On the Prefix Granularity Problem in NDN Adaptive Forwarding

Teng Liang
The University of Arizona
philoliang@cs.arizona.edu
Peng Cheng Laboratory
liangt@pcl.ac.cn

Junxiao Shi
National Institute of Standards and Technology (NIST)
junxiao.shi@nist.gov

Beichuan Zhang
The University of Arizona
bzhang@cs.arizona.edu

ABSTRACT

One unique architectural benefit of Named Data Networking (NDN) is **adaptive forwarding**, i.e., the forwarding plane is able to observe data retrieval performance of past Interests and use it to adjust forwarding decisions for future Interests. To be effective, adaptive forwarding assumes ***Interest Routing Locality***, meaning that Interests sharing the same prefix are likely to follow a similar forwarding path within a short period of time. Therefore, past observations can provide insight into how forwarding will likely perform for the same prefix in the near future. 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 measurements - we refer to this as the ***Prefix Granularity Problem***. The longer the common prefix is, the better Interest Routing Locality. However, finer grained-prefixes cover fewer Interests each and require a larger forwarding table. Existing adaptive forwarding designs use a static prefix length, which is known to encounter issues 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 longest prefix matching during the processing of Data packets. Simulations demonstrate that the proposed techniques can result in better forwarding decisions in the event of partial network failures with significantly reduced overhead.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Data path algorithms**; *Network measurement*;

KEYWORDS

Information-centric networking (ICN); Named-Data Networking (NDN); Adaptive forwarding; Prefix Granularity Problem

ACM Reference Format:
Teng Liang, Junxiao Shi, and Beichuan Zhang. 2020. On the Prefix Granularity Problem in NDN Adaptive Forwarding. In *ACM Conference on Information-Centric Networking (ICN '20), September 29–October 1, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3405656.3418712

ICN '20, September 29–October 1, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8040-9/20/09…$15.00
https://doi.org/10.1145/3405656.3418712

1 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 [13], 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 what we call ***Interest Routing Locality***, 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***. The challenge is to balance the trade-off between Interest routing locality and forwarding table size.

Existing adaptive forwarding designs tackle the prefix granularity problem by choosing a static Interest name prefix length, typically using the route name, to record path performance measurements from past data retrievals. However, this design encounters difficulties in handling partial network failures. Specifically, NDN route names are considerably shorter than Interest names,

because 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. Therefore, maintaining path performance measurements at the coarse granularity of route names contributes to poor Interest routing locality. 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 the forwarding table, i.e., disaggregating and aggregating name prefix in the forwarding table on the fly. Specifically, once the forwarding plane detects that the traffic under a route name has several different optimal forwarding paths, it will disaggregate the route name into several sub-namespaces, expanding the 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 unpredictable order, the proposed expanding algorithm may create unnecessary sub-namespaces. To address this issue, we also propose a forwarding table collapsing algorithm.

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 the exact match name lookup that is used when adaptive forwarding is not in use.

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 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 nexthop 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 propose and evaluate three FIB expanding and FIB collapsing techniques.
- We optimize the current data retrieval performance measurement process in adaptive forwarding, which significantly 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 2 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 3, which is followed by design details (Section 4). We evaluated our proposed solution in Section 5. Finally, Section 6 concludes the paper.

2 PROBLEM STATEMENT

2.1 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 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 cached Data that satisfies the Interest 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 the forwarding strategies can make forwarding decisions. Forwarded Interests are buffered in the PIT, allowing Data packets to be returned to the consumer 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 [13]. 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, improving data retrieval performance.

In the current adaptive forwarding design [13] 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 2.3.

