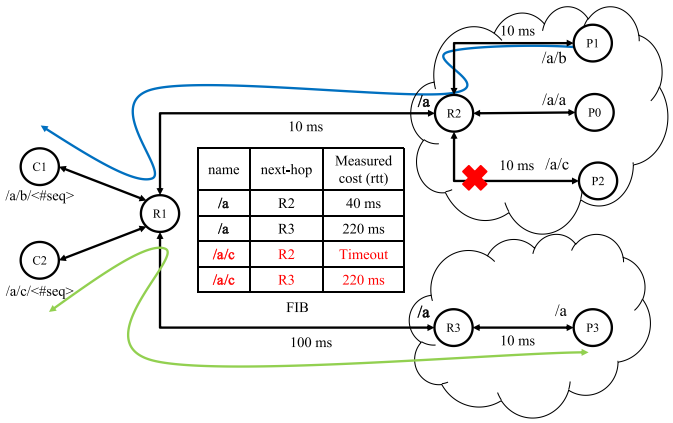


Fig. 4. Use FIB expanding to solve the prefix granularity problem in the simple scenario.

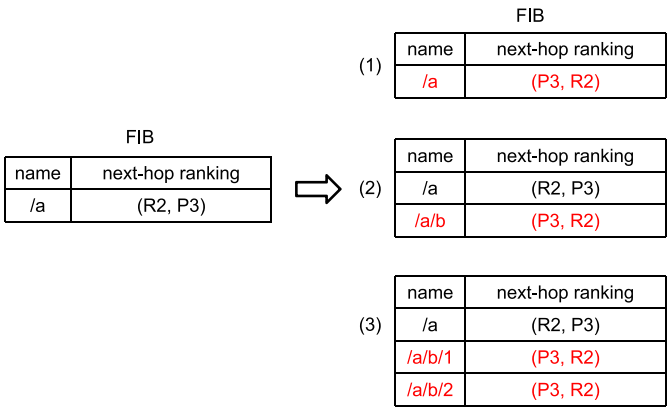


Fig. 5. Three examples to update FIB after measuring the next-hop ranking of multiple paths: (1) no FIB expanding, (2) FIB expanding with the accurate name prefix, and (3) FIB expanding with the name prefix longer than the accurate one.

two more name components. The goal is to expand the FIB name long enough so that all traffic under the expanded name share the same best path, but not too long to unnecessarily increase FIB size. We define the shortest expanded name that reflects the new network situation as the "accurate" name prefix. Therefore, a good FIB expanding algorithm should be able to find the accurate name prefix.

Another metric to evaluate a FIB expanding algorithm is the number of FIB updates required to find the accurate name prefix. Given that the accurate name prefix is not known in advance, and there are different ways to expand a name, e.g., adding one or two more name components, FIB expanding algorithms may need several rounds of measurements to find the accurate name prefix. Before approaching the accurate name prefix, each FIB expanding implies that a portion of traffic has not been forwarded via the optimal path. A good FIB expanding algorithm should be able to find the accurate name prefix with a smaller number of FIB updates. In this work, three different expanding algorithms are proposed (Section IV) and evaluated (Section VI) by the aforementioned metrics.

B. FIB Expanding Triggering

FIB expanding is triggered when the network situation has changed (e.g. a link failure in Figure 2), and the current next-hop ranking on a FIB name is not accurate anymore, meaning that a sub-namespace of the FIB name uses a different next-hop ranking that represents the current network situation the best. Therefore, FIB expanding is triggered only when a different next-hop ranking on a FIB name is observed.

We first make an assumption that a majority portion of traffic follows the first ranked path to retrieve data. The performance measurement on the single path is unable to generate a next-hop ranking. Only when traffic explores multiple paths, the measurements can generate a ranking of next hops, which then triggers FIB expanding if it is different from the current one. This analysis motivates us to rethink the usage of measurement, which is to generate the ranking of next hops. Therefore, if an Interest-Data exchange only measures a single path, the information is of little value. Based on this idea, we propose measurement management optimization in next section.

C. Measurement Management Optimization

The current adaptive forwarding design records path performance measurements on FIB entries, thus each Data reception will trigger a FIB lookup, which requires the longest prefix match, introducing significant performance overhead to the Data processing pipelines (Section II-C). Given that a majority of Interest-Data exchanges only use the first ranked path, the path performance measurement of the single path has little value because it cannot change the ranking among multiple next hops. Therefore, we propose to remove the binding between path measurement and FIB names.

Instead, the adaptive forwarding can record path measurement along with the PIT entry. If only one path has been used, the measurement is discarded along with the PIT entry; if multiple paths have been explored, and their measurement information generates a new next-hop ranking, FIB entry update and FIB expanding will be triggered.

The optimized packet processing workflow is depicted in Figure 6. With the optimization, path measurements are recorded with PIT entries, so that Data processing only requires one exact match lookup instead of separate lookups in PIT and FIB. Since only a small portion of traffic triggers FIB expanding, the majority of longest-prefix-match name lookups are avoided. More detailed analysis is in Section VI-D.

D. Dynamic FIB Collapsing

The FIB expanding algorithm adds more FIB entries when rankings are changed. The next two questions are when to remove the expanded FIB entries and how to manage the number of expanded FIB entries.

One possible mechanism is to add timers on newly expanded FIB entries, and these entries will be removed once the timers are expired. The timers may be set to match the routing announcement interval, because routing announcements can supply the latest next hop ranking according to routing protocol. However, periodical routing announcement update may not exist in environments such as a local network using self-learning [13], [14]. Moreover, the FIB expanding algorithm could generate more FIB entries than necessary, such as the example shown in Figure 7.

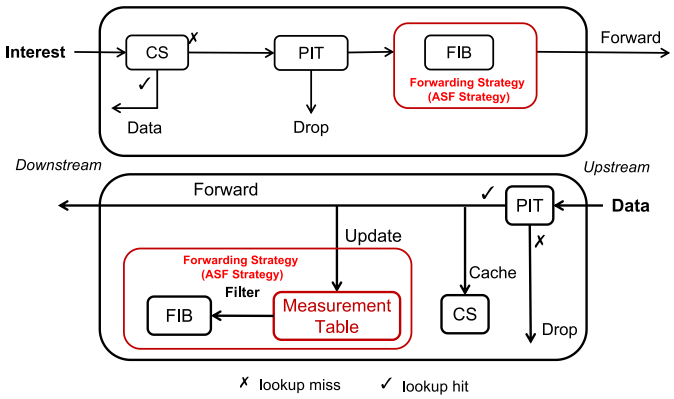


Fig. 6. Optimizing measurement management in the packet processing pipelines.

