Report Number: WUCSE-2016-002

2016-7

In-Network Retransmissions in Named Data Networking

Hila Ben Abraham and Patrick Crowley

The strategy layer is an important architectural component in both Content-Centric Networking (CCN) and Named Data Networking (NDN). This component introduces a new forwarding model that allows an application to configure its namespace with a forwarding strategy. A core mechanism in every forwarding strategy is the decision of whether to retransmit an unsatisfied Interest or to wait for an application retransmission. While some applications request control of all retransmissions, others rely on the assumption that the strategy will retransmit an Interest when it is not satisfied. Although an application can select the forwarding strategy used in the local host,... **Read complete abstract on page 2.**

In-Network Retransmissions in Named Data Networking

Hila Ben Abraham and Patrick Crowley

Complete Abstract:

The strategy layer is an important architectural component in both Content-Centric Networking (CCN) and Named Data Networking (NDN). This component introduces a new forwarding model that allows an application to configure its namespace with a forwarding strategy. A core mechanism in every forwarding strategy is the decision of whether to retransmit an unsatisfied Interest or to wait for an application retransmission. While some applications request control of all retransmissions, others rely on the assumption that the strategy will retransmit an Interest when it is not satisfied. Although an application can select the forwarding strategy used in the local host, it cannot guarantee the selection of the same strategy in other nodes in the network, especially in shared resource environments. In some scenarios, a developer must bind the implementation of the application to the details of the deployed forwarding strategy to guarantee the correctness of his application. In this paper we discuss the core mechanisms of a forwarding strategy in NDN, and we explore the importance and impact of in-network retransmissions on the application's performance and correctness. We propose and implement a simple forwarding strategy abstraction that allows the application to decide whether a network retransmission is required, and differentiate application retransmissions from network retransmissions. We show that in some scenarios, such as multiple producers application or multipath consumer-producer service, the proposed abstraction can significantly reduce the percentage of unsatisfied Interests.

In-Network Retransmissions in Named Data Networking

Hila Ben Abraham
Computer Science and Engineering
Washington University in st. louis
hila@wustl.edu

Patrick Crowley
Computer Science and Engineering
Washington University in st. louis
pcrowley@wustl.edu

ABSTRACT

The strategy layer is an important architectural component in both Content-Centric Networking (CCN) and Named Data Networking (NDN). This component introduces a new forwarding model that allows an application to configure its namespace with a forwarding strategy. A core mechanism in every forwarding strategy is the decision of whether to retransmit an unsatisfied Interest or to wait for an application retransmission. While some applications request control of all retransmissions, others rely on the assumption that the strategy will retransmit an Interest when it is not satisfied. Although an application can select the forwarding strategy used in the local host, it cannot guarantee the selection of the same strategy in other nodes in the network, especially in shared resource environments. In some scenarios, a developer must bind the implementation of the application to the details of the deployed forwarding strategy to guarantee the correctness of his application.

In this paper we discuss the core mechanisms of a forwarding strategy in NDN, and we explore the importance and impact of in-network retransmissions on the application's performance and correctness. We propose and implement a simple forwarding strategy abstraction that allows the application to decide whether a network retransmission is required, and differentiate application retransmissions from network retransmissions. We show that in some scenarios, such as multiple producers application or multipath consumer-producer service, the proposed abstraction can significantly reduce the percentage of unsatisfied Interests.

1. INTRODUCTION

In both Content-Centric Networking (CCN) and Named Data Networking (NDN), the forwarding strategy, sometimes referred to as the "strategy layer", is the architectural component that decides how to forward Interests when multiple next hops exist. 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 different NDN modules, including different forwarding strategies. Recent research has also focused on implementing NDN and related ICN architecture prototypes. Two on-going projects are the NDN forwarder (NFD) [1, 2], developed by the NDN research group, and the CCNx project [3, 4], developed by PARC. Both projects implement network forwarders and provide an API for NDN or CCN applications.

The current design of these NDN and CCNx software prototypes permits the application developer to pair a forwarding strategy with its application namespace, and therefore affects the way the application packets are forwarded in the network. Thus, the application developer can either choose an existing forwarding strategy or develop a new one to satisfy application-specific needs. While this might present new opportunities and advantages for ICN in isolated environments, it poses new challenges in shared resource environments.