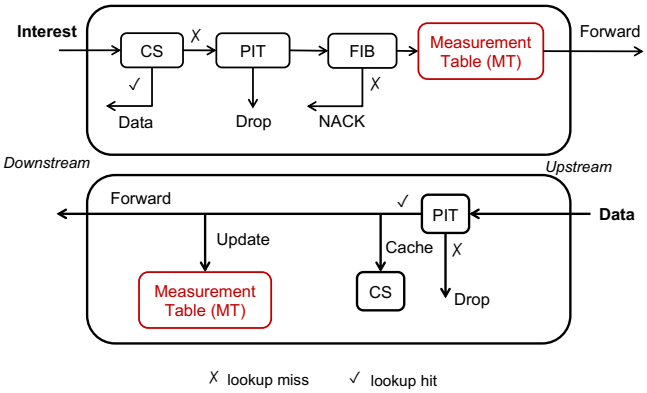


Figure 1: The current packet processing workflow with measurement management

2.2 Prefix Granularity Problem

Next, we introduce the Prefix Granularity Problem derived from NDN adaptive forwarding, and the limits of the current designs in solving this problem. The problem is elaborated using an example shown in Figure 2, in which two consumers $C1$ and $C2$ are connected to two producers (applications that serve data) $P1$ and $P2$, and a data repository $P3$, through routers $R1$ and $R2$.

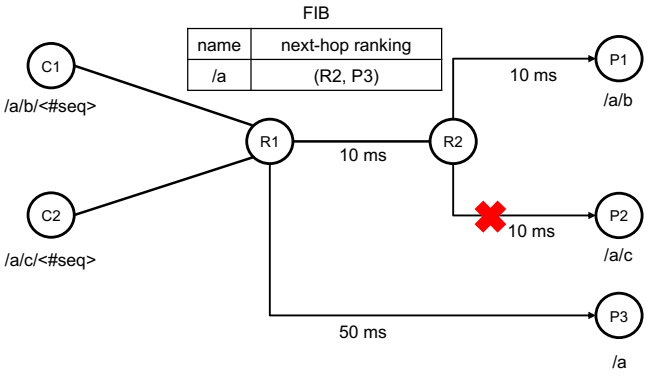


Figure 2: An example to demonstrate the FIB Prefix Granularity Problem

In this example, $P1$ and $P2$ are producers that serve data under name prefixes `/a/b` and `/a/c` respectively. $P3$ is a data repository that serves data under a more general name prefix `/a`, but it is connected over a link with much higher RTT. In the routing protocol, router $R2$ aggregates prefix registrations from $P1$ and $P2$ into a single announcement of `/a`. Router $R1$ has a routing entry `/a` with the initial next hop ranking such that $R2$ is preferred to $P3$.

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 $P1$, and Interests from $C2$ to $P2$, because $R1$ picks $R2$ as the best next hop. Then, the network link between $R2$ and $P2$ fails. If link layers can quickly detect the link failure, and inform the network layer, routing protocols will be triggered and routers will quickly receive new routing announcements. Accordingly, $R1$ will forward Interests from $C2$ to $P3$, because $P3$ becomes the only next hop for `/a/c`. In other cases where lower-level detection is unavailable, network layers can rely on routing protocols' periodic keep-alive messages to detect failures, which usually takes seconds or even tens of seconds. NDN's adaptive forwarding can quickly react to link failure without relying routing protocols.

With NDN adaptive forwarding, when the network link between $R2$ and $P2$ fails, $R1$ ideally should start forwarding Interests from $C2$ to $P3$, because the Interest timeouts resulted from the link failure would move $R2$ to a lower ranking in NDN adaptive forwarding. However, the current adaptive forwarding design, such as the ASF strategy, records next hop performance measurements at FIB entry level, i.e. `/a`. The link failure results causes the `/a` prefix to either keep the existing ranking ($R2$, $P3$) or change to ($P3$, $R2$). 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 $P3$, which has a higher round-trip time than retrieving from $P1$ via $R2$.

This scenario is simulated with ASF strategy in Section 5.1. The simulation result matches the first aforementioned case. After the link failure, $R1$ continues to forward Interests from $C2$ to $R2$, even if no data can be retrieved from this path. This is because the Interest-Data flow between $C1$ and $P1$ is still working after the link failure, and ASF measures RTT at FIB name `/a`, so it still considers $R2$ to be a better choice than $P3$ for the name prefix `/a`.