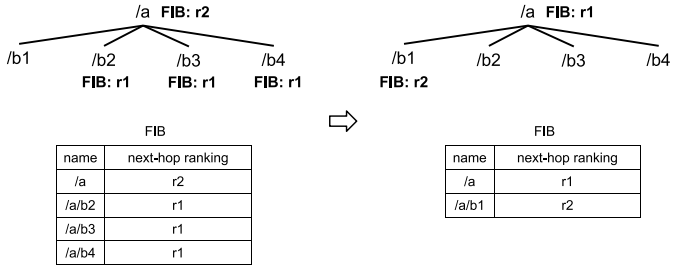


Fig. 7. An example of a FIB collapsing algorithm.

We propose a FIB collapsing algorithm to optimize the number of FIB entries created by FIB expanding algorithms. This algorithm not only removes invalid expanded FIB entries, but also optimizes FIB to reduce the number of FIB entries while keeping the same forwarding effects. For example, Figure 7 demonstrates that the FIB collapsing algorithm can consolidate four FIB names into two without changing forwarding behaviors for the specific name tree. Moreover, this algorithm does not rely on timers, but instead checks the ranking of next hops on its parent node and children nodes. More details are specified in Section IV.

E. Security Considerations

The original routes learned from either routing protocols or self-learning mechanisms are verified and trusted. The expanded FIB entries are created based on the measurements of real-time network conditions within the original route namespace. Therefore, FIB expanding mechanisms would not admit malicious routes creation.

If the Prefix Granularity Problem is not handled correctly, one potential vulnerability is "traffic hijacking". This was observed from a real scenario, that an NDN video streaming service was deployed on NDN Testbed [15]; multiple sites deployed NDN video servers serving the same name prefix, and one of them was serving an exclusive subnamespace for status collection, and this server is "hijacking" video requests that are supposed to be forwarded to other sites. One argument is that the status collection name prefix should be announced separately to the routing system, but this subnamespace can be any video file. The proposed FIB expanding techniques

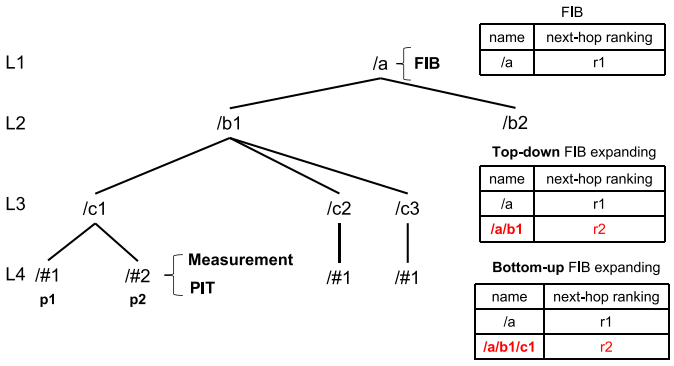


Fig. 8. An example to explain FIB expanding algorithms.

can solve this problem, and the real scenario was simulated in Section VI-C.

After introducing FIB expanding techniques, one concern is that attackers may design subtle Interest timeouts to trigger FIB expanding actions, which consumes both computation and storage resources on routers. However, this requires attackers controlling traffic. Moreover, this can be countered with better naming designs and smarter FIB expanding triggering. More study is needed for this topic.

IV. Design Details

A. FIB Expanding Detection

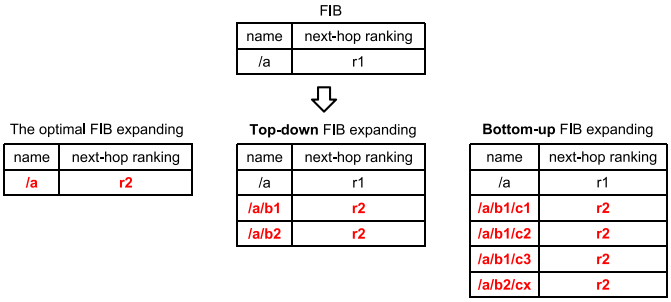
As analyzed in Section III-B, upon receiving a Data packet, the adaptive forwarding collects path performance measurement regarding the incoming face on the specific Interest name, and bounds it with the PIT entry. Based on our assumption that the majority of Interests are forwarded to the best path only, most observed path performance measurements are collected on a single path, which is unable to generate a route ranking, thus it will not trigger FIB expanding.

As introduced in Section II-A, Interest Retransmission Processing and Interest Probing are two mechanisms that can measure multiple paths. Interest Retransmission Processing forwards retransmitted Interests to different next hops in round-robin manner; Interest Probing occasionally forwards copies of Interests to alternative next hops. Both mechanisms allow the adaptive forwarding plane to measure the performance of multiple next hops, and generate a route ranking list. If the new ranking differs from the existing one in the FIB entry, the FIB expanding algorithm will be triggered.

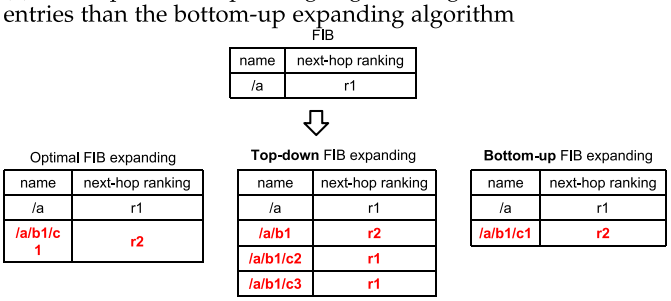
B. FIB Expanding Algorithms

This section defines three potential FIB expanding algorithms. Each algorithm decides how to create new FIB entries with longer names to record the path performance measurement. Figure 8 shows a name hierarchy, which has an existing FIB entry `/a` with the initial ranking $r1$. Suppose the Interest `/a/b1/c1/#1` was forwarded to multiple next hops and generated a different ranking $r2$ on the PIT entry $p1$, this new ranking would trigger the FIB expanding algorithm.

1) Top-Down Expanding Algorithm: This algorithm creates a new FIB entry at FIB + 1 level and records the new ranking. In Figure 8, the top-down expanding algorithm would create a FIB entry at `/a/b1`, one level down ($L2$) from the existing



(a) the top-down expanding algorithm generates fewer FIB entries than the bottom-up expanding algorithm



(b) the bottom-up expanding algorithm generates fewer FIB entries than the top-down expanding algorithm

Fig. 9. Expanded FIB results generated by the top-down expanding algorithm and the bottom-up expanding algorithm.

FIB entry ($L1$), and record the ranking $r2$ on this FIB entry. The name "top-down" reflects the behavior that it starts from the "top", which is the existing FIB entry.

2) Bottom-Up Expanding Algorithm: This algorithm creates a new FIB entry at PIT-1 level and records the new ranking. In Figure 8, the bottom-up expanding algorithm would create a FIB entry at `/a/b1/c1`, one level up ($L3$) from the PIT entry ($L4$), and record the ranking $r2$ on this FIB entry.

