2017 ACM/IEEE Symposium on Architectures for Networking and Communications

Controlling Strategy Retransmissions in Named Data Networking

Hila Ben Abraham & Patrick Crowley
Washington University in Saint Louis
Department of Computer Science and Engineering
{hila, pcrowley}@wustl.edu

ABSTRACT

Named Data Networking (NDN), an information-centric Internet architecture, contains a new architectural component named the strategy layer. This component introduces a new forwarding model, in which a forwarding strategy decides how to forward an Interest packet. In NDN, an application can pair its namespace to use a specific forwarding strategy in the local host, but has no control over the strategies used in remote routers. Despite the central role the forwarding strategy plays, its interaction with applications has not been explored or well understood.

In this paper we study and decompose the core mechanisms of a forwarding strategy in NDN. We illustrate how the correctness of some NDN applications can be affected by the coupling between the application design and the strategy decision to retransmit an unsatisfied Interest. This coupling creates challenges for application developers, who must implement their fixed application logic on a variable forwarding mechanism, and can lead to failure of application correctness and performance. We propose a new retransmission abstraction that decouples this strategy mechanism from the application design, and differentiates application Interests from network retransmissions. This allows every application to determine its own retransmission policy. We show that in some use cases the proposed abstraction can maintain continuous traffic flow regardless of the strategy used.

1. INTRODUCTION

In the last few years, as interest in future information-centric network (ICN) architectures has increased, we have witnessed continuous growth in research on the design and development of ICN architecture prototypes. Two on-going research projects, Named Data Networking (NDN) [1] and Content-Centric Networking (CCN) [2], provide two corresponding software prototypes: the NDN forwarder (NFD) [3, 4], developed by the NDN research group, and the CCNx project [5], developed by PARC. While the statements in this paper are true for both architectures, we focus our discussion on the NDN architecture as implemented in NFD.

There are two new packet types in NDN: an Interest packet to request a chunk of content, and a Data packet that retrieves it. Unlike the traditional IP forwarding, NDN uses a dynamic forwarding plane named the *strategy layer*. This architectural module contains multiple forwarding strategies, and each relies on a different set of input considerations to implement a specific forwarding behavior. This allows a spectrum of strategies to co-exist under the umbrella of the NDN architecture, and provides flexible forwarding behavior that can be manipulated according to the network requirements. For instance, ad-hoc networks can use a dynamic routing-less forwarding strategy [6], while a more traditional routing-based forwarding strategy can be used in core networks [7, 8]. To determine the strategy used, the router uses per packet name-based strategy selection, which means that the strategy is selected dynamically according to the incoming packet name. While supporting multiple forwarding strategies presents new advantages for ICN, it also poses new challenges for application developers.

In NDN, an application developer can pair the application namespace with a specific forwarding strategy in the local host. However, as a network layer component, the forwarding strategy must consider network characteristics, such as link bandwidth, routing preferences, and congestion policies. These and other network characteristics can change between routers, and therefore there is no guarantee that the same strategy paired with the application namespace on the local host would also be the one running in all remote routers. To avoid conflicts created by the use of different strategies, the forwarding strategy must not couple any application-specifics in its implementation. However, as we illustrate in this paper, one strategy mechanism, the decision whether to retransmit an unsatisfied Interest, has various possible outcomes, and each has different implications for the application's correctness.

Every forwarding strategy in NFD-based NDN architecture has three types of responses when an Interest is not satisfied: 1) drop the Interest, 2) retransmit the Interest, possibly on another face, and 3) send an upstream negative acknowledgment (NACK). This unpredictable response can have a substantial impact on application developers, who are required to apply a fixed application retransmission logic on a variable network behavior. While some applications request control of all retransmissions, others require that the strategy retransmit an Interest when it is not satisfied. Therefore, if the strategy chooses one retransmission policy when the application expects a different one, the performance and correctness of the application can be adversely affected.

One possible approach to address this problem is directing all applications to always implement their own retransmission mechanism, or to use an application framework that does it for them [9], and to avoid strategy retransmissions. While this approach guarantees correctness, it eliminates one of the biggest ICN advantages – the ability for in-network recovery. Unlike IP, NDN suggests a new premise of channel-less hop by hop forwarding, in which the Interest and Data packet follow the same but reversed paths. Due to this symmetric forwarding characteristic, an intermediate router can identify a failure and redirect the packet towards another potential path, with no need for the sender to discover a loss and re-express the Interest. This recovery capability has not just proven to provide an efficient multi-path forwarding [10], but as shown in this paper, can guarantee the correctness of some multiple producer applications. Thus, our proposed solution does not eliminate strategy retransmissions, but provides the application with the ability to control them.

While the retransmission policy is only one characteristic of a much more complex forwarding strategy, it is crucial. Therefore, in this paper we propose a strategy abstraction in which each strategy is implemented to support both types of applications: applications that request control of all retransmissions, and applications that rely on strategy retransmissions. This proposed abstraction, achieved by adding a new field to the Interest packet, allows the application to specify whether a network retransmission is required, and requires the strategy to determine only how the application requirement will be achieved. In other words, we suggest that the application determines the policy, while the strategy determines the mechanism. Thus, the usage of different strategies does not cause a variable retransmission outcome, and application correctness can be guaranteed. The contributions of this paper are as follows:

- We study and decompose the main mechanisms of a forwarding strategy in NDN.
- We identify the importance of the retransmissions mechanism by illustrating its impact in different use cases.
- We propose and evaluate a new forwarding strategy flag that decouples an application's expectation for in-network retransmission from the implementation details of any specific strategy.
- We propose another new flag that allows application and network-level retransmission interests to be treated differently inside the network, enabling otherwise inefficient operations like face probing for dynamic path discovery.

The remainder of this paper is organized as follows. The next section introduces background and related work. In section 3 we decompose the strategy mechanisms. Section 4 illustrates the importance and impact of the failure recovery mechanism on applications, using three use cases. Section 5 presents the details of our proposed decoupling retransmission mechanism. Section 6 provides evaluation results of the proposed mechanism. Section 7 summarizes our work and discusses future work.

2. BACKGROUND AND RELATED WORK

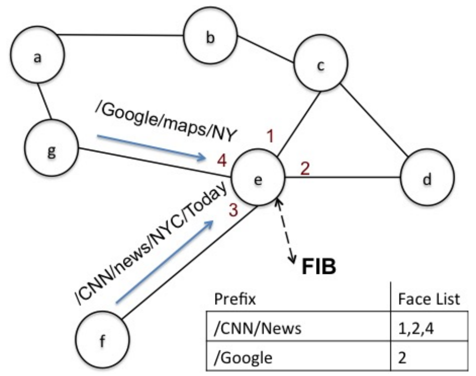
2.1 Named Data Networking

NDN is a consumer driven architecture that uses *Interest* and *Data* packets to request and retrieve content. To request a content item, a consumer expresses an Interest packet. The packet is forwarded in the network until it arrives at a router that can satisfy the request, either by retrieving the data from its local Content Store (CS) or by retrieving it from a local application that serves as the content producer. Then the content is returned to the consumer in a Data packet that follows the reverse path of the Interest packet. When a router determines that it cannot satisfy an incoming Interest packet, it searches for the next hop in its Forwarding Information Base (FIB) table. Before forwarding the interest to the determined next hop, the router registers the Interest packet and its incoming face in the Pending Interest Table (PIT). The PIT is used later to identify retransmissions of the same interest and to aggregate similar requests from additional consumers. In addition, when returning a Data packet to the consumer(s), the router uses the information in the PIT as breadcrumbs to follow the reverse path of the Interest packet(s).