When an application configures its namespaces to use a specific forwarding strategy, it couples its implementation with the strategy mechanisms. Therefore a change in the strategy mechanism could impact the application. While a change can work well in isolated environments, where the change is application specific, it can create conflicts when multiple applications use the same strategy. In addition, in shared resources environments such as core networks it is unlikely that the application developer would have the freedom to select the strategies used. In these environments, the network operator usually decides what strategies will be used, and therefore can revisit and change the application and strategy pairing. In other words, the application developer can pair a strategy with its application namespace on its local host, but has no control over the configured strategies in the core network.

One core mechanism of a forwarding strategy chooses what to do when an Interest is not satisfied. The router maintains an entry for a forwarded Interest in the Pending Interest Table (PIT) as long as the Interest's lifetime has not expired. In such cases, the forwarding strategy can decide to retransmit when the Interest packet was not satisfied by a Data packet or when the strategy asks to explore and probe additional faces [5, 6].

While some strategies retransmit unsatisfied Interests, other strategies leave the application with the de-

cisions of whether and when to retransmit. In addition, some applications request control of all retransmissions, but others assume that the strategy will retransmit an Interest that it is not satisfied. Therefore, if the network strategy chooses one retransmission policy when the application expects a different one, the performance and correctness of the application can be adversely affected [7].

While the retransmission policy is only one characteristic of a much more complex forwarding strategy, it is crucial. Therefore, we propose a strategy abstraction in which each strategy is implemented to support both applications that request to control all retransmissions, and applications that request the network to do it for them. In the proposed abstraction, the application determines whether a network retransmission is required, and the strategy determines only how it will be achieved. The contributions of this paper are as follows:

- We identify and discuss the main mechanisms of a forwarding strategy in NDN.
- We evaluate the importance and impact of in-network retransmissions in different use cases.
- We propose and evaluate a simple forwarding strategy mechanism that allows applications to decide whether a network retransmission is required, and that can be adapted by different forwarding strategies in shared resource environments.
- We propose a mechanism that helps forwarding strategies to differentiate a network retransmission from an application retransmission, and supports face probing that avoids loop-detection NACKs in NDN.

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

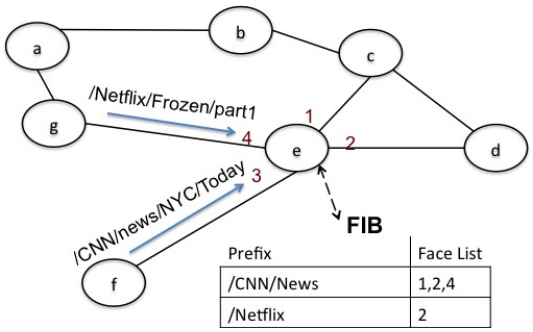

Figure 1: NDN Forwarding Information Base.

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 avoid 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 implementation of the NDN forwarder, the router responds with NACK when a duplicated Interest is detected.

The Interest and Data packets in NDN use type-length-value (TLV) encoding to represent each exchanged value. This provides an easy and dynamic platform for adding new information to either the Interest or the Data packet.

2.2 NDN Forwarding Plane

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 /Netflix/Frozen/part1, 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 NDN

In this section we briefly describe the forwarding strategies in related work, and in the implementation of NFD 0.4.

The first implemented forwarding strategy in ICN was the *default CCNx* strategy, which is also known as the *ncc* strategy as implemented 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 *prediction* time set by the strategy, then the face is remembered as the "best" face, and it is used to forward future interests of the same name. If the strategy timer expires before the arrival of a Data packet, the strategy retransmits the interest again to another available face. The CCNx default strategy is unique in the way it adjusts the prediction timer. Every time a Data packet is returned on the selected best face, the wait time is adjusted down, so the timer will expire faster the next time. When the Interest is not satisfied within the predicted 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. In *best-route*, every Interest packet is forwarded to the cheapest face, which is determined according to the cost assigned by the routing protocol. Named-data link state routing protocol (NLSR) [8] is currently the routing protocol configured to work with the best-route strategy. 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 the 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 [9].

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.

The *GreenYellowRed* strategy [5] is an adaptive forwarding that ranks the faces according to different forwarding policies, and chooses among the ranked groups in a round-robin fashion. As of the today, this strategy is not part of the official NFD distribution.

3. FORWARDING MECHANISMS

In this section, we identify the core mechanisms of a forwarding strategy. First, when the FIB entry consists of multiple faces, the forwarding strategy must decide on what face or faces an Interest should be sent. The strategy can choose the least expensive face according to the face *cost*, determined by the routing protocol, or the best performing face according to the face *rank*, determined by previous measurements of the forwarding strategy. In addition, 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 face, or when the strategy is designed to multicast interests.