Both algorithms are simple to implement, but take the opposite strategies. The top-down expanding algorithm is more aggressive while the bottom-up expanding algorithm is more conservative when inserting new FIB entries. However, either algorithm may generate more FIB entries than the optimal number. Figure 9 demonstrates two examples of FIB table expanded by these two algorithms. In Figure 9a, the next-hop ranking should have been changed at `/a`, but both the top-down expanding and the bottom-up expanding algorithms generate more FIB entries. Specifically, the top-down expanding algorithm creates FIB entries with the new ranking at all $L2$ names, while the bottom-up expanding algorithm creates FIB entries with the new ranking at all $L3$ names. In Figure 9b, the optimal FIB expanding should have added a FIB entry at */a/b1/c1*, but the top-down expanding algorithm will be triggered twice to reach the accurate name prefix, while the bottom-up expanding algorithm can get the optimal results right away. These examples demonstrate that the effectiveness of top-down and bottom-up expanding algorithms depend on the distance between the accurate name level, the existing FIB name level, and the PIT name level. If the accurate name level is closer to the FIB name level than its distance to the PIT name level, the top-down expanding algorithm works better than the bottom-up expanding algorithm.

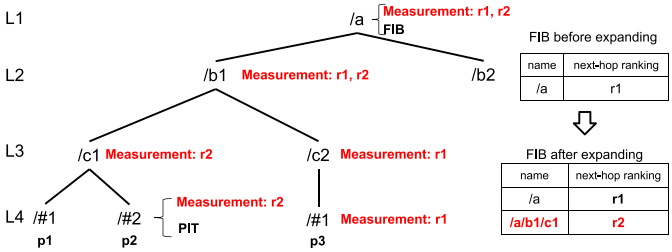


Fig. 10. An example of FIB expanding using the SS expanding algorithm.

Because the accurate name prefix depends on the network situation and is unknown a priori, neither algorithm can provide a guaranteed performance. Therefore, we intend to design an algorithm that can better locate the accurate name prefix.

3) Shortest Name Prefix With the Solo Route Ranking (SS) Expanding Algorithm: The accurate FIB name to be expanded has two characteristics. First, all Interests under this FIB name share the same route ranking. If a FIB name has different route rankings observed, the FIB name needs to be expanded. Second, the accurate FIB name has to be the shortest name prefix, so it is able to cover the whole name subtree that share the same route ranking; otherwise, more FIB names need to be added to cover the name subtree. Based on the analysis, the goal is to find the shortest name prefix that has the solo route ranking, and we name the algorithm as the SS expanding algorithm.

To achieve the goal, our idea is to add the measured route ranking on the path from the FIB name to the PIT name. By collecting more path measurement on the name tree, the SS expanding algorithm is able to find the shortest name prefix that has the solo route ranking, which is the accurate FIB name to be expanded.

In Figure 10, suppose the PIT node $p2$ with name `/a/b1/c1/2` has a route ranking $r2$ that differs from the ranking $r1$ at the FIB name */a*. When the PIT node $p3$ with the name `/a/b1/c2/1` records the route ranking $r1$ first, $r1$ is recorded on the $p3$'s name path (i.e., `/a/b1/c2`, `/a/b1`, and `/a`) in the measurement table. Then, when the PIT node $p2$ measures a new route ranking $r2$, $r2$ will be recorded on $p2$'s name path (i.e., `/a/b1/c1`, `/a/b1`, and `/a`). After this update, the SS expanding algorithm finds that the FIB name `/a/b1` is the longest name that has two different route rankings, therefore its child node `/a/b1/c1` will be created with the ranking $r2$ in FIB, as this child node is the shortest name prefix with a different solo route ranking. Eventually, the SS expanding algorithm results in FIB entries `/a` $= r1$ and `/a/b1/c1` $= r2$.

The SS expanding algorithm may have different outcomes, depends on the order of the route ranking observations. In Figure 11, if the PIT node $p2$ with the ranking $r2$ arrives first, the route ranking $r2$ at $p2$ will be updated on its name path, (i.e., `/a/b1/c1`, `/a/b1`, and `/a`). After this update, the route ranking at the FIB node `/a` will be changed to $r2$, because `/a` is the shortest name prefix that has observed only one route ranking. Next, the PIT node $p3$ observes the route ranking $r1$, which will be recorded on $p3's$ name path

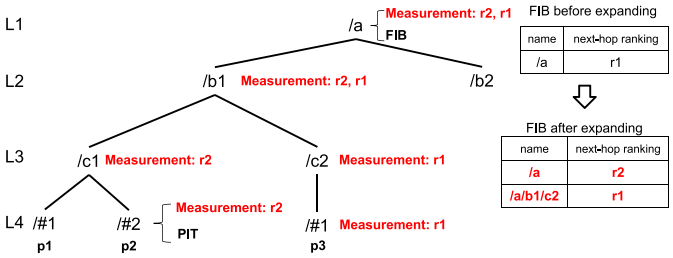


Fig. 11. An example of FIB expanding using the SS expanding algorithm with a different measurement order.

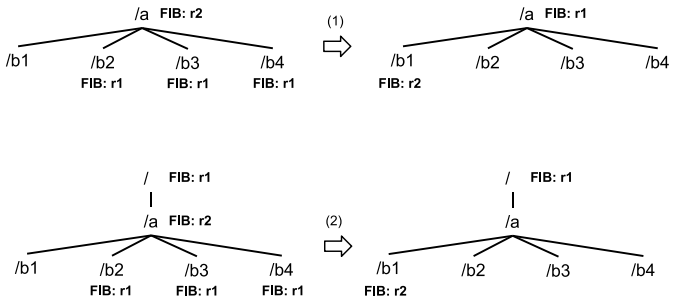


Fig. 12. Two FIB collapsing scenarios.

(i.e., `/a/b1/c2`, `/a/b1`, and `/a`), then a new node with the name `/a/b1/c2` and the ranking $r1$ will be added to the FIB. Eventually, this results in FIB entries `/a` $= r2$ and `/a/b1/c2` $= r1$.

Having different outcomes depending on route ranking observation order is an expected behavior, because FIB expanding algorithms react to the past observed route ranking. For each new observed route ranking, the FIB name tree will be expanded to match the current network situation.

C. FIB Collapsing Mechanisms

Using FIB expanding algorithms alone may cause the FIB to grow indefinitely and the router would eventually run out of memory. As explained in Section III-D, deleting dynamic FIB entries using timers may not work in all network environments. Moreover, our simulations show that FIB expanding algorithms may generate FIB trees with local optimum, such as those shown on the left side of Figure 4.3. We propose a FIB collapsing algorithm to optimize FIB tree to have a global optimum, shown on the right side of Figure 4.3, which has fewer entries but keeps the same forwarding behaviors.

The FIB collapsing algorithm requires each FIB name to remember ranking counters: how many of its children have the same route ranking. For example, on the upper left tree in Figure 12, entry `/a` should have records stating that it has 1 child with ranking $r1$ and 3 children with ranking $r2$.