To detect loops, every Interest packet carries a nonce generated by the application. When an incoming Interest packet contains the same name and nonce as previously recorded in the PIT, the Interest is detected as duplicated and dropped by the router. In recent implementations of the NDN forwarder, the router responds with an upstream negative acknowledgment (NACK) when a duplicated Interest is detected.

Each NDN packet is encoded in a Type-Length-Value (TLV) format that provides a dynamic platform for adding new fields to either the Interest or the Data packet. Our proposed solution for decoupling strategy retransmissions uses this flexible encoding by adding a new TLV to the interest packet.

2.2 NDN Forwarding Plane



Prefix	Face List
/CNN/News	1,2,4
/Google	2

Figure 1: NDN Forwarding Information Base.

As in the IP architecture, the NDN router uses the information in its FIB table to determine the packet's next hop. While a FIB entry in the IP architecture consists of an IP address and one port, each entry in the NDN FIB consists of a namespace and a list of possible faces. Each face represents an interface to a possible next hop, which can be a remote NDN entity or a local application. When the faces list consists of only one face, the Interest is forwarded on this face. However, when the faces list contains more than one face, the forwarding plane needs to decide on which face(s) to forward the interest. The forwarding decision is determined by the selected *forwarding strategy* of the requested namespace. Therefore, when forwarding an Interest, the NDN router performs two operations: 1) A FIB lookup to find the longest prefix match of the requested name. 2) A selection of one or more face(s) to be the interest's next hop(s).

Figure 1 shows a network and the FIB table in node e. In this example, when e receives an interest for /Google/maps/NY, it can forward it only on face number 2 towards d. However, e can choose from a list of faces when receiving an interest for /CNN/news/NYC/Today. In this case, after finding the correct FIB entry, e follows the forwarding strategy paired with /CNN/news namespace to decide on which face(s) the Interest should be forwarded.

2.3 Forwarding Strategies in Related Work

In this section we describe the details of the strategies implemented in NFD 0.4, and in CCNx version 0.82, and we briefly discuss the forwarding strategies in related work.

The first forwarding strategy implemented in ICN was the *default CCNx* strategy, which is also known as the *ncc* strategy in NFD. In this strategy, the NDN router forwards a received Interest packet on one face and waits for a Data packet to be returned. If the packet arrives within a specific predicted time set by the strategy, usually referred to as the *prediction* timer, then the face is remembered as the "best" face, and it is used to forward future interests of the same name. If the prediction timer expires before the arrival of a Data packet, the strategy retransmits the interest again to another available face. The CCNx default strategy is distinctive in the way it adjusts its prediction timer. Every time a Data packet is returned on the selected best face, the predicted wait time is adjusted down, so the prediction timer will expire faster the next time. When the Interest is not satisfied within the predicted wait time, the prediction timer is adjusted up. Thus, the strategy tries another available face whenever the prediction timer is too short to allow a successful response from the previously working face. When that happens, the predicted time is adjusted up again to allow the new face to respond with data. Thanks to this mechanism, the strategy timer approaches the actual round trip times after an initial exploration phase. In addition, this mechanism guarantees that other faces will be eventually given a chance to satisfy a namespace.

The *best-route* strategy, also used in NFD 0.4, is the default strategy for new applications and the gateway routers in the NDN testbed [11]. In *best-route*, every Interest packet is forwarded to the cheapest face, which is determined according to the cost assigned by the routing protocol. The named-data link state routing protocol (NLSR) [12] is currently the routing protocol configured to work with the best-route strategy on the testbed. When the face fails to respond on time, the strategy drops the Interest, and the application can choose whether to retransmit the Interest again. The strategy decides whether to suppress or to forward the application retransmission on a different face. This decision is made by a suppression timer set by the strategy. The suppression timer algorithm has changed several times in recent NFD versions. The *best-route* strategy keeps sending future Interests on the same face as long as that face has the cheapest cost, regardless of its success in returning the requested data. If the face is unresponsive, the routing protocol might delete the face from the FIB table [13].

The *loadsharing* strategy as implemented in CCNx 0.82, follows the same logic and forwards the Interest to the best available face selected according to feedback received in previous transmissions.

The *multicast* strategy, as implemented in NFD 0.4, forwards the Interest packet to all the available faces simultaneously. If there is no available face to forward the packet on, the strategy replies with a NACK packet. This strategy is similar to the *parallel* strategy implemented in CCNx 0.82.

The *access* strategy trades off between the best-route and multicast strategies. It first learns which next hop can satisfy an Interest by multicasting it to all possible next hops. The first upstream face to respond with the content is then remembered and used for future Interests. If the preferred upstream face later fails to satisfy an Interest with a similar name, a retransmission triggered by the consumer causes the strategy to start a new discovery phase by multicasting the Interest again.

The principles of an adaptive forwarding strategy are discussed in [14], and the details of such a strategy, the *GreenYellowRed* strategy, are described in [7]. A dynamic forwarding mechanism designed to discover temporary copies of content items is presented in [15]. The work in [16] propose a revised forwarding strategy that can better prevent or detect loops in NDN.

Related work have also explored potential strategies in Wireless networks, such as a strategy for vehicular ad hoc networks in [6], and a set of adaptive forwarding strategies that can use multiple access networks simultaneously in [17].

The work in [18] presents a probability-based adaptive forwarding strategy, including a statistical model to compute strategy retransmission intervals.

While related work explores different mechanisms and approaches for forwarding strategies in ICN, our work is mainly focused on exploring the dynamics between applications and forwarding strategies.

3. FORWARDING MECHANISMS

In this section, we decompose the operations of a forwarding strategy in NDN into two core mechanisms: face selection and failure response.

3.1 Next Hop Selection

When the FIB entry consists of multiple faces, the forwarding strategy must decide on what face an Interest should be sent. When selecting the Interest next hop, the strategy may choose to send the Interest on a single face, a subset of faces, or all available faces. It may do that when there is no single best-face to use, when the strategy asks to probe additional faces, or when the strategy is designed to multicast interests. The selection of the next hop can rely on external input, such as the face *cost* determined by the routing protocol, or on an internal feedback previously collected by the strategy itself, usually referred as the face *rank*.