Second, the forwarding strategy must decide how to react when the Interest is not satisfied within a specific amount of time. The NDN router initiates a timer for every forwarded Interest. When a Data packet is received before the timer expires, the forwarding plane sends the Data back to the consumer by following the information kept in its PIT. 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 special **NACK packet** to the previous hop [10].

We consider these two mechanisms, face selection and retransmission policy, as the core mechanisms of any forwarding strategy in NDN.

An important attribute of a forwarding strategy is its adaptation to changes. In NDN, a data packet is forwarded on the same but reverse path of the Interest packet. Therefore an **Adaptive Forwarding** strategy records the performance of each face to learn if it works and how well it performs. Then it uses this knowledge to improve future decisions. A **Static Forwarding** strategy does not change its preferences, and can rely exclusively on the decisions made by the routing protocol. In this case, to stop using a face, a static forwarding strategy requires the routing protocol to remove the face from the FIB.

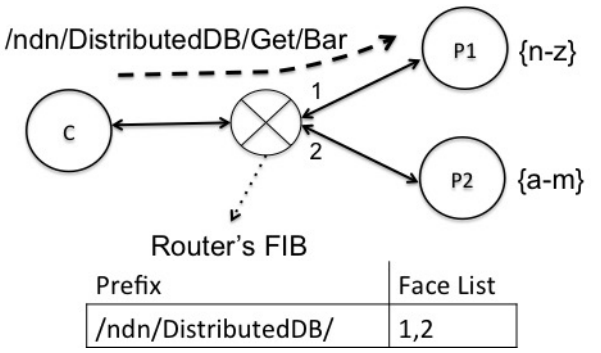

Figure 2: Static Strategy with Multiple Producers

4. IN-NETWORK RETRANSMISSIONS

In this section, we discuss a few scenarios in which applications can benefit from the in-network retransmissions mechanism provided by the strategy.

Ad-hoc and dynamic environments can benefit from in-network retransmissions that can quickly recover from a change in the network topology or a link failure. Those environments can rely exclusively on an adaptive forwarding strategy to eliminate the overhead created by a routing protocol. In such cases, strategy retransmissions can be also used to probe the network and find a better path to the application producer.

Even with the support of a routing protocol in a relatively static topology, an in-network retransmission mechanism that triggers Interest retries by the strategy could recover from a link failure much faster than a routing protocol, and therefore could better support multipath consumer-producer services.

Another interesting use case consists of a multiple producer service, in which each producer holds a disjoint part of the data provided by the service. Figure 2 presents an example of such an application, a distributed database in which the stored information is distributed among multiple producers. Here, in-network retransmissions can immediately explore possible outputs, and guarantee that the correct output face will be used for every Interest expressed by the application, without having to differentiate the producer namespaces, and without forcing application retransmissions. Here, in-network retransmissions can be crucial to support the application correctness since the forwarding strategy cannot rely on either a routing decision or on previously collected forwarding measurements when selecting the next hop output.

One could claim that a better namespace design, such as one that differentiates producer namespaces to reflect the content held by each producer, would be a better approach. While we agree that the namespace design is crucial, we believe that some applications would not be able to follow such a namespace design, or would not understand the importance of this requirement in the designing phase of the application and namespaces. Moreover, we believe that the NDN platform should be able to support new types of applications, and therefore should not enforce strict rules on the namespace design due to strategy characteristics.

Another use case also consists of multiple producers, but simplifies the previous scenario by having a complete replication of the content in all existing consumers. In this case, when one producer is overloaded with requests, and therefore fails to respond to new Interests, in-network retransmissions can immediately probe other faces and find another operating producer to fetch the data from. Here, the link remains active and responsive, and it is the producer who drops the packets; therefore a routing plane may fail to detect the problem, and would not inform the forwarding plane of a face failure.

It may be that the use cases above could be resolved by application retransmissions, without any retransmission mechanism in the strategy layer. However, this approach could result in greater round-trip-times and would impact the application's performance, In addition, the application has to consider details of the suppression mechanism used by the strategy, if such a mechanism exists. If the strategy does implement a suppression timer algorithm, as in the best-route strategy, the application might face a big challenge. 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 following software versions. For instance, if the suppression mechanism of the strategy determines that, to avoid suppression, an Interest must be retransmitted X ms after the first attempt, then the application has to follow the requirement and implement its retransmission code blocks and timers according to the X ms requirement. Any strategy modification to this X ms suppression requirement would require a modification in the application code as well. Therefore, the application implementation must be coupled with the strategy design.

