Reining in Redundant Traffic through Adaptive Duplicate Suppression in Multi-Access NDN Networks

Saurab Dulal
University of Memphis
Memphis, TN, USA
sdulal@memphis.edu

Lan Wang
University of Memphis
Memphis, TN, USA
lanwang@memphis.edu

ABSTRACT

Named Data Networking (NDN) provides native support for multi-party communication. However, the current NDN forwarder lacks a duplicate suppression mechanism for multicasting in a multi-access network, potentially leading to network congestion and significant degradation in overall packet delivery performance. In this paper, we introduce Adaptive Duplicate Suppression (ADS) for one-hop multicasting in multi-access NDN networks. ADS utilizes the duplicate count per Interest and Data name observed in the network to dynamically adjust the suppression time that a node waits before forwarding a packet. We have implemented ADS in the NDN forwarding daemon (NFD) and assessed its performance using Mini-NDN. Our evaluation demonstrates that ADS can effectively reduce redundant network traffic under various network conditions, resulting in significantly improved application goodput and reduced transfer times.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Wireless access networks**; **Wired access networks**; **Ad hoc networks**.

KEYWORDS

Named Data Networking (NDN), Multicast Suppression, Duplicate suppression, Multi-access Networks

ACM Reference Format:
Saurab Dulal and Lan Wang. 2023. Reining in Redundant Traffic through Adaptive Duplicate Suppression in Multi-Access NDN Networks. In *10th ACM Conference on Information-Centric Networking (ACM ICN '23), October 9–10, 2023, Reykjavik, Iceland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3623565.3623717

1 INTRODUCTION

Named Data Networking (NDN) [23] is a data-centric Internet architecture that can be incrementally deployed over the current Internet. Departing from the conventional host-centric approach, NDN introduces a paradigm where each data unit is uniquely named and digitally signed. Data consumers request data using Interest packets containing the data names. Data packets are signed and optionally encrypted by the producer at the time of creation. As such, NDN can decouple data from their producers – any node with a copy of some data can serve it and the data consumer can verify the data authenticity using the data signature and a trust schema associated with the data name [22].

NDN's name-based data-centric approach naturally supports multi-party communication (one-to-many and many-to-many), which is crucial in modern applications such as vehicular networks, online gaming, video conferencing, and disaster management. It is especially suitable for wireless applications where identifying data producers can be challenging, and their addresses may change over time. For example, a vehicle traveling on a highway with poor cellular coverage may need to get the area map, but it may not know which other vehicles are willing to provide the information. With NDN, the vehicle can name the data using the highway name, the closest exit number, and information type, and send an Interest containing the data name through broadcast. The Interest may be forwarded multiple hops, and whichever vehicle willing to provide the information can send back a Data packet containing the requested information.

As we started developing NDN applications over wireless networks [6–8], however, we discovered that ***the current NDN forwarder lacks a duplicate suppression mechanism for multicasting in a multi-access network, which can cause a high level of network congestion***. For example, if several consumers fetch the same map simultaneously, many Interests and Data packets of the same name may be sent over the same link, leading to high losses and delays. This will severely impact the application performance and reduce users' quality of experience. Let us illustrate the problem of duplicate Interest and Data packets using an example. In Figure 1(a), node A multicasts an Interest (I) with the name `/I40/exit213/map` to all the other nodes on the wireless network at time 0. After 1ms, another node D multicasts an Interest with the same name. This situation can occur when D sends its Interest before receiving A's Interest, due to propagation delay. It can also occur after D receives A's Interest – if D does not have a route for the name prefix, D will drop A's Interest without recording it in its Pending Interest Table (PIT) (see Section 2 for more information about PIT). In either case, D's PIT cannot prevent D duplicate Interest because A's Interest is not recorded in D's PIT. The problem becomes worse as the number of consumers increases. A similar case can occur for the corresponding Data packet. For example, if several nodes in the network, except for A and D, have the data, all of them can reply at once, as shown in Figure 1(b). Thus, without suppressing duplicate traffic, network congestion may increase significantly.

ACM ICN '23, October 9–10, 2023, Reykjavik, Iceland

ACM ISBN 979-8-4007-0403-1/23/10…$15.00
https://doi.org/10.1145/3623565.3623717

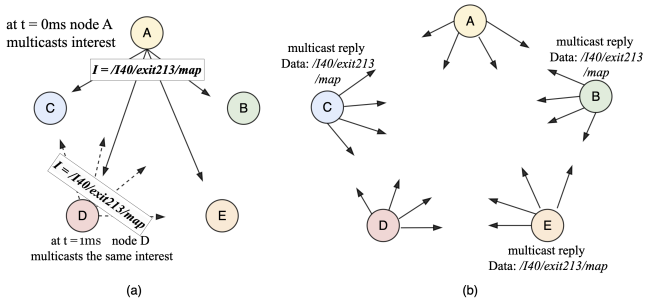

Figure 1: Duplicate NDN Interest and Data Packets in a Wireless Network

We note that duplicate suppression is different from collision avoidance [12]. Duplicate suppression aims to prevent redundant traffic from entering the network so there will be less traffic overall. On the other hand, collision avoidance takes place after duplicate suppression, and it prevents outgoing packets from colliding with each other. Collision avoidance does not suppress any packets received from the upper layers. We also point out that NDN nodes can identify and suppress duplicate traffic at the *network layer* because every Interest/Data packet is uniquely identified by a name visible at the network layer. Such duplicate suppression is much more difficult in non-ICN architectures such as TCP/IP due to the lack of data names at the network layer.