An important attribute of a forwarding strategy is its adaptation to changes. In NDN, a data packet is forwarded on the reverse path of the Interest packet. Therefore a strategy can record the performance of each face to learn if it works and how well it performs. Then, it can use this knowledge to update the face's rank and improve future next-hop decisions. However, a strategy is not required to do so, and can elect to exclusively rely on the input provided by external services, such as a routing protocol. Best-route is a good example of such a strategy. When relying on internal strategy feedback, every strategy is free to choose the metric of the collected feedback according to its goals. For instance, the face rank can be determined according to the upstream round-trip-time (RTT), the number of hops to the producer, or the face successful delivery rate.

3.2 Failure Response

After selecting the Interest next hop(s), the forwarding strategy must decide how to react when the Interest is not satisfied within a specific amount of time. For each forwarded Interest, the NDN router initiates a timer for the period in which it expects to receive back a Data packet. When a Data packet is received within this period, the packet is forwarding back to the consumer by following the information kept in its PIT. At this time, the forwarding strategy can update the face rank for future use. However, there are three possible outcomes when the timer expires prior to the reception of a Data packet: 1) The strategy **drops** the Interest packet. 2) The strategy **retransmits** the packet on the same or a different face(s). 3) The strategy replies with a NACK packet to the previous hop [19]. As we illustrate in the next section, this mechanism critically affects on the application correctness. Thus, the application developer must be aware of the approach taken by the forwarding strategy to best decide how to design the application namespace and how to handle application's retransmissions.

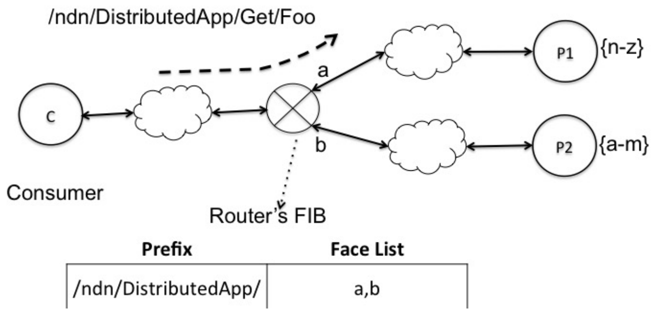
4. FORWARDING AND APPLICATIONS

To understand the importance and impact of the failure response mechanism, we illustrate three use cases in which the application's correctness is affected when the application's expectation for in-network retransmission is not met by the strategy.

4.1 Use Case 1 - Multiple Producers with Disjoint Data

In this use case of a multiple producers application, we discuss a scenario in which each producer holds a disjoint part of the content provided by the service. Figure 2 illustrates the application *DistributedApp* as an example of such an application. Here, the content held by the service is distributed between two producers, and each can satisfy only a subset of the requests, depending on the content first letter. In this example, the router's FIB consists of exactly one entry with two upstream faces that can potentially satisfy the application namespace. For every incoming Interest, the strategy has three possibilities for next hop selection: face a, face b, or multicast on both. Since flooding the network for every incoming Interest paired with this strategy is not a scalable solution, the strategy can try to make an educated decision based on previous collected data feedback. If the strategy chooses the wrong face for a request, the Interest remains unsatisfied. Here, if the strategy retransmits the Interest on another upstream face right after the previous one timed out, it can explore other upstream faces and guarantee that all requests for every content held by a producer are satisfied.

If the strategy does not retry the Interest on a different face, then the consumer can resend it again after the Interest lifetime expires. However, this approach introduces a big challenge to the application developer. As described in the Background section, some strategies implement a suppression mechanism to block flooding attacks, and therefore drop retransmitted Interests received within a specific amount of time. Therefore the application developer must consider the details of the suppression mechanism used by



Prefix	Face List
/ndn/DistributedApp/	a,b

Figure 2: Multiple producers with disjoint content

the strategy. For instance, if the suppression mechanism dictates that an Interest must be retransmitted 20ms after the first attempt, then the application has to implement its retransmission code blocks and timers according to the 20 ms requirement. This approach has two major drawbacks: First, there is no guarantee that the same strategy is used in all routers, and therefore it is complicated to develop a dynamic application logic for more than one possible suppression algorithm. Second, while the details of the strategy may be well known in the application development phase, there is no guarantee that the strategy will maintain the same algorithm in subsequent software versions. Therefore, every modification to the strategy suppression timer would require a modification in the application code as well. This creates a strong and dangerous coupling between the application and the network protocols.

When considering these drawbacks of consumer retransmissions, we make the claim that forwarding strategy retransmissions is a potential straightforward solution to the disjoint producers problem. However, if the application developer decides to rely on a strategy that performs in-network retransmissions, and the network does not meet the application's expectations, the application performance and even correctness can be affected.

It is possible that a better namespace design, such as one that differentiates producer namespaces to reflect the content held by each producer in two separate FIB entries, would be a better approach then relying on strategy retransmissions. While we agree that the namespace design is crucial, we argue that some applications would not be able to follow such a namespace restriction, or that their developer might not understand the importance of this requirement when designing the application's namespace. Moreover, we believe that the NDN platform should be able to support different types of applications, and should not enforce strict rules on the namespace design simply because of strategy characteristics.

4.1.1 Use Case 1 illustration

To illustrate the correctness problem in Use Case 1, we used four hosts in the Open Network Lab (ONL) [20] to experiment with the topology described in Figure 4.1; Each host ran NFD version 0.4 and the latest version of ndn-traffic [21]. We used two instances of ndn-traffic-server to illustrate the two disjoint producers, and one instance of ndn-traffic to illustrate the consumer. We set the consumer to request 100 content items, with approximately 50% of the requests for content held by producer 1, and the rest for content held

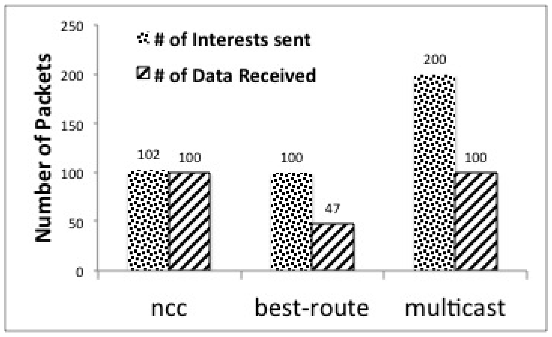


Figure 3: Number of Packets Sent over Different Strategies

by producer 2. We repeated the experiment three times for three strategies implemented in NFD 0.4 (ncc, best-route, and multicast), and recorded the total number of Interests sent and the total number of Data packets received at the router.

As expected, the *multicast* strategy sent each Interest on both faces, and therefore the number of the Interests sent was twice the number of requests. The results also show that *best-route* retrieved only 47% of the content items due to its selection of exactly one face, with no strategy retransmissions. However, *ncc* successfully retrieved all the content items because its failure response mechanism retransmits the Interest on the second face when the first one times out. Here, the results demonstrate how the application's correctness is affected by a forwarding strategy, that cannot guarantee 100% Interest satisfaction.