Intuitively, the demonstrated problem can be solved by the current adaptive forwarding if $R2$ does not aggregate `/a/b` and `/a/c` into a single routing announcement `/a`, which can help $R1$ to learn the fine-grained route names. However, another partial network failure may occur that affects a longer route name (e.g., `/a/c/d`), then the existing route name `/a/c` still suffers from the name prefix granularity problem.

The root problem is that the performance measurement collected from each Interest-Data exchange only indicates the route situation for a specific data name, but the ranking change can be applied to any prefix granularity between data name and FIB name. The ASF strategy records the ranking at FIB name granularity, which can be inaccurate, thus making wrong forwarding decisions and limiting the network performance.

2.3 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 [6]. 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, the current adaptive forwarding plane design records measurements along with the 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 [10].
- **PIT lookup** conducts *Exact Match* against Interest name.
- **FIB lookup** conducts *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 the current adaptive forwarding design decides to take measurements bound 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 [11]. To further improve this lookup performance, PIT token [9] 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 *Longest Prefix Match (LPM)* against Data name to find a FIB name, since measurements are bound with FIB name. This lookup is the bottleneck of Data processing.

To summarize, without sacrificing 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, so as to significantly reduce LPM in updating MT (Section 3.3).

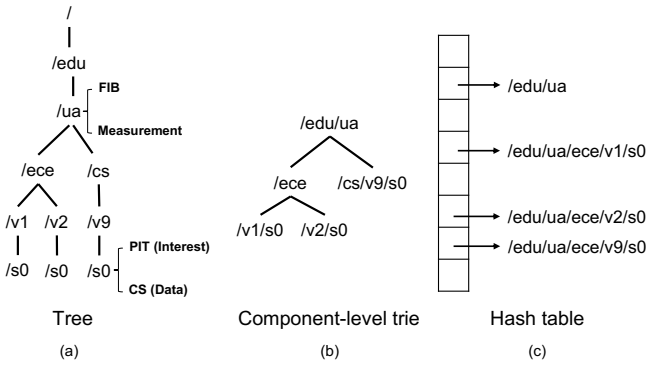


Figure 3: Three data structures to store name-based entries

Another major research topic is to design data structures to improve name lookup performance, such as to use trie [3], hash table [10] and Bloom filter [12]. Figure 3 gives an example of three different data structures. Regarding the tree data structure, LPM require 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* [10] 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 from any data structure.

3 DESIGN RATIONALE

In this section, we consider the design rationale of our FIB expanding and FIB collapsing algorithms.

3.1 Dynamic FIB Expanding

As elaborated in Section 2.3, 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. We believe that this problem can be solved by dynamically 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. We refer to this solution as dynamic FIB expanding.

Figure 4 gives three examples of FIB update after observing the past path performance measurements. Example (1) updates the nexthop ranking on FIB name directly, which is how existing adaptive forwarding designs work. Examples (2) and (3) update the nexthop 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

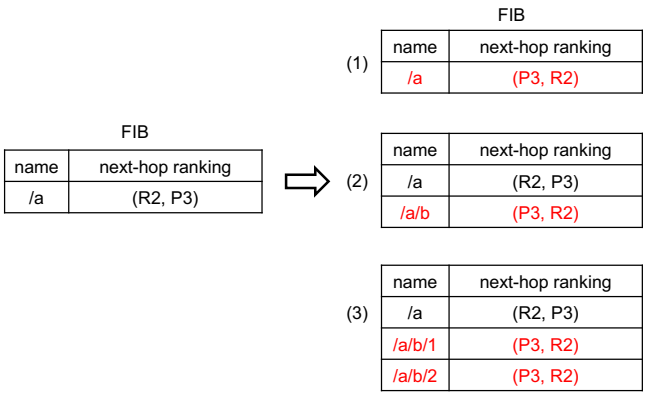


Figure 4: Three examples to update FIB after measuring the nexthop ranking of multiple paths: (1) no FIB expanding, (2) FIB expanding with the accurate name prefix, and (3) FIB expanding with the name prefix longer than the accurate one

component, while example (3) expands it with 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 a priori, 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 4) and evaluated (Section 5) by the aforementioned metrics.