It is important to note that, while some applications require or could benefit from in-network retransmission, others request complete control of their traffic, and avoid network retransmissions. For instance, a video streaming application such as the one described in [11] builds its own traffic control mechanism to support a continuous and interruption-free communication. An example for such a mechanism is relying on the network RTT to determine the streaming rate at any point in time. In this scenario, network retransmission could negative impact the application's measurements, and therefore degrade the correctness and performance of the application.

Therefore, we suggest that the decision to perform

in-network retransmission should be made by the application, and only executed by the forwarding strategy. We discuss our proposal in detail in the next section.

5. RETRANSMISSION ABSTRACTION

Our suggested retransmission mechanism performs two independent yet complementary functions, application abstraction and retransmission differentiation.

5.1 Application Abstraction

As described in previous sections, we strongly believe that it is the application that needs to decide if an Interest should be retransmitted by the network or not. Therefore, we suggest adding a new TLV to the Interest packet to specify the application retransmission policy. We name this TLV the 'Interest Retransmission Policy' (IRP) flag.

By using IRP, the application can determine the policy for every expressed Interest. A forwarding strategy supports this policy by providing two retransmission mechanisms as part of its implementation, one that supports in-network retransmissions and another that supports application retransmissions.

Algorithm 1 presents a simplified framework for a forwarding strategy that supports both retransmissions mechanisms by checking the IRP flag after receiving an Interest.

```
Function ForwardInterst(interest):
    face_list = SelectNextHop(Interest)
    IRP = GetIRP(Interest)
    SendInterest(faces)
    if IRP then
        schedule retransmission at time x
    else
        set suppression timer for application
        retransmission
    end
    return
```

Algorithm 1: Application Abstraction Framework of a Forwarding Strategy

It is important to note that the IRP flag does not determine the in-network retransmission algorithm, but only requires that one exists. Therefore, 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 examples of possible retransmission algorithms provided by a forwarding strategy. The strategy is free to choose any algorithm to support the two options. It may be that a core network strategy would choose a retransmission algorithm that address congestion issues and relies on collecting round-trip-times, while an access strategy retransmission algorithm would simply follow a list of given faces and retransmit an Interest after a fixed time interval.

5.2 Retransmission Differentiation

We believe that the NDN forwarding plane should be able to differentiate an Interest expressed by an application from an Interest injected or retransmitted by the network. Therefore we suggest adding a second TLV to the Interest packet to differentiate application Interests from network retransmissions. We name this TLV the 'Network Retransmission' (NR) Field.

We describe two scenarios in which the NR field is required. First, in dynamic networks, such as in a vehicular network [12] or in access wireless networks[6], where an adaptive forwarding strategy can probe faces to explore additional next-hops. In such cases, it may be useful for the strategy to differentiate the probing Interest from others, and therefore to support different strategy mechanisms for control and data traffic.

Another scenario is an existing problem in the current ncc strategy and the NDN forwarding mechanism, in which loop detection caused by nonces can prevent better face exploration. As presented in figure 3, R1 has a faster path to R4 through R3, but the ncc strategy previously selected R1 as the best performing face. This can happen if R3 was previously congested and therefore had longer RTT times to R4. As explained in the Background section, R1 maintains prediction timer that approaches the best-face actual RTT. When the prediction timer is adjusted down to a value that causes an Interest timeout on face 1, R1 retransmits the Interest on face 2. Router R4, which previously received the Interest from R2, recognizes the Interest 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 RTT is twice than face 2. The ncc strategy will switch to use face 2 only if the retransmitted Interest arrives at R4 before the original Interest. In other words, the strategy will change its bets-face selection only if the 'prediction' plus the 'one way trip time through R3' is less than the 'one way trip time through R2'.

Although the described problem is unique to the specific ncc implementation in the NDN architecture, this problem can also occur in other adaptive forwarding strategies that ask to explore potential faces.