4.2 Use Case 2 - Multiple Producers with the Same Replica

This use case also consists of multiple producers, but simplifies the previous scenario by having a complete replication of the content in all existing consumers. Here, we argue that the application correctness can be affected even when all producers hold replications of the same content. An example of this scenario is an NDN application that asks to save previously generated Data packets in a persistent storage. For such an application, producing content can be a time consuming operation, and therefore while it is producing one requested content, it can not respond to other incoming requests. Therefore, the application design seeks to store every Data packet it produced in a repository to remain idle for other future requests, and to avoid producing the same content again. Since the content produced is large, the application could not rely on the NDN content store to satisfy all previously produced Data packets. In this case, in-network strategy retransmissions can immediately fetch the content from the repository face when the application is busy and does not respond with data on time. However, as illustrated in Use Case 1, if the application expects in-network retransmission that the strategy does not provide, the interest satisfaction rate of the producer can be reduced, degrading the application correctness. We present a larger scale evaluation for this use case in Section 6.

4.3 Use Case 3 - Notification Services

Notification service applications are distributed applications in which the application's parties do not have clear consumer-producer relations, and the distributed parties make requests to inform each other about their status. A good example of such an application is a synchronization service, in which each of the participating hosts requests to inform the others about an update made to its local repository [22, 23, 24, 25]. We specifically discuss the design of the CCNx Sync synchronization protocol [22] to demonstrate how it is coupled with the failure response mechanism of the ncc strategy.

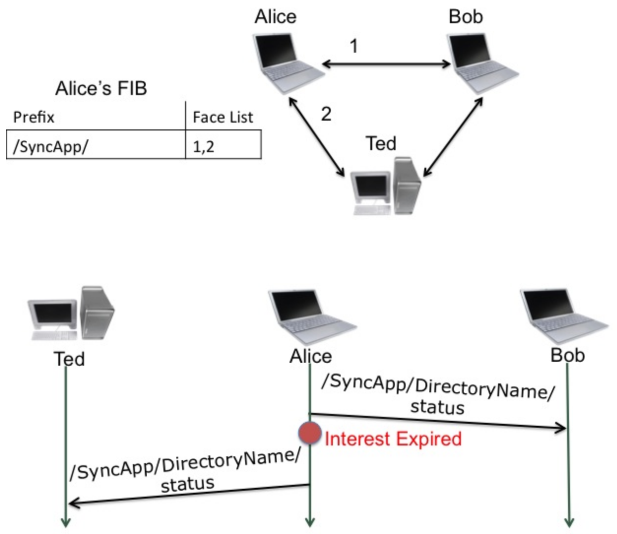


Figure 4: CCNx Sync Use of Strategy Retransmission

In our example, CCNx Sync is used by a dropbox style application to keep the content of a directory up-to-date among participants. Whenever a change is made to the synced directory, CCNx Sync sends a notification in the form of an Interest packet to inform the other participants about the new status of the directory. This specific protocol is coupled with the default CCNx forwarding strategy, the original implementation of ncc, and therefore uses a single Interest packet to notify all the participants of the change. As described in the Background section, the ncc strategy selects one face when forwarding an Interest, but retransmits the Interest on additional faces if the Interest timer expires before the Data packet is received. In the CCNx Sync protocol, the participants do not usually reply to the notification Interest with a Data packet, and therefore the Interest expires and is retransmitted to the next available face by the strategy.

Figure 4 shows the FIB entries and the message flow of our example. The figure presents the timeline of Alice, Bob, and Ted, following a request by Alice to notify them of a change to her local directory. Here, Alice, using CCNx Sync, sends one Interest packet to communicate the change to Bob and Ted. The strategy selects Bob's face as the face to send the Interest on, and when the Interest timer expires with no response from Bob, the strategy retransmits the Interest on Ted's face. In this example, the sync service sends exactly one Interest, and the ncc strategy retransmits the packet. This strategy mechanisms and the lack of Data response to the Interest guarantees that the notification is eventually transmitted on all faces. This retransmission expectation creates a strong coupling between CCNx Sync and the failure response mechanism of the ncc strategy. Here, the protocol is designed not to respond to notification Interests, and therefore it is expected that all faces will eventually time out. If the sync namespace is paired with another strategy that does not retransmit in respond to timeouts, then the notification Interest will not reach all participants. Alternatively, if the protocol keeps using ncc but is modified to respond with a Data packet to every Interest notification, then Bobs' face will not time out and the failure mechanism is never triggered. Therefore, a change to the strategy retransmission mechanism without a corresponding change to the application impacts the application correctness. As mentioned before, the application has no control over the strategies used in remote routers, and therefore the correctness of the CCNx sync protocol under this design is fragile.

4.3.1 *Use Case 3 illustration*

To demonstrate our use case 3 arguments, we measured the performance of a notification service, CCNx Sync, when using two forwarding strategies, the default CCNx that retransmits Interests in case of timeouts, and the loadsharing strategy that does not. We used 16 hosts in the ONL [20]; each host ran CCNx version 0.82.

We inserted a timestamp as a content item into the repository of one of the 16 hosts, and measured the time it took CCNx Sync to synchronize the content among all the participants. We repeated the same experiment using different forwarding strategies and over two different topologies. In the mesh topology, each of the parties was connected to two other parties, and in the fully connected mesh topology each party was connected to all the other. We selected weak and well connected networks to explore whether the connectivity of the network reduces the impact of the forwarding strategy. We performed the described experiment. Figure 5 shows the average recorded CCNx Sync synchronization time over a mesh topology and a fully connected mesh.

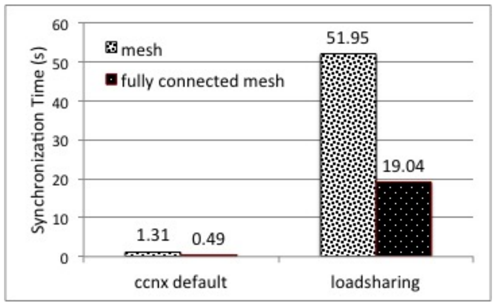


Figure 5: Average Synchronization Time

Clearly, the synchronization time when using loadsharing strategy is approximately 97% slower than when using the default CCNx Strategy in both topologies. The selected forwarding strategy greatly impacts the application's performance, and this impact is not mitigated by the network connectivity.

5. RETRANSMISSION ABSTRACTION

As shown in the previous section, some applications require or could benefit from strategy retransmission as a response to face timeout. However, it might be that other applications require complete control of their traffic, and request to avoid network retransmissions. An example of such an application is the video streaming application described in [26]. This application builds its own traffic control mechanism to support a continuous and interruption-free communication by relying on its messages' round-trip-times (RTT) to determine the streaming rate at any point in time. In this scenario, network retransmission negatively impacts the application's measurements, and therefore degrades the correctness and performance of the application.