The FIB collapsing algorithm works in two stages. The first stage is *collapsing triggering*. When a new FIB name with a route ranking is inserted in the FIB, the ranking counters on the parent nodes should be updated. If the difference among counters of different rankings exceeds a threshold (e.g., 2), and the most popular ranking differs from the ranking used on that node, the FIB collapsing execution is triggered.

The second stage, *collapsing execution*, has three steps:

1) Remove FIB entries from children that have the most popular ranking. In Figure 12, FIB entries `/a/b2`, `/a/b3`, and `/a/b4` are removed.
2) Add FIB entries to children that share the current ranking of the parent node. In Figure 12, a new FIB entry `/a/b1` is added with ranking $r2$.
3) Update the node's FIB entry with the most used ranking if needed. In Figure 12 (top), the FIB entry `/a` is updated with ranking $r1$. In Figure 12 (bottom), the FIB entry `/a` is removed before it has the same ranking as its parent `/`.

As these collapsing execution may change ranking counters on the parent node, the collapsing algorithm is recursively applied to the node ancestors until no collapsing is needed.

V. Implementation

We choose NFD [16] to implement the proposed techniques, because it is a widely-used open-source forwarding software running NDN protocols. More specifically, we modify the ASF adaptive forwarding strategy [4] to be able to plug FIB expanding and collapsing algorithms. Both NFD and the ASF strategy have been deployed and tested on NDN Testbed [17].

Thanks to its modular design and high-quality code, the ASF forwarding strategy is easy to extend. In addition, NFD has the core features supported. More specifically, the data structure of MT is a tree, with the lookup and insertion operations implemented in NFD; our implementation extends the operations on MT in three pipelines, via adding methods and logics of using low-level APIs. Note that the ASF forwarding strategy follows the packet processings shown in Figure 1.Evaluation details are discussed in Section VI-F. The modified modules are explained in the following.

Interest Forwarding: uses *getBestFaceForForwarding* method to find the best Face based on its measured performance information at the FIB name node. Given that FIB expanding creates new measured Face information at longer names, the measured Face information must be retrieved from the name node with the longest prefix match. More specifically, we add *getLPMFaceInfo* to replace *getFaceInfo* method to get Face information.

Interest Probing: is managed by the *NamesapceInfo* at the FIB name node. FIB expanding creates new measurement node as well as *NamespaceInfo*. Interest Probing is then managed by the *NamespaceInfo* retrieved from the name node with the longest prefix match.

Data Receiption: is the procedure to check and expand FIB when necessary. The new logics of processing Data packets are added to *beforeSatisfyInterest* method, whose pseudocode is shown in Algorithm 1. FIB expanding algorithms are triggered when the new measured route ranking is different from the current route ranking, which is implemented as *FIBExpanding* method, while FIB collapsing algorithm (implemented as *FIBCollapsing* method) is checked after each FIB expanding.

Algorithm 1 FIB Expanding Framework

```
1: procedure BEFORESATISFYINTERESTS
2:     namespaceInfo ← getNamespaceInfo()
3:     curRanking ← namespaceInfo.routeRanking
4:     newRanking ← makeRanking()
5:     if newRanking! = curRanking then
6:         newNamespaceInfo ← FIBExpanding()
7:         newNamspaceInfo.updateRouteRanking()
8:         newNamspaceInfo.updateFaceMeasurement()
9:         FIBCollapsing()
10:    else
11:        namespaceInfo.updateFaceMeasurement()
```

VI. Evaluation

A. Evaluation Setup

The evaluation is conducted in three major aspects, including network simulation, model analysis, and implement evaluation. First, we use a network simulator to evaluate the impact of FIB expanding on consumer performance in two different scenarios. The simulator is ndnSIM [18], which is built on top of NS-3 with NFD ported. Routers in both scenarios are running NFD that is configured to use ASF forwarding strategy with and without FIB expanding algorithms implemented. Then, we theoretically analyze measurement management optimization. Next, we compare and analyze the cost of the proposed FIB expanding algorithms using a name tree simulator. Last, we evaluate our implementation. All simulation results presented in this paper are confirmed after five runs with different seeds.

B. FIB Expanding in the Simple Scenario

In the first evaluation, we simulate the simple scenario shown in shown in Figure 2 in ndnSIM. The consumer applications use BIC congestion control scheme (implemented as PCON [19] consumers). The throughput bottleneck for both consumers are 8 Mbps. All applications start running at $0s$. A link failure happens between $R2$ and $P2$ at $6s$. Data retrieval rates, data retrieval delay, and Interest timeouts are measured at both consumers to demonstrate the how application performance is affected by the proposed FIB expanding algorithms.

The consumer performance of the current ASF forwarding strategy (no FIB expanding) is shown in Figure 15a. The left side shows the consumer goodput, and the right side shows data delay and Interest timeouts at consumers. The figure demonstrates that the adaptive forwarding is unable to retrieve data for the application on $C2$ after the link failure, even there exists an alternative path. This is because the Interest-Data flow between $C1$ and $P1$ keeps the adaptive forwarding plane on $R1$ believe that $R2$ is the best next hop for traffic under */a*, which makes a wrong forwarding decision for Interests from $C2$. Therefore, the consumer goodput drops to 0 after $6s$, and an Interest timeout exists at $6s$ without any delays measured afterwards.

In contrast, Figure 15b demonstrates that if the adaptive forwarding is able to add FIB entry at an accurate name prefix, $R1$ is still able to help $C2$ to retrieve data from an alternative path, without affecting traffic from $C1$. This is because the adaptive forwarding added a new FIB name `/a/c` with the next hop $P3$ in the FIB on $R1$, which forwards traffic from $C2$ to $P3$.

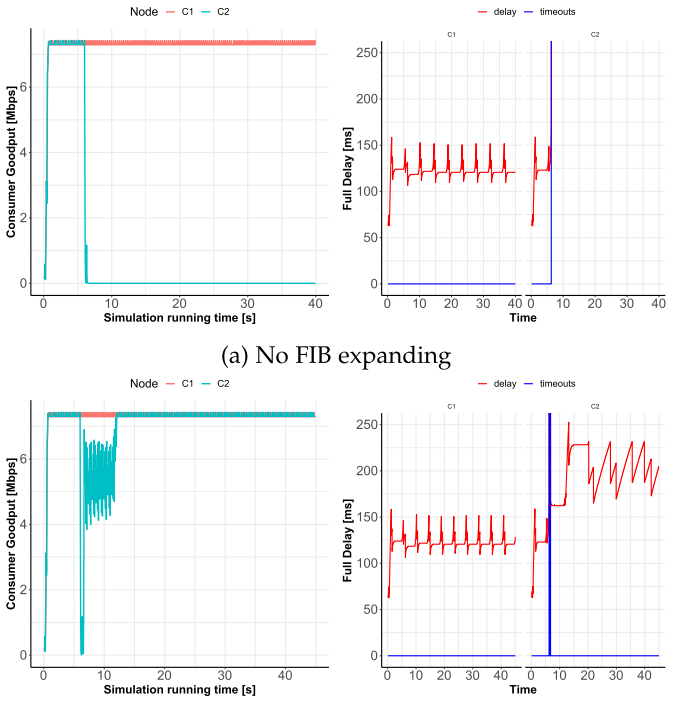


Fig. 13. Application data retrieval rates with the adaptive forwarding in the topology shown in Figure 2 under two cases: no FIB expanding and FIB expanding with the accurate name prefix hard-coded.