This problem could be solved by adding the NR TLV to the retransmitted Interest of ncc strategy in NFD 0.4. In the proposed solution, the ncc strategy differentiates between the retransmitted Interest and the original one, and does not detect the Interest as a duplicate

one, thus enabling better face exploration. By adding NR TLV and processing Interests with the same nonce, we interrupt the core mechanism of loop detection in NDN. Therefore, another mechanism, such as the TTL TLV in CCN, should be supported to avoid forever forwarded Interests. In our implementation, we used a non-negative-integer to represent NR TLV. We set the initial value of the NR 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 NR 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 NR mechanism should be better explored as part of future work. We present the implemented loop detection in algorithm 2.

Unlike NDN, CCN does not use nonces to detect loops, and therefore the problem described in 3 might not occur in CCN. However, we argue that the proposed differentiation can be useful in the CCN architecture as well to supports more intelligent forwarding strategies that can differentiate an application Interest from an Interest injected by the network.

```
Function DetectLoop(interest):
    face_list = GetUnusedFaces(Interest)
    if face_list is empty then
        send NACK
    else
        if (nonce previously recorded) AND (NR is
        equal to 10) then
            send NACK
        else
            ForwardInterst(i) [algorithm 1]
        end
    end
    return
```

Algorithm 2: Loop Detection using NR

We implemented the proposed retransmission mechanism in NFD 0.4 by adding the two suggested TLVs to the Interest packet, and by modifying the loop-detection mechanism that follows algorithm 2.

6. EMPIRICAL RESULTS

We tested the proposed in-network retransmission abstraction by running a set of experiments using the emulated NDN testbed in the open Network Lab (ONL) [13, 14]. The emulated environment consists of 26 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 as used by the corresponding world-wide NDN testbed [14] gateway, and ran NLSR 0.2.2 as the network routing protocol to distribute the gateways' namespaces. The emulated testbed also consists of 66 links that are configured with costs that match the 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 4 presents our emulated gateways topology.

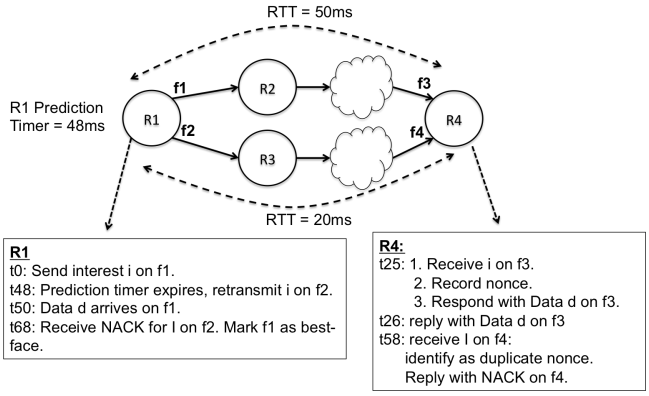

Figure 3: Nack problem

We modified the best-route and ncc strategies to check the IRP flag in order to determine if in-network retransmission is required by the application, and used NR 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.

We designed three experiments that demonstrate the impact of in-network retransmissions on the application correctness, and evaluated 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 WU gateway, and two instances of ndn-traffic-server as multiple producers on the VMS connected to ORAMGE and KISTI gateways. We configured both servers to respond to the Interests sent by consumer, hence, both producers hold 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