3.2 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 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.

3.3 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 2.3). 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 5. 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.

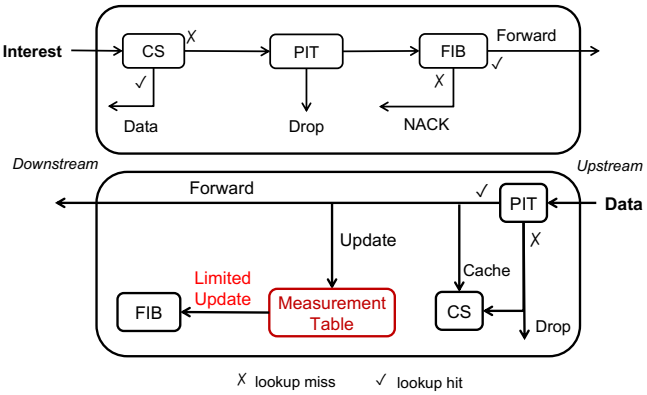


Figure 5: Optimizing measurement management in the packet processing pipelines

3.4 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 [8]. Moreover, the FIB expanding algorithm could generate more FIB entries than necessary, such as the example shown in Figure 6.

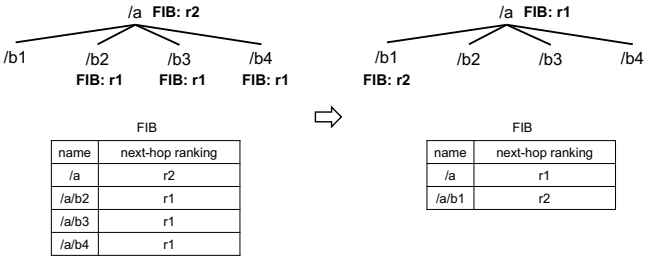


Figure 6: An example of a FIB Collapsing algorithm

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 6 demonstrates that the FIB collapsing algorithm can consolidate four FIB names into two without changing the 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 4.

3.5 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.

4 DESIGN DETAILS

4.1 FIB Expanding Detection

As analyzed in Section 3.2, upon receiving a Data packet, the adaptive forwarding plane collects measurements regarding the incoming face on the specific Interest name, and records the measurement on 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 of a single path. The single path performance measurement is unable to generate a route ranking, thus will not trigger FIB expanding.

As introduced in Section 2.1, 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.

4.2 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 7 shows a name hierarchy, which has an existing FIB entry `/a` with 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.

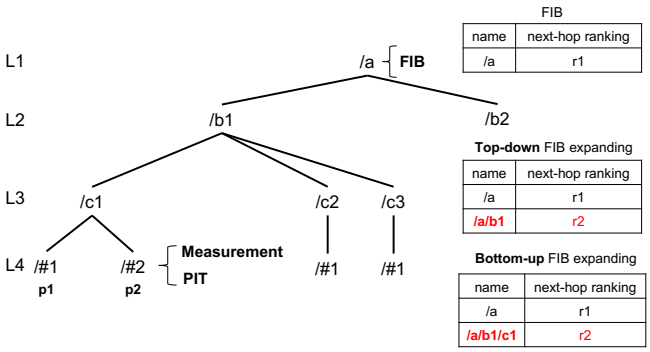
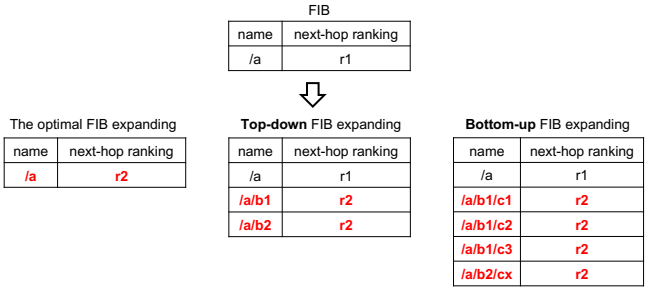