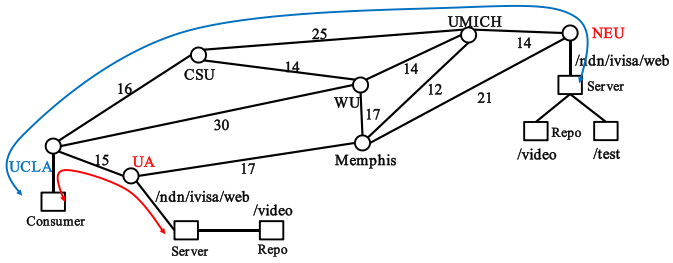
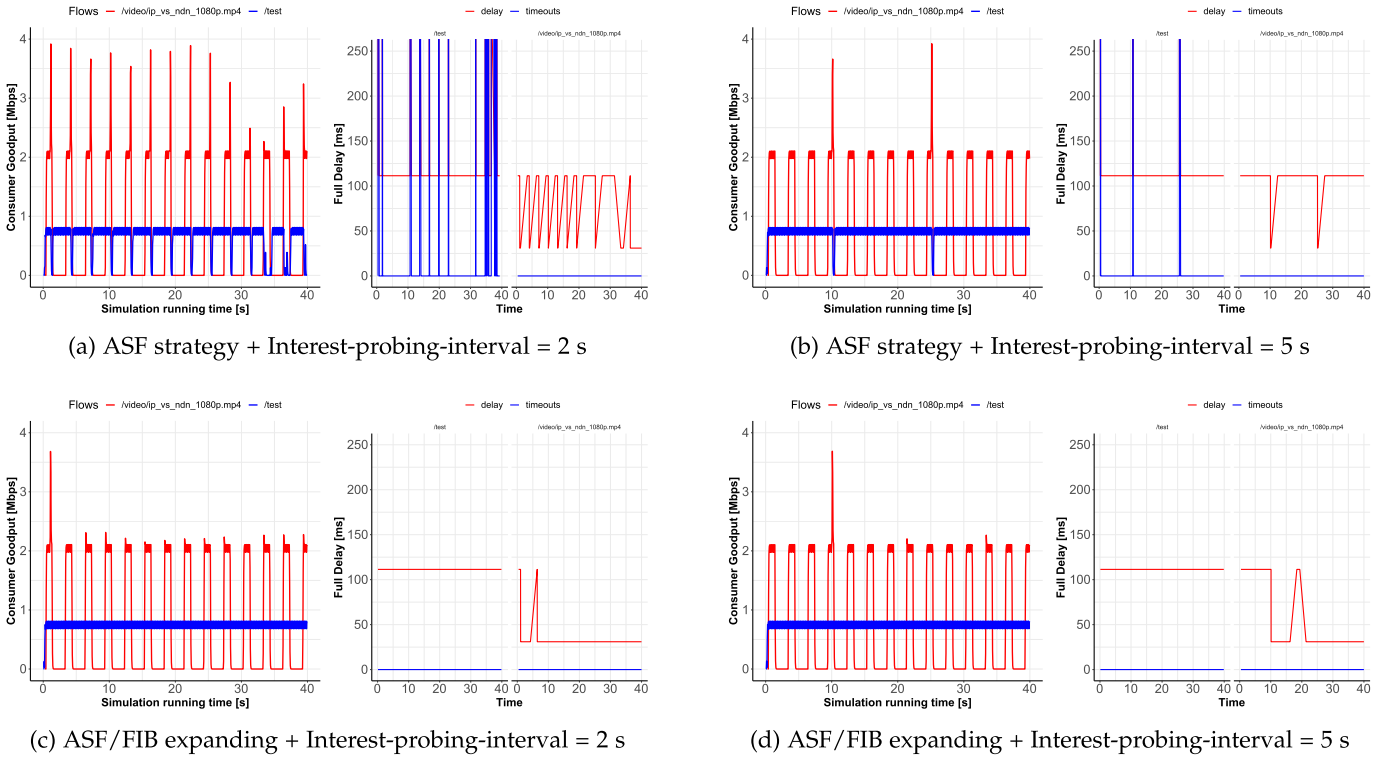


Fig. 14. The simulation topology of iVisa video streaming service on the NDN testbed.

C. A Real Scenario Observed From NDN Testbed

Next, we simulate a real scenario observed from NDN Testbed [17]. Ghasemi developed an adaptive video streaming service using NDN, and deployed it on NDN Testbed [15]. The video data repositories are put at multiple different sites, so as to satisfy nearby consumers. However, an observation is made that consumers are not always retrieving data from the nearest video repository. Based on this observation, we find and define the Prefix Granularity Problem, and propose dynamic FIB expanding techniques to solve it in this work.

To reproduce the observation, we simulate the partial network topology of NDN Testbed in Figure 14. Video servers are set up at two sites, connecting to the hub UA and NEU respectively. Both servers are serving data starts with name prefix `/ndn/ivisa/web`, except that NEU is serving an



(a) ASF strategy + Interest-probing-interval = 2 s

(b) ASF strategy + Interest-probing-interval = 5 s

(c) ASF/FIB expanding + Interest-probing-interval = 2 s

(d) ASF/FIB expanding + Interest-probing-interval = 5 s

Fig. 15. The simulation results of consumer goodput and delay for the real scenario of running iVisa video streaming service on the NDN testbed (Fig. 14).

exclusive subnamespace `/test`. In the real scenario, this unique subnamespace was used for status collecting. In theory, this subnamespace can represent any video file only serving at NEU site. A consumer connects to the hub UCLA, and it generates two Interest-Data flows, in addition to the common name prefix `/ndn/ivisa/web`, one flow is requesting data starts with name prefix `/video/ip_vs_ndn_1080p.mp4`, and the other one is requesting data starts with name prefix `/test`. For the former name prefix, the consumer is retrieving data of 2-second video frames every 2 seconds, which is to simulate the behavior of adaptive video streaming. For the latter name prefix, the consumer is sending 96 Interests per second constantly, which is to simulate status collection. The ideal behavior is that the consumer is retrieving video `/video/ip_vs_ndn_1080p.mp4` from the hub UA (using the lower path), while collecting status at the hub NEU (using the upper path). As a result, the delay of video retrieval is lower and the delay of status collection.