In this paper, we focus on duplicate suppression in *single-hop scenarios* where consumers can reach producers in one hop. In multi-hop scenarios, a node may overhear an Interest and still decide to send its own Interest of the same name since the producer may not be directly reachable by the original Interest. Although our focus in this work is not multi-hop duplication suppression, ADS can serve as a basis for developing a solution tailored to multi-hop scenarios.

A straightforward approach to suppressing duplicates is using random wait intervals [3, 21]. In this technique, a node waits for a random time period between 0 and a preconfigured maximum waiting time before forwarding an Interest or Data packet. If another Interest or Data packet with the same name is overheard during the waiting period, the node cancels its own transmission. However, this approach fails to adapt to network conditions. For instance, in scenarios where there is a large number of nodes requesting the same data, the random wait interval should be longer. Additionally, in a lossy network, it may be preferable to allow some duplicates, whereas in a stable network, minimizing duplicates is more advantageous.

In this paper, we propose *Adaptive Duplicate Suppression (ADS)* to reduce redundant NDN Interest and Data packets in multi-access networks such as WiFi and Ethernet networks. In ADS, each node keeps track of the count of duplicates per Interest and Data name in the network, and dynamically adjust the suppression time, i.e., the average wait time before forwarding an Interest or Data packet, based on the duplicate count. If an Interest or Data packet of the same name has been recorded recently or transmitted during the wait time by another node, this node will not forward its own copy. Our evaluation shows that ***ADS can reduce redundant network traffic significantly in different network conditions, resulting in much higher application goodput and lower file transfer time***. In a 9-node wireless topology consisting of 1 producer and 8 consumers, the producer receives 80% fewer Interests and sends 83% fewer Data packets with ADS compared to scenarios without duplicate suppression. Similarly, each consumer receives 80% fewer Interests and 75% fewer Data packets, and sends 81% fewer Interests with ADS. In addition, ADS increases the consumers' goodput by more than 5 times and reduces their file transfer time by 82%.

The key contributions of our work are summarized as follows:

- We designed the ADS scheme to effectively eliminate duplicate Interests and Data packets within single-hop Multi-Access NDN networks. ADS relies solely on observation of the received packets, without introducing additional protocol messages among network nodes.
- We implemented ADS in NFD's Face System for multicast-capable faces only, ensuring that non-multicast packets do not experience extra delay. Additionally, our implementation does not require any modifications to the adaptation layer code and does not affect existing NFD Interest and Data pipelines or forwarding strategies.
- We performed an emulation-based evaluation of ADS in networks of various sizes. Our evaluation shows that, compared with random suppression and no suppression, ADS reduces significantly more redundant traffic without introducing much extra delay, thus achieving superior performance in terms of bandwidth utilization, total transfer time, and per-packet latency.

The remainder of the paper is structured as follows. In Section 2, we provide a brief overview of NDN's packet forwarding process. In Section 3, 4, and 5, we present our design, implementation, and evaluation results, respectively. We review existing work related to our research in Section 6 and discuss two design issues in Section 7. Finally, we conclude our work and outline future directions in Section 8.

2 NDN PACKET FORWARDING

An NDN forwarder typically consists of four key modules: Pending Interest Table (PIT), Forwarding Information Base (FIB), Content Store (CS), and Forwarding Strategy. The CS caches previously received data to fulfill future Interests. If an incoming Interest does not match any stored data in the CS, the forwarder utilizes the FIB and a forwarding strategy to determine the appropriate interface(s) for forwarding the Interest. The PIT keeps track of forwarded Interests that have not been satisfied yet. When multiple Interests with the same name are received, the forwarder forwards the first Interest and aggregates the subsequent ones using the PIT. Upon receiving a Data packet, the forwarder transmits it to all the incoming interfaces associated with the matching Interests recorded in the PIT. This capability enables multicast data delivery to multiple interfaces.