Figure 7: An example to explain FIB expanding algorithms

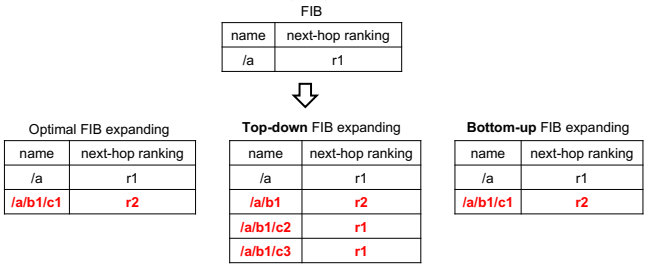
Top-down Expanding Algorithm. This algorithm creates a new FIB entry at FIB+1 level and records the new ranking. In Figure 7, the top-down expanding algorithm would create a FIB entry at `/a/b1`, one level down ($L2$) from the existing FIB entry ($L1$), and record the ranking $r2$ on this FIB entry. The name "top-down" reflects the behavior that it starts from the "top", which is the existing FIB entry.

Bottom-up Expanding Algorithm. This algorithm creates a new FIB entry at PIT-1 level and records the new ranking. In Figure 7, 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 8 demonstrates two examples of FIB table expanded by these two algorithms. In Figure 8a, 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 8b, 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.



(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

Figure 8: Expanded FIB results generated by the top-down expanding algorithm and the bottom-up expanding algorithm

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.

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.

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 9, 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`,

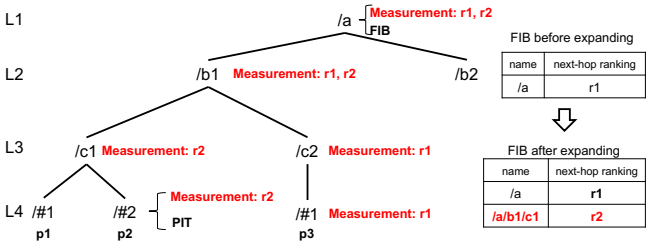


Figure 9: An example of FIB expanding using the SS expanding algorithm

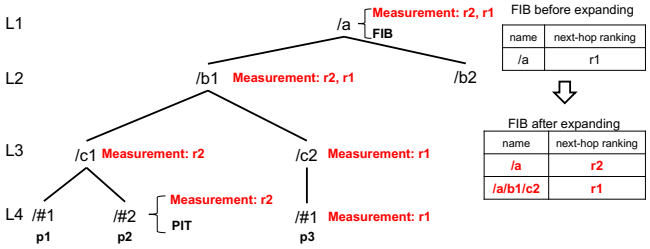


Figure 10: An example of FIB expanding using the SS expanding algorithm with a different measurement order

`/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 10, 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 (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.

4.3 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 3.4, 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.

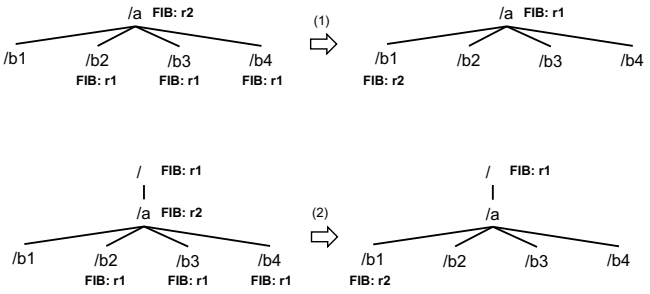


Figure 11: Two FIB collapsing scenarios

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 11, 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 has the most popular ranking. In Figure 11, FIB entries `/a/b2`, `/a/b3`, and `/a/b4` are removed.
(2) Add FIB entries to children that shares the current ranking of the parent node. In Figure 11, 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 11 (top), the FIB entry `/a` is updated with ranking $r1$. In Figure 11 (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 ancestors until no collapsing is needed.