To support all different types of applications, we suggest that the decision to perform in-network retransmission should be made by the application, and executed only by the forwarding strategy. This way, the decision whether to retransmit in the network is decoupled from a variable strategy implementation, and made only by the application. Our proposed retransmission mechanism performs two independent yet complementary functions: retransmissions decoupling and retransmission differentiation.

5.1 Retransmission Decoupling

As described in previous sections, the application cannot always rely on the variable failure response of the strategy, specifically when it expects one retransmission behavior and the strategy chooses another. Thus, we suggest adding a new TLV to the Interest packet to specify the application retransmission requirement. We name this Boolean type field the *'Interest Retransmission Policy'* (IRP) flag.

By setting the IRP to True or False, the application dictates whether the strategy should or should not retransmit an Interest as part of its failure response mechanism. A forwarding strategy can then support each option by providing the two failure response mechanisms as part of its implementation, one that performs in-network retransmissions and another that does not.

Algorithm 1 presents a simplified framework for a forwarding strategy that supports both retransmission mechanisms using the IRP flag.

```
Function ForwardInterst($interest$):
    $face\_list$ = SelectNextHop($interest$)
    IRP = GetIRP($interest$)
    SendInterest($interest$, $face\_list$)
    if $IRP$ then
        schedule retransmission at time x
    else
        wait for application retransmission
    end
    return
```

Algorithm 1: Strategy framework that decouples in-network retransmissions

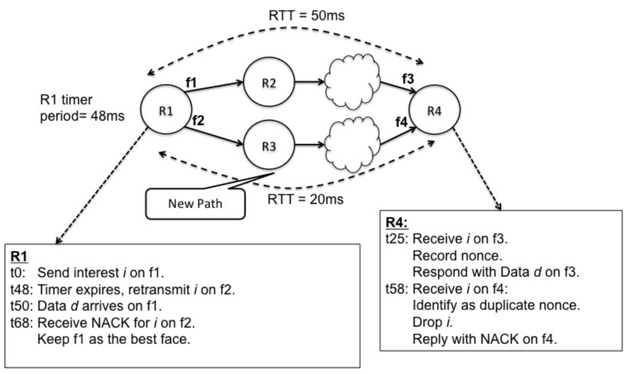
It is important to note that the IRP flag does not determine the in-network retransmission algorithm, but only requires that one exists. Thus, the application decides whether an Interest should be retransmitted by the network, while the strategy determines the in-network retransmission algorithm, that is, when to retransmit and which next hop(s) to choose. The retransmission and suppression timers presented in algorithm 1 are only placeholders for possible retransmission algorithms provided by a forwarding strategy. The strategy is free to choose any algorithm to support the two options. For example, a core network strategy might choose a retransmission algorithm that addresses congestion issues and relies on collecting round-trip-times, while an access strategy retransmission algorithm might simply follow a list of given faces and retransmit an Interest after a fixed time interval. The work in [18] proposes a statistical model to compute retransmission intervals.

5.2 Retransmission Differentiation

We suggest adding a second Interest TLV, the *'Network Retransmission Differentiation' (NRD)*, to differentiate application Interests from network retransmissions. Using the NRD TLV, strategies can support different mechanisms for control and data traffic, and can collect performance measurements of alternative next hops in dynamic environments. We describe two scenarios in which the NRD field is required.

First, in dynamic networks, such as in a vehicular network [6] or in wireless networks[17], an adaptive forwarding strategy is used to probe faces to explore alternative next-hops. Such a strategy might want to differentiate the probed Interests from the Interests generated by the application to support designated strategy mechanisms for control and data traffic.

The second scenario is an existing problem in the current ncc strategy and the NFD forwarding mechanism, in which loop detection caused by nonces can prevent better face exploration. As explained in the Background section, ncc adjusts its retransmission timer up whenever the best face upstream times out, and adjusts it back down whenever the face upstream successfully retrieves data. Adjusting the timer down for every successful data retrieval guarantees that at some point, the timer period is less than the upstream RTT, and therefore allows ncc to explore other potential upstreams. However, because of the duplicated nonce mechanisms, ncc can fail to explore potentially better performing upstreams.



R1
t0: Send interest *i* on f1.
t48: Timer expires, retransmit *i* on f2.
t50: Data *d* arrives on f1.
t68: Receive NACK for *i* on f2.
Keep f1 as the best face.

R4:
t25: Receive *i* on f3.
Record nonce.
Respond with Data *d* on f3.
t58: Receive *i* on f4:
Identify as duplicate nonce.
Drop *i*.
Reply with NACK on f4.

Figure 6: NACK problem

As presented in Figure 6, R1 has a new faster path to R4 through R3, but the ncc strategy has previously selected R1 as the (only) best performing upstream face, thus it sends all Interests on face 1. When the R1 timer approaches 48ms, which is smaller than the actual 50 ms RTT through R2, the timer of Interest *i*, sent at t0, causes R1 to retransmit the Interest on face 2 at t48. Router R4, which received Interest *i* from R2 at t25, receives it again at t58. R4 recognizes the second *i* and its nonce as a duplicate Interest, and therefore drops it and replies with NACK. Here, ncc on node R1 does not receive a Data packet on face 2, and therefore continues to use face 1 as the best-performing face, although its upstream RTT is more than twice that of the other new path. The ncc strategy will switch to use face 2 only if a retransmitted Interest arrives at R4 before the original Interest. In other words, the strategy changes its best-face selection only if the "timer period" plus the "one way trip time through R3" is less than the "one way trip time through R2".

This problem could be solved by adding the NRD TLV to the retransmitted Interest, and differentiating the retransmitted Interest from the original one. This way, the strategy does not detect the Interest as a duplicate one, thus enabling better face exploration. However, by adding NRD TLV and processing Interests with the same nonce, we interrupt the core mechanism of loop detection in NDN. Therefore, using the NRD as a simple Boolean flag does not solve the problem. In our implementation, we used a non-negative-integer to represent the NRD TLV. We set the initial value of the NRD TLV to 0, and increased it by one every time the Interest was retransmitted by the strategy to an additional face. In our experiments, we selected 10 as the maximum number of allowed retransmissions, and replied with NACK if the Interest's nonce was previously recorded and the NRD TLV was equal to 10. In addition, we implemented the strategy mechanism to reply with NACK when there were no unused upstream faces to use. Although our implementation provided us with the desired behavior, the NRD mechanism should be better explored as part of future work. We present a framework of our implementation in algorithm 2.

```
Function DetectLoopAndRetransmissions(interest):
    if nonce previously recorded then
        if NRD == 'MaxAllowed' then
            send NACK
        else
            interest.NRD++
            HandleRetransmission(interest)
        end
    else
        interest.NRD++
        ForwardInterst(interest) [algorithm 1]
    end
    return
```

Algorithm 2: Retransmission Differentiation using NRD

Unlike NDN, CCN does not use nonces to detect loops, but uses an additional Time-To-Live(TTL) TLV to avoid infinite loops. While the problem described in Figure 6 might not occur in CCN, we suggest that the proposed differentiation can be useful in the CCN architecture to support more intelligent forwarding strategies that can differentiate an application Interest from an Interest injected by the network.