The simulated consumer performance results are illustrated in Figure 15. The simulation purpose is to demonstrate the impact of FIB expanding and Interest probing in handling the Prefix Granularity Problem. Figure 15a and 15b plot consumer performance results with the current adaptive forwarding design (i.e., ASF forwarding strategy) and different Interest probing intervals ($2s$ and $5s$). Given that the delay of most data retrieval is above $100ms$, both flows are retrieving data from NEU (the upper path). When the Interest probing interval is smaller, it is more likely and frequently for Interest starting with name prefix `/video` or `/test` to be forwarded to the UA. For probing Interests requesting for `video`, it retrieves data with a lower delay, which then changes the route ranking in ASF strategy, and causes a timeout for Interest starts with name prefix `test` afterwards. The timeout leads to the route ranking changing back. Therefore, Interests for `test` has more timeouts and Interest for `video` retrieves data with more lower delays, when the Interest probing interval is smaller.

To demonstrate the effectiveness of FIB expanding, we can make a comparison between Figures 15a, 15b and Figures 15c, 15d. With FIB expanding, Interests for `test` are forwarded to the upper path only, while Interests for `video` are forwarded to the lower path. This conclusion is proved by Figures 15c and 15d, in which the consumer has no timeouts for Interests starting with name prefix `test`, and has lower delay for Interests retrieving video data. The different probing intervals have an impact on how fast the video flow can be forwarded to the lower path with shorter delay. The shorter the probing interval, the sooner the shorter path will be probed which then will trigger FIB expanding.

D. Measurement Management Optimization Analysis

As described in Section III-C, this work optimizes measurement management as shown in Figure 6. The current adaptive forwarding design updates measurements at FIB entries on receiving each Data packet. Therefore, it introduces one FIB lookup, i.e., LPM in Data processing pipelines, which is significant overhead comparing to the pipelines without measurement that only involves exact name matching, i.e., PIT and CS (Section II-C).

With the proposed design, measurements are made along with PIT entries, so it introduces no extra name lookup. Assume 1%[2] traffic measures multiple next hops and generates

[2]Because lack of real traffic, this work is not intended to give a specific number of how much traffic that will measure multiple next hops, which can also be 0.1% or 10%.

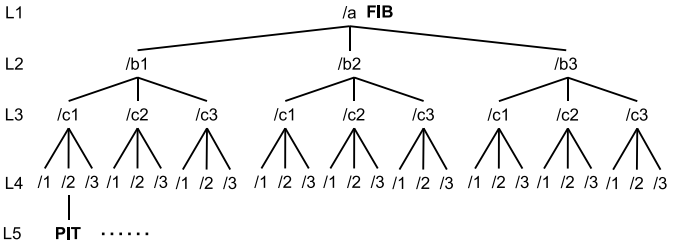


Fig. 16. A 5-depth 3-out-degree name tree.

a ranking list, then only 1% traffic triggers FIB lookup, which reduces FIB lookups to 1%.

Comparing both processing logics, existing implementation (i.e., the ASF strategy) updates MT on each Data reception. Because MT is bound with FIB, hence this is equivalent to adding a FIB lookup in Data processing. In our design, MT is bound with PIT, therefore their lookups can be merged into one, thus no extra lookup is added. FIB lookup is needed only when new routes ranking is observed, which means that the network situation is changed. As a result, FIB lookup is significantly reduced.

E. FIB Expanding Algorithms

Next, we evaluate the performance of three different FIB expanding algorithms, the Top-down expanding algorithm, the Bottom-up expanding algorithm, and the SS expanding algorithm (Section IV-B). Because the FIB collapsing algorithm (Section 12) is able to optimize the results of FIB expanding algorithms. This section evaluates the results of FIB expanding algorithms before the FIB collapsing algorithm is triggered.

The assumption is that during a short period of time, only a small portion of FIB names suffer from the Prefix Granularity Problem (e.g., less than 5%). The evaluation is made on one FIB entry, and it can be applied to the estimated number of FIB entries to evaluate the overall performance during a short period of time.

As analyzed in Section IV-B, two parameters determine the expanded FIB tree starting from one FIB root node, *the accurate name granularity level* and *the observed route ranking order*. Therefore, our idea is that to give a name tree with the root node as a FIB entry (e.g., `/a`) with a ranking list of next hops (e.g., $r1$), then we pick a name (e.g., `/a/b`) that uses a different route ranking (e.g., $r2$), and the traffic under this picked name will report their route ranking ($r2$), while the traffic not under the picked name will report the original route ranking ($r1$). Using this rule, we generate different traffic traces (i.e., PIT name and route ranking) in random orders. The proposed FIB expanding algorithms will be evaluated using these traffic traces.

More specifically, after studying the name tree design in two NDN applications [20], [21], we use a similar name tree structure, which is a 5-depth 3-out-degree name tree (Figure 16). This specific data structure is simple but good enough to represent the name tree design of several existing applications. Moreover, it is good enough to evaluate the performance of the proposed FIB expanding and collapsing algorithms.

The network situation: /a: r1 /a/b1/c1: r2

PIT Name	The Measured Route Ranking
/a/b1/c1/1/1/#seg=1	r2
/a/b1/c1/1/1/#seg=2	r2
/a/b1/c2/1/1/#seg=1	r1
/a/b1/c2/1/2/#seg=1	r1
/a/b1/c2/2/2/#seg=1	r1
/a/b1/c1/2/1/#seg=1	r2
/a/b1/c1/2/1/#seg=0	r2
…	…

Traffic with route ranking in random order set (1)

PIT Name	The Measured Route Ranking
/a/b1/c2/1/1/#seg=1	r1
/a/b1/c2/1/1/#seg=2	r1
/a/b1/c1/1/1/#seg=1	r2
/a/b1/c2/1/2/#seg=1	r1
/a/b1/c1/2/2/#seg=1	r2
/a/b1/c3/2/1/#seg=1	r1
/a/b1/c1/2/1/#seg=0	r2
…	…

Traffic with route ranking in random order set (2)

Fig. 17. An example of two sets of random traffic trace containing PIT names and the measured route ranking after a partial network failure.

The initial FIB entry (generated from routing protocols) is `/a` at level 1 with the route ranking $r1$. PIT names are generate at level 5 of the name tree. Then, we generate a new route ranking ($r2$) at different levels as a parameter (e.g., `/a`, `/a/b1`, `/a/b1/c1`, and `/a/b/c1/1`), which reflects the new route ranking at different name granularity after partial network failures. Next, we generate 5 sets of traffic traces with the deserved path performance measurements (i.e., a sequence of PIT name with route ranking) in the event of the given network failure in random orders. The ranking of PIT is either the route ranking at the original FIB or the newly added route ranking, depending on the longest prefix matching. For example, Figure 17 shows two random sequences of PIT name with its measured route ranking. The original route ranking was $r1$ at `/a`, and the new route ranking $r2$ was given to `/a/b1/c1`. Traffic traces within these namespaces should observe the longest prefix matched route ranking, which will be the input to our simulator.