5 EVALUATION

The evaluation is done in two stages. First, the scenario described in Section 2 is simulated to demonstrate both the limitation of the current adaptive forwarding, and the effectiveness of the FIB expanding in solving the Prefix Granularity Problem. The second sets of experiments are designed to compare the effectiveness and efficiency of the proposed FIB expanding and the FIB collapsing algorithm.

5.1 The Necessity of FIB Expanding

The topology shown in Figure 2 is simulated in ndnSIM [1], a widely used simulator for NDN simulations, which is built on top of NS-3 with NFD [2] ported. Each test runs five times with different seeds input. For consumers, PCON [7] consumer application is used with CUBIC congestion control scheme configured. Three producers are serving names prefixes shown in the figure. The throughput bottleneck for both consumers are 10 Mbps. The NFD on $R1$ is configured with the ASF strategy. All applications start running at $0s$. A link failure happens between $R2$ and $P2$ at $6s$. Data retrieval rates are measured at both consumers to demonstrate the how much application performance is improved by our proposed FIB expanding algorithms. The ASF adaptive forwarding strategy is modified with two versions, no FIB expanding (the current code), and the FIB expanding with the hard-coded accurate name prefix.

As shown in Figure 12a, the existing adaptive forwarding strategy (i.e., ASF) has no FIB expanding, which 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 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 $C2$.

In contrast, Figure 12b 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 $P3$.

5.2 Measurement Management Optimization Analysis

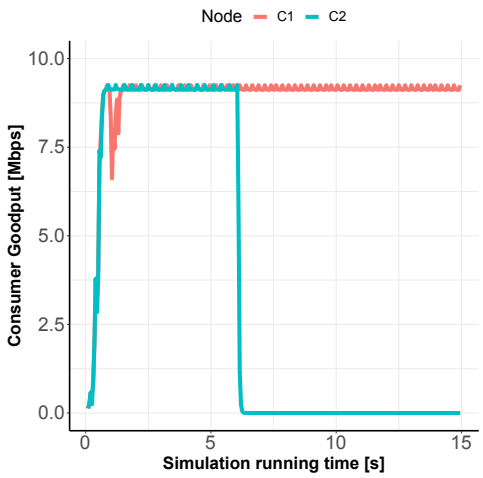
As described in Section 3.3, this work optimizes measurement management as shown in Figure 5. The current adaptive forwarding design updates measurements at FIB entries on receiving each Data packet. Therefore, it introduces one FIB lookup, i.e., LPM in Data processing pipelines, which is significant overhead comparing to the pipelines without measurement that only involves exact name matching, i.e., PIT and CS (Section 2.3). With the proposed design, measurements are made along with PIT entries, so it introduces no extra name lookup. Assume 1% [1] traffic measures multiple next hops and generates a ranking list, then only 1% traffic triggers FIB lookup, which reduces FIB lookups to 1%.

5.3 FIB Expanding Algorithms

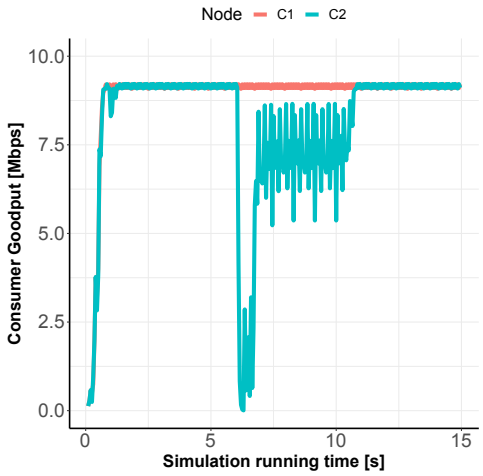
Next, we evaluate the performance of three different FIB expanding algorithms, the Top-down expanding algorithm, the Bottom-up expanding algorithm, and the SS expanding algorithm (Section 4.2). Because the FIB collapsing algorithm (Section 11) is able to optimize the results of FIB expanding algorithms. 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

[1] 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%.