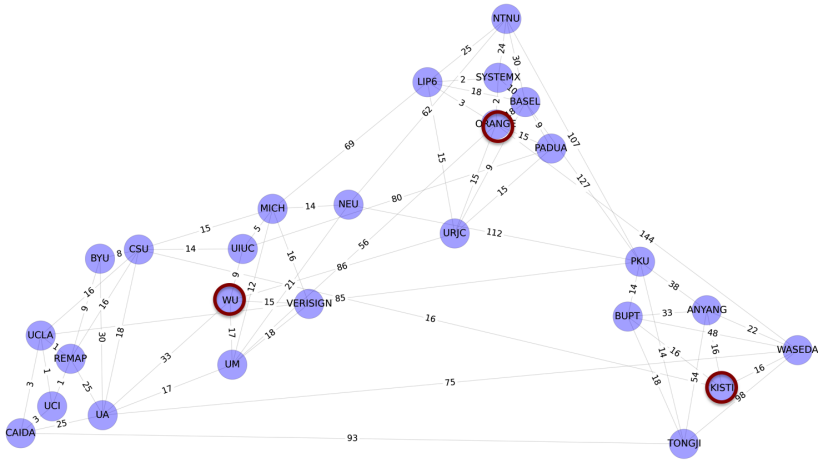

Figure 4: Emulated NDN testbed

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

during the run of the experiment. According to the testbed link costs, it is cheaper to get from WU to ORANGE than it is to KISTI, and therefore we selected ORANGE VM to be the congested producer. We set the consumer's Interests IRP flag to True, and therefore required in-network retransmission from the strategy. We did not provide any retransmission mechanism for unsatisfied Interest 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 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 packet received from each producer, and the number of Interest 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 bets-route-r, ncc or ncc-r. In addition, table 1 shows that the number of Interests sent by WU gateway when using bets-route-r was twice the number of 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, that is shorter than the actual round-trip time in the used topology. This represents a detail in the in-network retransmission mechanism that should be better explored as part of future work. However, this experiment demonstrates that a simple change to the best-route strategy, such as the support of IRP flag, can dramatically improve unsatisfied Interest rate in the case of multiple producers with congested node, and therefore supports a wider range of applications.

It is important to note that our statistical analysis of the results did not point on any statistical difference between ncc and ncc-r, Therefore, we can conclude and report that the support of IRP flag does not change the performance and correctness of strategies that already support in-network retransmissions as part of their default implementation.

6.2 Multiple Producers with Congested Producer and Congested Gateway

To emulate a congested gateway, we repeated the previous set of experiments and configured WU gateway with a different drop rate each time. We used drop rates of 5%, 20% and 50%. Figure 5 presents the unsatisfied Interest rate of each of the explored strategies. As shown by the figure, in the best-route strategy the rate of unsatisfied Interests reaches to an average of 70% when the congested gateway drops 50% of the packets. However the unsatisfied Interests rate remains less than

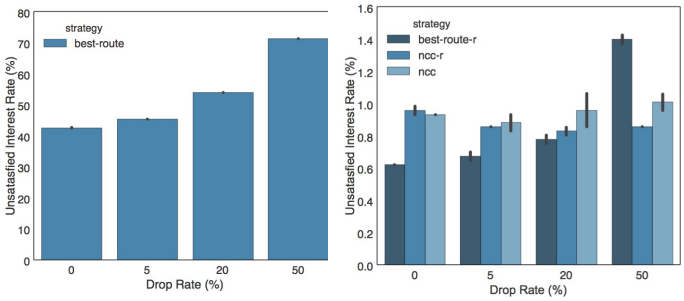

Figure 5: Unsatisfied Interest Rate over Different Loss Rate

1.5% when using best-route-r, ncc and ncc-r. This 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 node is congested and drops 50% of the packets. As in the previous experiment, our results did not point on any statistical difference between ncc and ncc-r.

6.3 Multipath with Congested Link

In this experiment, we used a simple consumer-producer service using ndn-traffic as the consumer running on WU VM, and ndn-traffic-sever as the producer running on KISTI VM. As before, we modified the consumer to set IRP to True, and did not support any application retransmission mechanism. 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 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. 10 seconds later, we configured the link between WU and UIUC to drop all application packets sent by WU gateway. We recorded the traffic for 120 seconds before removing the dropping filter. We continued to record measurements for additional 120 seconds before we stopped the consumer's traffic.

Figure 6 shows the traffic as recorded on the producer and consumer VMs when using the best-route strategy, and figure 7 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 WU links. We report the results received when using the best-route strategy in figure 8, and the results received using the best-route-r