The described scenario is implemented in a name-tree simulator[3] with FIB and measurement entries. The simulator is able to do FIB/MT insertion and lookup. In addition, a sequence of PIT name with measured ranking can be inputted, and the simulator is able to expand and collapse FIB with the proposed algorithms.

To quantify the expanded FIB tree, two metric are used:

- *The Number of Newly Inserted FIB Names*: the smaller this value is, the better the performance. First, newly inserted FIB entries increase the size of FIB table and write operations, thus the smaller the number of the inserted FIB entries is, the smaller the FIB operational overhead. Moreover, a newly inserted FIB entry indicates that a better forwarding path is found, which means a certain amount of Interests has been forwarded to a non-optimal path, hence the smaller of this value means less non-optimal forwarding.
- *The Average Height of Newly Inserted FIB Name in Name Tree*: the smaller this value is, the less FIB lookup overhead (Section II-C).

The simulation results are shown in Figure 18 and 19. The number of the newly inserted FIB entries inserted is predictable for both the Top-down and the Bottom-up expand-

[3]The simulator in implemented in Python with open source link: https://github.com/philoL/MT-simulator

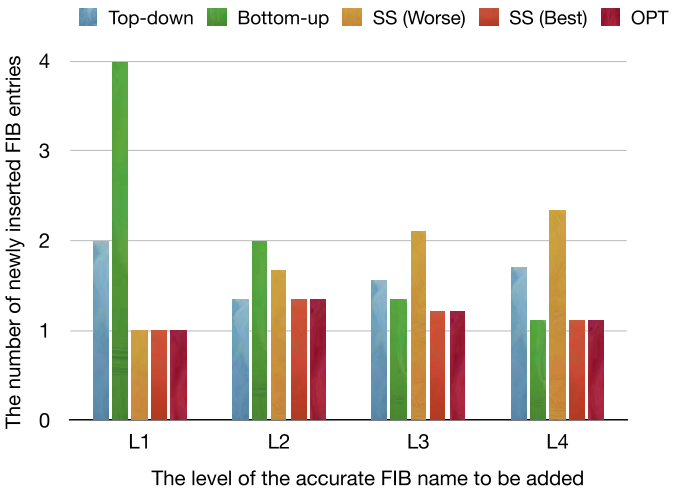


Fig. 18. The number of newly inserted FIB entries.

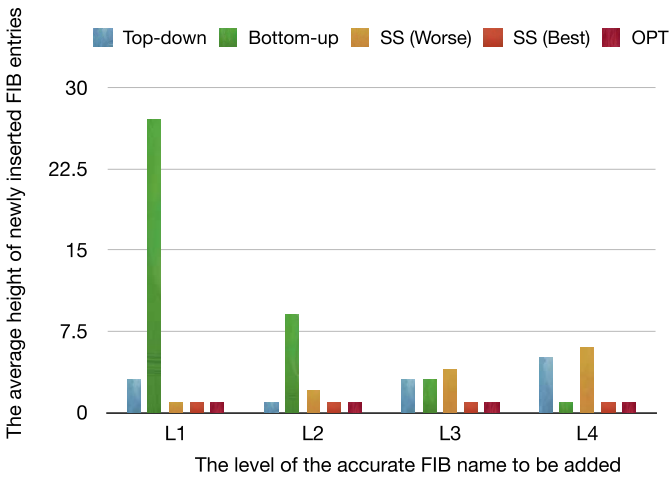


Fig. 19. The average height of newly inserted FIB entries.

ing algorithm no matter which order of measurements is fed. In general, if the new ranking is generated near the FIB name, the Top-down expanding algorithm generates less FIB entries than the Bottom-up expanding algorithm. On the contrary, if the new ranking is generated near the PIT name, the Bottom-up expanding algorithm generates less FIB entries than the Top-down expanding algorithm. However, the worst case for the Bottom-up expanding algorithm generates a big number of FIB entries, i.e., when the new ranking is at applied at level-1. This is because the tree has exponentially more nodes at higher levels. Overall, the Top-down expanding algorithm performs better than the Bottom-up expanding algorithm.

The SS expanding algorithm may generate a different number of FIB entries giving a different order of path performance measurements. The figures show both the best case and the worst case for the SS expanding algorithm. In the best case, the SS expanding algorithm generates the optimal (OPT) number of FIB entries. In the worst case, it generates slightly more entries than Top-down algorithm.

Figure 19 shows the average height of newly inserted FIB entries by the proposed FIB expanding algorithms. The results should be analyzed together with Figure 18. For example, after one FIB entry with height of 3 (`/a/b1/c1`) is newly inserted, PIT names under this subtree are infected by this FIB entry because of LPM. Combine the number of newly inserted FIB entries with their average height, we can speculate how many PIT names will incur more overhead on FIB lookups. The result shows that Bottom-up algorithm has the most overhead; the SS expanding algorithm has the least overhead (optimal) in the best case, but has similar overhead to Top-down algorithm in the worst case.

The SS expanding algorithm expands FIB into different tree results, depending on the order of measurements. The best case happens when another measurement path (first recorded) intersects with the new measurement path at the target name granularity, e.g., if the first measurement at `/a/b1/c1/d1/0` has ranking $r1$, and the second one at `/a/b1/c1/d2/0` has ranking $r2$, the SS expanding algorithm will insert FIB entry with ranking $r2$ at `/a/b1/c1/d2` as the best case. On the contrary, if the order changes, the SS expanding algorithm may update FIB at $/a$ with $r2$, and falls into the worst case that inserts more FIB entries.

Note that the SS expanding algorithm has more overhead than the other two algorithms on MT insertion, as each PIT updates leaves a MT entry on the whole path.

To conclude, all three algorithms are able to expand FIB to make better forwarding decisions than no expanding algorithms. In this evaluation, given a namespace similar to Figure 16 under a FIB entry, the Top-down expanding algorithm is better than the other two algorithms regarding the introduced overhead and its implementation simplicity. Note that an important assumption is that only a small portion of traffic triggers FIB expanding algorithms. Although the SS expanding algorithm can expand FIB with the minimum overhead in the best case, the SS expanding algorithm introduces massive overhead in MT update. One potential future study is to improve the SS expanding algorithm.