6. EMPIRICAL RESULTS

We implemented the proposed retransmission mechanism in NFD 0.4 and added the two suggested TLVs to the Interest packet. We modified the loop-detection mechanism to follow algorithm 2, and tested the proposed in-network retransmission abstraction by running a set of experiments using the emulated NDN testbed in the Open Network Lab (ONL) [27, 11]. The emulated environment consisted of 26

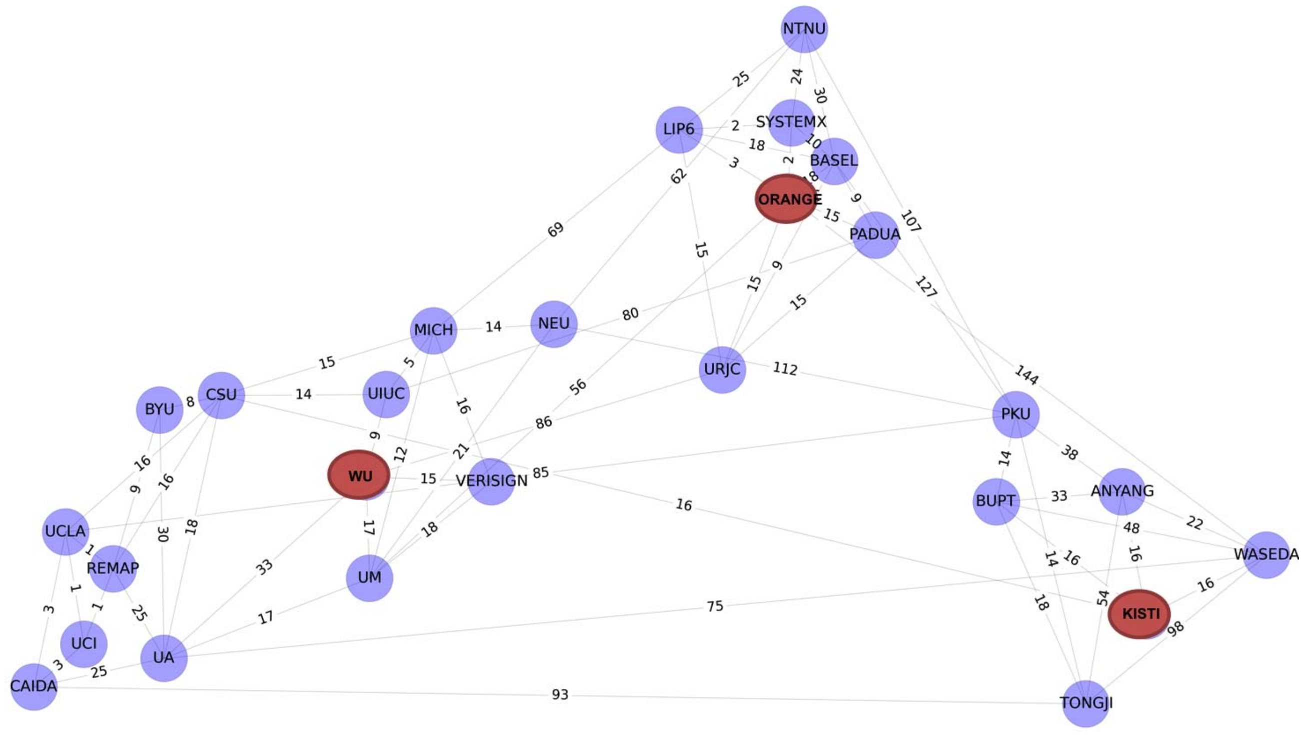


Figure 7: Emulated NDN testbed

dual-core machines that represent the testbed gateways, 26 virtual machines(VM) that represent end hosts, and four software routes. All these machines run Ubuntu 12.04.5 and our modified version of NFD 0.4. We configured each gateway to publish the same set of namespaces used by the corresponding world-wide NDN testbed [11] gateway, and ran NLSR 0.2.2 as the network routing protocol to distribute the gateways' namespaces. The emulated testbed also had 66 links that were configured with costs that match world-wide NDN testbed costs. We connected one VM to each of the gateway machines to emulate one end host connected to each gateway. Figure 7 presents our emulated gateways' topology. To simplified the presented topology, Figure 7 does not present the end-hosts connected to each gateway.

We modified the best-route and ncc strategies so they checked the IRP flag in order to determine if in-network retransmission was required by the application, and used NRD TLV to differentiate in-network retransmissions from application Interests. We used algorithm 2 to prevent infinite loops of retransmitted Interests. We named the modified best-route and ncc strategies the 'best-route-r' and 'ncc-r' strategies.

Three experiments in the scale of the NDN testbed demonstrated the impact of in-network retransmissions on the application correctness, and provided the cost of retransmissions in each of the scenarios.

6.1 Multiple Producers Application with One Congested Producer

In this experiment, we used modified versions of ndn-traffic and ndn-traffic-server to generate Interests and Data packets. We ran ndn-traffic consumer on the VM connected to the WU gateway, and two instances of ndn-traffic-server as multiple producers on the VMs connected to the ORAMGE and KISTI gateways. These three gateways are colored in red in Figure 7. We configured both servers to respond to the Interests sent by the consumer, hence both producers held the same replication of content. To emulate a use case in which one producer is congested and therefore fails to respond with Data packet, we stopped one producer for 10 seconds during the run of the experiment. According to the testbed link costs, it was cheaper to get from WU to ORANGE than it was to KISTI, and therefore we selected ORANGE VM as the congested producer. We set the consumer's Interests IRP flag to True, and thereby required in-network retransmission from the strategy. We did not provide any retransmission mechanism for unsatisfied Interests in the application scope. The total traffic sent over the network consisted of the traffic generated by our producer as well as the traffic generated by NLSR.

The details of the experiments can be summarized as follows: At the beginning of the experiment, we configured the consumer to start expressing Interest packets at the rate of 50 Interests per second, and the producers to respond with Data packets for each received Interest. We stopped the producer on ORANGE VM 10 seconds after the start point, and brought it back up again 10 seconds later for an additional five seconds.