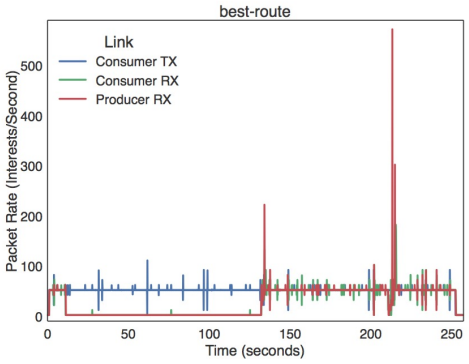

Figure 6: End Hosts Traffic Over Time with best-route

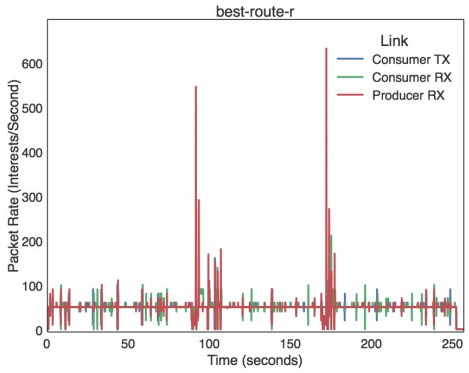

Figure 7: End Hosts Traffic Over Time with best-route-r

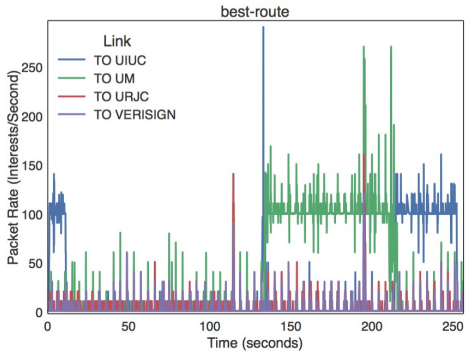

Figure 8: WU Traffic Over Time with best-route

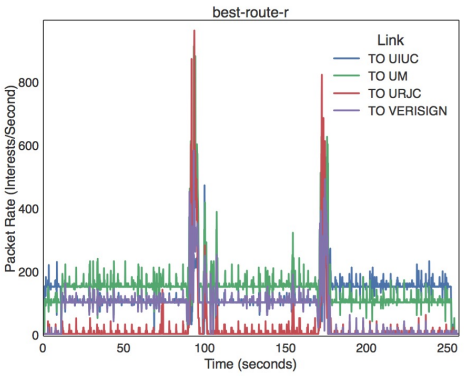

Figure 9: WU Traffic Over Time with best-route-r

strategy in figure 9.

As shown in figure 8, due to 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 became congested and dropped all the Interest packets transmitted by the consumer, the traffic on this link dropped to almost zero. It is important to note that 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 does not present an absolute zero. At t=135, NLSR determined the link to UM as 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 terminated dropping UIUC packets, NLSR determined UIUC as the least expensive next hop again, and rerouted the traffic towards that link.

Figure 9 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, 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. This is achieved due to the IRP flag set by the application, and executed by the best-route-r strategy. The strategy continues to use UM link until UIUC becomes available again.

It is important to clarify that our intention in this experiment was not to compare forwarding recovery time to routing convergence time [9], but to demonstrate how multipath consumer-producer service can maintain a continuous traffic flow even when the network is congested, and without forcing the application to implement its own retransmission mechanism as required by the existing best-route strategy. Moreover, as can be seen in figure 9, the Interest rate sent by the best-route-r strategy on 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, and should be better explored by strategy developers.

7. CONCLUSION AND FUTURE WORK

In this paper, we discuss one of the core mechanisms of a forwarding strategy in NDN: the in-network retransmission mechanism when an Interest remains unsatisfied. In many cases, the application has to be coupled with the details of the paired forwarding strategy to guarantee correctness of its retransmission policy. We argue that this coupling can be easily interrupted when the selection of the forwarding strategy is overwritten by the network operator. By adding the IRP flag to the Interest packet, we propose an abstraction to the in-network retransmissions mechanism in which the application decides whether an in-network retransmission is required, and leaves the strategy to decide how. By using this abstraction, an application can maintain its correct traffic flow regardless of the underlying strategy, and can eliminate its dependency on the strategy in-network retransmission policy. In addition, we propose adding the NR TLV to differentiate retransmitted Interests from others.

In this work, we focus on proposing the retransmission abstraction and differentiation, and made simple modifications to existing strategies to support it. However, the details of the in-network retransmission mechanism, 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 mechanism that outperforms others for 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 NR 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 NSF grants CNS-1040643 and CNS-1345282. 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] Alexander Afanasyev et al. Nfd developer's guide. Technical report, NDN-0021, NDN, 2014.

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

[3] Marc Mosko. Ccnx 1.0 protocol specification roadmap. 2013.
[4] Project CCNx. `http://www.ccnx.org/`.
[5] Cheng Yi et al. A case for stateful forwarding plane. *Computer Communications*, 2013.
[6] 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.
[7] Hila Ben Abraham and Patrick Crowley. Forwarding strategies for applications in named data networking. In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, pages 111–112. ACM, 2016.
[8] 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.
[9] 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.
[10] Alberto Compagno et al. To nack or not to nack? negative acknowledgments in information-centric networking. *arXiv preprint arXiv:1503.02123*, 2015.
[11] 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.
[12] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. Navigo: Interesa ct 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.
[13] 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.
[14] NDN Testbed. `http://ndnmap.arl.wustl.edu/`.