F. Implementation Evaluation

Because NFD is modular and uses the same tree data structure as the index for CS, PIT, FIB, and MT, thus adding new logic and operations on MT becomes simple with low-level APIs supported. Our implementation (Section V) is evaluated in three folds. First, we add unit tests for our added functions, thanks to NFD's testing framework. Second, our network simulations are conducted in ndnSIM, which uses the source code of NDN forwarding and management, to ensure the simulations are maximally realistic and can be reproduced in real environments. Therefore, our previous simulations have tested the implementation. Last, we reproduce the simple scenario (Fig. 2) using three connected virtual machines; each virtual machine runs the modified NFD. The results show that the implemented techniques are as effective as observed in the simulation.

VII. Conclusion

In NDN, route names can be the prefixes of hierarchical data names. Given the limited FIB size, route names are designed to be considerably shorter than Interest names. Therefore,

it leads to poor Interest routing locality in partial network failures. This problem is inherited in NDN adaptive forwarding. Adaptive forwarding is an important feature of NDN, that the forwarding plane is able to observe past data retrieval performance and use it to adjust future Interest forwarding. Because of the difference between route name and data name, there exits the Prefix Granularity Problem - what prefix length should be used to record path measurement. Existing adaptive forwarding designs fail to solve this problem as they use a static name prefix to record path measurement. This work tackles the Prefix Granularity Problem by dynamically disaggregating and aggregating FIB names, allowing path performance measurement to be recorded at a fine-grained name prefix granularity that best reflects the network situation. In addition, this work optimizes MT storage and operations in Data processing, by storing MT with PIT instead of FIB, and adding a filter for FIB updates, hence limiting FIB's the-longest-prefix-match lookups. Evaluation results show that the top-down FIB expanding algorithm is a good candidate to address the Prefix Granularity Problem in NDN adaptive forwarding, because of its simple implementation, low cost, and efficient expanding results. The challenge of proposing a new FIB expanding algorithm is to ensure its effectiveness while reducing its overhead. This work should be the beginning of more significant work in this area. Future work can add more comprehensive evaluation, including more complex network topologies, more realistic naming designs, and bigger traffic traces.

ACKNOWLEDGMENT

The authors are grateful for invaluable suggestions made by Ken Calvert and Lixia Zhang. Any findings, discussions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsor.

REFERENCES

[1] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, Jul. 2012.
[2] T. Liang, J. Shi, and B. Zhang, "On the prefix granularity problem in NDN adaptive forwarding," in *Proc. 7th ACM Conf. Inf.-Centric Netw.*, 2020, pp. 41–51.
[3] C. Pu, "Adaptive forwarding strategy based on MCDM model in named data networking," *TechRxiv*, 2020, doi: 10.36227/techrxiv.13296128.v1.
[4] V. Lehman *et al.*, "An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Service (IWQoS)*, Jun. 2016, pp. 1–10.
[5] W. Milliken, T. Mendez, and D. C. Partridge, *Host Anycasting Service*, document RFC 1546, Nov. 1993. [Online]. Available: https://rfc-editor.org/rfc/rfc1546.txt
[6] A. Afanasyev *et al.*, "NDNS: A DNS-like name service for NDN," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2017, pp. 1–9.
[7] NDN Project Team. *NDN Protocol Design Principles*. Accessed: Nov. 17, 2020. [Online]. Available: http://named-data.net/project/ndn-design-principles/
[8] W. So, A. Narayanan, and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables," in *Proc. Architectures for Netw. Commun. Syst.*, Oct. 2013, pp. 215–225.
[9] NDN Project Team. (2020). *NDN Packet Format Specification Version 0.3*. [Online]. Available: https://named-data.net/doc/NDN-packet-spec/current/interest.html
[10] J. Shi, D. Pesavento, and L. Benmohamed, "NDN-DPDK: NDN forwarding at 100 Gbps on commodity hardware," in *Proc. 7th ACM Conf. Inf.-Centric Netw. (ICN)*, 2020, pp. 30–40.
[11] C. Ghasemi, H. Yousefi, K. G. Shin, and B. Zhang, "On the granularity of trie-based data structures for name lookups and updates," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 777–789, Apr. 2019.
[12] M. Varvello, D. Perino, and J. Esteban, "Caesar: A content router for high speed forwarding," in *Proc. 2nd ed. ICN Workshop Inf.-Centric Netw.*, 2012, pp. 73–78.
[13] J. Shi, E. Newberry, and B. Zhang, "On broadcast-based self-learning in named data networking," in *Proc. IFIP Netw.*, 2017, pp. 1–9.
[14] T. Liang *et al.*, "Enabling named data networking forwarder to work out-of-the-box at edge networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.
[15] C. Ghasemi. (2019). *Visa NDN Video Streaming*. [Online]. Available: https://ivisa.named-data.net/
[16] A. Afanasyev *et al.*, "NFD developer guide," Dept. Comput. Sci., Univ. California, Los Angeles, NV, CA, USA, Tech. Rep. NDN-0021, 2014.
[17] J. Hartman. (2014). *NDN Testbed*. [Online]. Available: https://named-data.net/ndn-testbed/
[18] A. Afanasyev *et al.*, "NDNSIM: NDN simulator for NS-3," Univ. California, Los Angeles, NV, USA, Tech. Rep. 4, 2012.
[19] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proc. 3rd ACM Conf. Inf.-Centric Netw.*, Sep. 2016, pp. 21–30.
[20] T. Liang, J. Pan, and B. Zhang, "Ndnizing existing applications: Research issues and experiences," in *Proc. 5th ACM Conf. Inf.-Centric Netw.*, 2018, pp. 172–183.
[21] H. Zhang *et al.*, "Sharing mhealth data via named data networking," in *Proc. 3rd ACM Conf. Inf.-Centric Netw.*, 2016, pp. 142–147.

Teng Liang received the B.S. degree from Beijing Institute of Technology and the Ph.D. degree from The University of Arizona in 2020. He is currently an Post-Doctoral Researcher with Peng Cheng Laboratory. He also contributes to NFD and the NDN software forwarder codebase. His research interests focus on named data networking, including forwarding, deployment, applications, and security in challenging scenarios.

Junxiao Shi received the Ph.D. degree in computer science from The University of Arizona. He is currently a Guest Researcher with the Advanced Network Technologies Division, National Institute of Standards and Technology. His research interest includes named data networking.

Yi Wang (Member, IEEE) received the Ph.D. degree in computer science and technology from Tsinghua University in July 2013. He is currently a Research Associate Professor with the Sustech Institute of Future Networks, Southern University of Science and Technology. His research interests include future network architectures, information centric networking, software-defined networks, and the design and implementation of high-performance network devices.

Beichuan Zhang (Member, IEEE) received the B.S. degree from Peking University and the Ph.D. degree from UCLA. He is currently a Professor with the Department of Computer Science, The University of Arizona. His research interest is in internet routing architectures and protocols. He has been working on named data networking, green networking, and inter-domain routing systems. He received the Applied Networking Research Prize in 2011 by ISOC and IRTF and the Best Paper Awards at IEEE ICDCS in 2005 and IWQoS in 2014.