We repeated the experiment five times with each of the following strategies: best-route, best-route-r, ncc, and ncc-r. We collected the total number of Interests sent by the consumer, the total number of Data packets received from

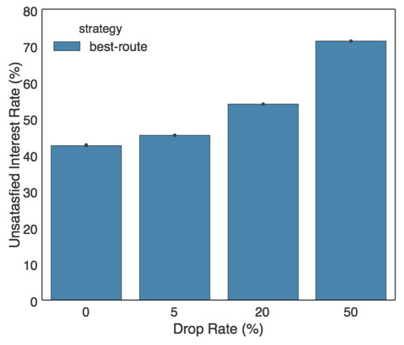
Strategy	Unsatisfied Interest Rate(%)	Total Interest Sent by WU Gateway	Std Sample
best-route	42.55	1700	0.09
best-route-r	0.621	3563	0.00048
ncc	0.95	5322	0.044
ncc-r	0.93	5490	0.00073

Table 1: Multiple Producers Results Summary

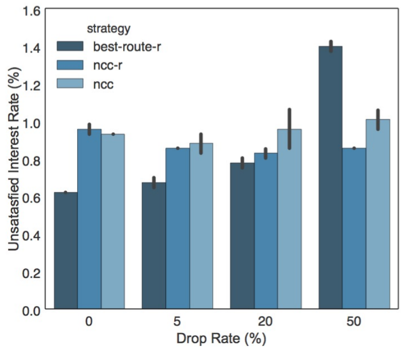
each producer, and the number of Interests sent by WU gateway. The average results are presented in Table 1.

As shown in Table 1, when using best-route as the strategy paired with the application's namespace, an average of 42% of the expressed Interests remain unsatisfied. However, less than 1% of sent Interests remain unsatisfied when the application's namespace is configured with best-route-r, ncc, or ncc-r. In addition, Table 1 shows that the number of Interests sent by the WU gateway when using best-route-r was twice the number of Interests send by WU when using best-route. This difference is explained by the specific implementation of best-route-r, in which the strategy retransmits an Interest after a fixed amount of time, which is shorter than the actual round-trip time in the used topology. This detail in the in-network retransmission mechanism should be better explored as part of future work. However, the experiment demonstrates that a simple change to the best-route strategy, supporting the IRP flag, can dramatically improve the unsatisfied Interest rate in the case of multiple producers with a congested node, and therefore supports a wider range of applications.

Our statistical analysis of the results did not indicate any statistical difference between ncc and ncc-r, Therefore, we can conclude that supporting the IRP flag does not change the performance and correctness of strategies that already support in-network retransmissions as part of their default implementation.



(a)



(b)

Figure 8: Unsatisfied Interest Rates for Different Link Loss Rate: (a) best-route. (b) best-route-r, ncc-r, and ncc. Note the very different Y axis ranges

6.2 Multiple Producers with a Congested Producer and a Congested Gateway

To emulate a congested gateway, we repeated the previous set of experiments and configured the WU gateway with a different drop rate each time. We used drop rates of 5%, 20%, and 50%. We present two figures due to the different scale of the results: Figure 8a presents the unsatisfied Interest rates of the best-route strategy and Figure 8b presents the unsatisfied Interest rates of the other strategies explored.

For the best-route strategy the rate of unsatisfied Interests reaches an average of 70% when the congested gateway drops 50% of the packets. However the unsatisfied Interests rate remains less than 1.5% when using best-route-r, ncc and ncc-r. This contrast again shows how a simple support of in-network retransmission in the best-route strategy can improve the performance of a multiple producer application, even when one of the gateway nodes is congested and drops 50% of the packets.

6.3 Multipath with a Congested Link

In this experiment, we used a simple consumer-producer service using ndn-traffic as the consumer running on the WU VM, and ndn-traffic-sever as the producer running on the KISTI VM. As before, we modified the consumer to set IRP to True, and did not support any application retransmission mechanism in the application's scope. To emulate a congested link, we set a drop rate of 100% on the link between WU and UIUC, which is the least expensive next-hop to reach the producer from the WU gateway. We collected RX and TX counters every 0.1 seconds on all participating links.

The details of the experiments can be summarized as follows: As before, we started the experiment by configuring the consumer to send 50 Interest packets per second. Ten seconds later, we configured the link between WU and UIUC to drop all application packets sent by the WU gateway. We recorded the traffic for 120 seconds before removing the dropping filter, and continued to record measurements for an additional 120 seconds before stopping the consumer's traffic. The total runtime of the experiement was 250 seconds.

Figure 9 shows the traffic recorded on the producer and consumer VMs when using the best-route strategy, and Figure 10 shows the traffic recorded using the best-route-r strategy. From these two figures we learn that all Interests sent during the dropping interval remained unsatisfied when using best-route, while the consumer-producer traffic remained unaffected when using best-route-r.

To better explore the strategy behavior, we recorded all the traffic transmitted on the WU gateway links to the following immediate hops: UIUC, UM, URJC, and VERISIGN. We show the results using the best-route strategy in Figure 11, and the results using the best-route-r strategy in Figure 12.

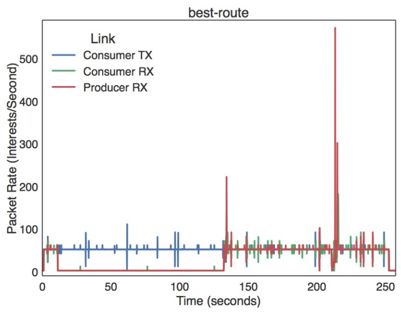


Figure 9: End Hosts Traffic over Time with best-route

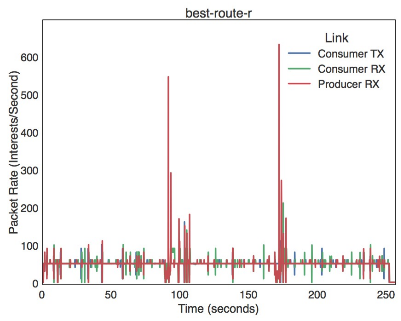


Figure 10: End Hosts Traffic over Time with best-route-r

As shown in Figure 11, due to the NLSR costs configured on the emulated testbed, UIUC was selected as the best next-hop by the best-route strategy. At t=10, when the link towards UIUC started dropping all the Interest packets transmitted by the consumer, the traffic on this link dropped to almost zero. Due to our overlay network setup on top of the ONL machines, the traffic reported in these figures contains NLSR traffic, and therefore the recorded TX counters on this link do not present an absolute zero. At t=135, NLSR determined the link to UM is the new least expensive nest-hop to the producer, and therefore the best-route strategy rerouted all the traffic to use this link. At t=210, 100 seconds after we stopped dropping UIUC packets, NLSR determined UIUC as the least expensive next hop again, and rerouted the traffic towards that link.

Figure 12 presents the measurements of the same WU links when using the best-route-r strategy. As before, the link to UIUC was first selected as the best next-hop towards the producer. At t=10, when the link towards UIUC started dropping the consumer packets, the best-route-r strategy retransmitted all unsatisfied Interests towards UM, without waiting for NLSR to declare this face as the least expensive next-hop. The strategy continues to use UM link until UIUC becomes available again. This quick response was achieved because the application set the IRP flag to require in-network strategy retransmissions.