(a) No FIB expanding



(b) FIB expanding with accurate name prefix

Figure 12: 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

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 4.2, 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.

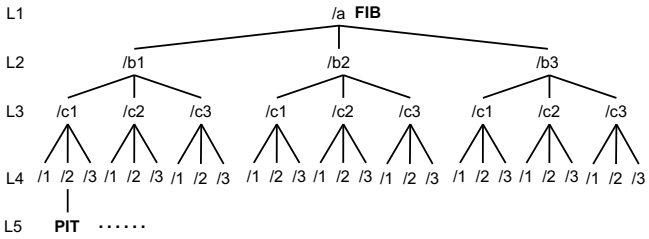


Figure 13: A 5-depth 3-out-degree name tree

More specifically, after studying the name tree design in two NDN applications [5] [14], we use a similar name tree structure, which is a 5-depth 3-out-degree name tree (Figure 13). 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 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 14 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 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)

Figure 14: An example of two sets of random traffic trace containing PIT names and the measured route ranking after a partial network failure

The described scenario is implemented in a name-tree simulator [2] with FIB and measurement entries. The simulator is able to do FIB/MT insertion and lookup. In addition, a sequence of PIT name

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

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 of this value provides better performance. First, newly inserted FIB entries increase the size of FIB table and write operations, thus the smaller number of the inserted FIB entries, the smaller FIB overhead introduced. 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 of this value means less overhead on FIB lookup, since the longer FIB name prefix the more FIB lookup operations needed (Section 2.3).

The simulation results are shown in Figure 15 and 16. The number of the newly inserted FIB entries inserted is predictable for both the Top-down and the Bottom-up expanding 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 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 significantly 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 worse 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.

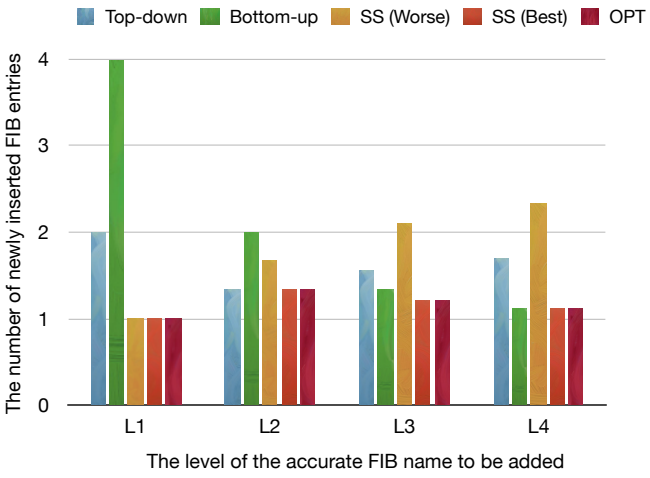


Figure 15: The number of newly inserted FIB entries

Figure 16 shows the average height of newly inserted FIB entries by the proposed FIB expanding algorithms. The results should be analyzed together with Figure 15. 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 worse 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.

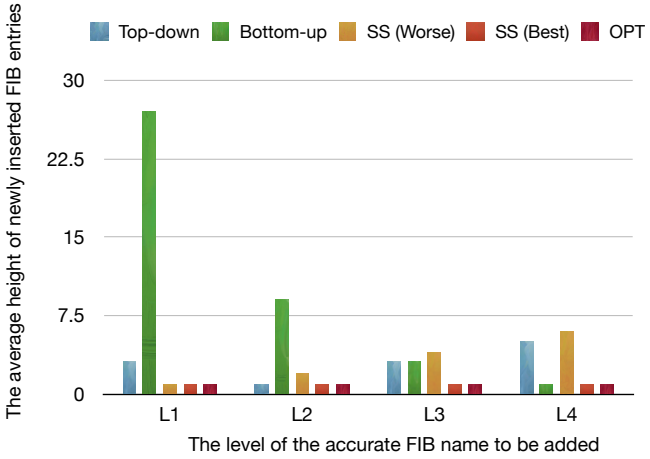


Figure 16: The average height of newly inserted FIB entries

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 13 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 minimum overhead in its best case, the SS expanding algorithm introduces massive overhead in Measurement Table update. One potential study is to improve the SS expanding algorithm.