It is important to clarify that our intention in this experiment was not to compare forwarding recovery times to routing convergence times [13]. Instead we sought to demonstrate how a multipath consumer-producer service can maintain a continuous traffic flow even when the network is congested, and without forcing the application to implement a

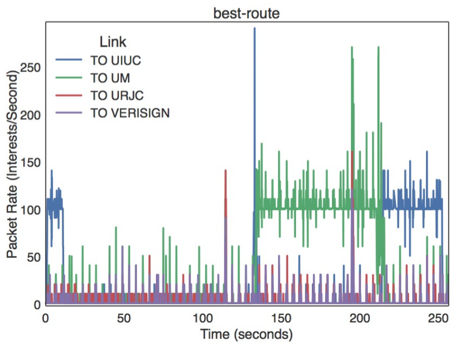


Figure 11: WU Traffic over Time with best-route

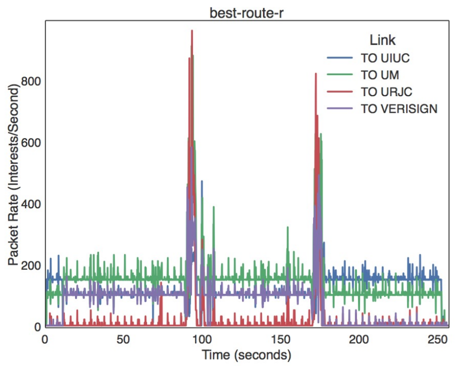


Figure 12: WU Traffic over Time with best-route-r

retransmission mechanism, as required by the existing best-route strategy. Moreover, as can be seen in Figure 12, the Interest rate sent by the best-route-r strategy on the WU gateway is on average twice the rate sent on WU using best-route. As in the previous experiments, this is a direct outcome of the fixed retransmission intervals we implemented in best-route-r, which is shorter than the actual round-trip time in the used topology, and will be better explored as part of future work.

7. CONCLUSION AND FUTURE WORK

In this paper, we discuss one of the core mechanisms of a forwarding strategy in NDN: the failure recovery mechanism when an Interest remains unsatisfied. We show that in some use cases, the application's correctness can be negatively affected when the strategy failure mechanism does not meet the application's retransmission expectations. We argue that relying on a dynamic strategy output creates a coupling between the application design and the strategy mechanism which can be easily interrupted when the selection of the forwarding strategy is overwritten by the network operator. Thereby, the application performance and correctness can be negatively affected in some use cases. We propose to decouple in-network strategy retransmissions from applications by adding the IRP flag to every Interest packet, and by letting the application developer decide whether an in-network strategy retransmission is required. In our proposed solution, the application provides the requirement, and the strategy decides how to fulfill it. By using the IRP flag, an application can maintain correct traffic flow regardless of the underlying strategy, and can eliminate its dependency on the strategy's in-network retransmission policy. In addition, we propose adding the NRD TLV to differentiate retransmitted Interests from others.

In this work, we focus on proposing two additional TLVs to the Interest packet, one for decoupling and another for differentiating strategy and application retransmissions. We made simple modifications to existing strategies to support and evaluate the proposal. However, the details of strategy retransmissions algorithm, such as the waiting time before retransmitting an unsatisfied Interest, should be further explored as part of future work. We would also like to extend our set of experiments to use real-world applications on top of NDN to determine whether there is one in-network retransmission algorithm that outperforms others over a wide range of applications. Another important future work is exploring the security aspects of the proposed mechanism. If the Interest packet is signed, changing the NRD field may cause validation challenges. However, we believe that a future solution to a changed nonce in NDN Interest, or a changed TTL in CCN Interest, could be applied here as well.

Acknowledgement

This work was supported by CNS-1040643 and CNS-1345282 NSF grants, and by a research gift from Cisco. The authors would also like to thank John DeHart and Jyoti Parwatikar for their assistance with the Open Network Laboratory and the NDN testbed.

8. REFERENCES

[1] Lixia Zhang et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 2014.

[2] Project CCNx. http://www.ccnx.org/.

[3] Alexander Afanasyev et al. Nfd developer's guide. Technical report, NDN-0021, NDN, 2014.

[4] NFD Named Data Networking Forwarding Daemon. http://named-data.net/doc/NFD/current/.

[5] Marc Mosko. Ccnx 1.0 protocol specification roadmap. 2013.

[6] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. Navigo: Interest forwarding by geolocations in vehicular named data networking. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, pages 1–10. IEEE, 2015.

[7] Cheng Yi et al. A case for stateful forwarding plane. *Computer Communications*, 2013.

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

[9] Ilya Moiseenko, Lijing Wang, and Lixia Zhang. Consumer/producer communication with application level framing in named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 99–108. ACM, 2015.

[10] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, Michele Papalini, and Sen Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2013.

[11] NDN Testbed. http://ndnmap.arl.wustl.edu/.

[12] AKM Hoque et al. Nlsr: Named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, pages 15–20. ACM, 2013.

[13] Cheng Yi et al. On the role of routing in named data networking. In *Proceedings of the 1st international conference on Information-centric networking*, pages 27–36. ACM, 2014.

[14] Cheng Yi, Alexander Afanasyev, Lan Wang, Beichuan Zhang, and Lixia Zhang. Adaptive forwarding in named data networking.

[15] Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio, Dario Rossi, and Giuseppe Rossini. Inform: a dynamic interest forwarding mechanism for information centric networking. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, pages 9–14. ACM, 2013.

[16] JJ Garcia-Luna-Aceves. A fault-tolerant forwarding strategy for interest-based information centric networks. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[17] Klaus M Schneider and Udo R Krieger. Beyond network selection: Exploiting access network heterogeneity with named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 137–146. ACM, 2015.

[18] Haiyang Qian, Ravishankar Ravindran, Guo-Qiang Wang, and Deep Medhi. Probability-based adaptive forwarding strategy in named data networking. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 1094–1101. IEEE, 2013.

[19] Alberto Compagno et al. To nack or not to nack? negative acknowledgments in information-centric networking. *arXiv preprint arXiv:1503.02123*, 2015.

[20] Charlie Wiseman et al. A remotely accessible network processor-based router for network experimentation. In *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 2008.

[21] ndn traffic. https://github.com/named-data/ndn-traffic-generator.

[22] CCNx Sync. https://www.ccnx.org/releases/latest/doc/technical/SynchronizationProtocol.html.

[23] Zhenkai Zhu and Alexander Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *ICNP*, pages 1–10, 2013.

[24] Minsheng Zhang, Vince Lehman, and Lan Wang. Partialsync: Efficient synchronization of a partial namespace in ndn. Technical report, Technical Report NDN-0039, NDN, 2016.

[25] Wenliang Fu, Hila Ben Abraham, and Patrick Crowley. Synchronizing namespaces with invertible

bloom filters. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems*, pages 123–134. IEEE Computer Society, 2015.

[26] Peter Gusev and Jeff Burke. Ndn-rtc: Real-time videoconferencing over named data networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 117–126. ACM, 2015.

[27] Ze'ev Lailari, Hila Ben Abraham, Ben Aronberg, Jackie Hudepohl, Haowei Yuan, John DeHart, Jyoti Parwatikar, and Patrick Crowley. Experiments with the emulated ndn testbed in onl. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 219–220. ACM, 2015.