6 CONCLUSIONS

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. While it brings adaptiveness to the forwarding plane, it introduces the Prefix Granularity Problem - what prefix length should be used to record path measurement. Existing adaptive forwarding designs fails to solve this problem as they use a static name prefix to record path measurement. This work tackles the Prefix Granularity Problem by proposing dynamic FIB expanding and collapsing algorithms, which dynamically disaggregate and aggregate 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 optimize measurement management in Data processing pipelines, by binding measurement to PIT instead of FIB, significantly reduces the usage of longest prefix match, which is the performance bottleneck of Data packet processing. Simulation results show that the top-down FIB expanding algorithm combined the FIB collapsing algorithm is a good candidate in simple scenarios, because they can are simple to implement, achieve good results in expanding FIB, and involve small overhead. 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 richer network topologies, more complex naming problems, and a bigger traffic trace.

7 ACKNOWLEDGMENTS

We are grateful for invaluable suggestions made by the Shepherd, Ken Calvert, and all comments from anonymous reviewers. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1629009. It was also supported by National Key R&D Program of China (2019YFB1802600), the Key-Area Research and Development Program of Guangdong Province (No.2019B121204009), FANet: PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (No. LZC0019), and the Guangdong Basic and Applied Basic Research Foundation (No. 2019B1515120031). 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] Alexander Afanasyev, Ilya Moiseenko, Lixia Zhang, et al. 2012. ndnSIM: NDN simulator for NS-3. *University of California, Los Angeles, Tech. Rep* 4 (2012).

[2] Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, et al. 2014. NFD developer guide. *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021* (2014).

[3] Chavoosh Ghasemi, Hamed Yousefi, Kang G Shin, and Beichuan Zhang. 2019. On the Granularity of Trie-Based Data Structures for Name Lookups and Updates. *IEEE/ACM Transactions on Networking* 27, 2 (2019), 777–789.

[4] Vince Lehman, Ashlesh Gawande, Beichuan Zhang, Lixia Zhang, Rodrigo Aldecoa, Dmitri Krioukov, and Lan Wang. 2016. An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[5] Teng Liang, Ju Pan, and Beichuan Zhang. 2018. NDNizing existing applications: research issues and experiences. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*. 172–183.

[6] NDN Project Team. [n. d.]. NDN Protocol Design Principles. http://named-data.net/project/ndn-design-principles/.

[7] Klaus Schneider, Cheng Yi, Beichuan Zhang, and Lixia Zhang. 2016. A practical congestion control scheme for named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 21–30.

[8] Junxiao Shi, Eric Newberry, and Beichuan Zhang. 2017. On broadcast-based self-learning in named data networking. In *IFIP Networking*.

[9] Junxiao Shi, Davide Pesavento, and Lotfi Benmohamed. 2020. NDN-DPDK: NDN Forwarding at 100 Gbps on Commodity Hardware. In *7th ACM Conference on Information-Centric Networking (ICN 2020)*.

[10] Won So, Ashok Narayanan, and David Oran. 2013. Named data networking on a router: Fast and DoS-resistant forwarding with hash tables. In *Architectures for Networking and Communications Systems*. IEEE, 215–225.

[11] NDN Team. 2020. *NDN Packet Format Specification version 0.3.* https://named-data.net/doc/NDN-packet-spec/current/interest.html

[12] Matteo Varvello, Diego Perino, and Jairo Esteban. 2012. Caesar: A content router for high speed forwarding. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*. 73–78.

[13] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. 2012. Adaptive forwarding in named data networking. *ACM SIGCOMM computer communication review* 42, 3 (2012), 62–67.

[14] Haitao Zhang, Zhehao Wang, Christopher Scherb, Claudio Marxer, Jeff Burke, Lixia Zhang, and Christian Tschudin. 2016. Sharing mhealth data via named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM, 142–147.