An NDN packet can be sent over various communication channels, such as physical interfaces, Unix sockets, and UDP/TCP tunnels. These communication channels are called *faces* in NDN – an NDN face is a network interface to another node or an intra-node communication channel to another process on the same node. In the NDN Forwarding Daemon (NFD) [19], a face implementation consists of two components: Link Service (upper component) and

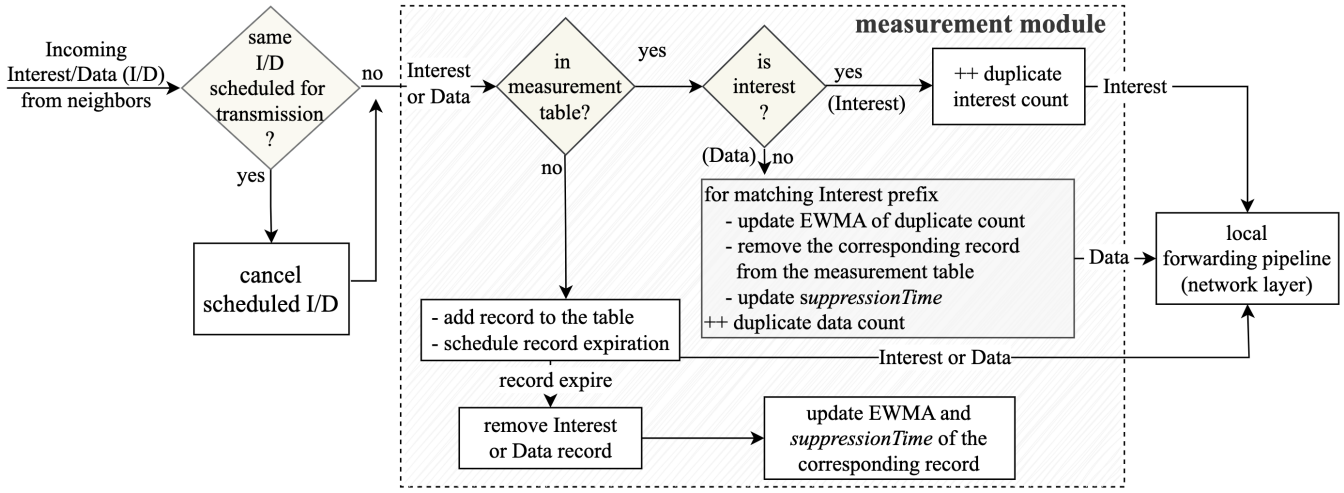

Figure 2: Incoming Interest/Data Pipeline in ADS

Transport (lower component). Link Service is primarily responsible for translating between network layer packets and lower layer packets. It also provides additional services such as fragmentation, link failure detection, and retransmission. Transport encapsulates the underlying communication mechanism (e.g., UDP, Ethernet, WebSocket) and provides a best-effort packet delivery service. We implemented ADS in the Link Service module.

3 ADS DESIGN

In this section, we outline our design goals (Sectiion 3.1), and present key elements of our design, including the measurement module (Section 3.3), incoming packet processing (Section 3.4), outgoing packet processing (Section 3.5), and Adaptive Suppression Time calculation (Section 3.6).

3.1 Goals

Our main goal is to reduce duplicate NDN traffic in multi-access networks in order to improve application performance. Moreover, the design should allow users to control the level of duplicate traffic based on the network condition, e.g., allow more duplicate packets in a lossy environment. In addition, the design should automatically adjust the time period a node waits before sending a packet based on the amount of duplicate traffic in the network, e.g., use a longer waiting time when the observed duplicate traffic is higher than the target level. In this paper, we focus on single-hop scenarios where the producers and consumers can reach each other in one hop.

3.2 Design Overview

In ADS, every node uses a measurement table to record the number of Interest and Data packets of the same name sent to or received from a multicast face. Whenever ADS receives an Interest or Data packet for transmission, it checks whether the packet is already in the measurement table. If so, it drops the packet. Otherwise, it sets a waiting timer based on the suppression time computed using the algorithm in Section 3.6. While waiting, if the same Interest/data is overheard, the forwarding is canceled. We use a user-specified target duplicate count called *duplicate threshold* and the *observed duplicate count* to adjust the suppression time value.

3.3 Measurement Table

The measurement table maintains duplicate count information for Interests and Data packets. When a new record is created, the initial duplicate count is set to 1. Subsequently, if the same Interest or Data packet is received again, the corresponding record's duplicate count is incremented. Each record in the table has a relatively short lifetime. Typically, we set this to twice the maximum propagation delay[1] within the multi-access network. This ensures that in an ideal scenario, a node will be able to receive and record all the duplicates related to a record before the record expires. Expired records are removed from the table. Additionally, an Interest record is also removed upon receiving the matching Data packet. Upon removal of a record, the suppression time for the corresponding Interest or Data name prefix is updated using the duplicate count present in the record (refer to Section 3.6). Note that while each duplicate count is associated with a specific Interest or Data name, a suppression time estimate is associated with a name prefix rather than a full name, and it is computed for Interest and Data name prefixes separately, so it applies to all the Interests or Data packets under the corresponding name prefix.

Let's use an example to illustrate how the measurement table works. Suppose a consumer is fetching */alice/videoA* which is divided into multiple data segments */alice/videoA/1*, */alice/videoA/2*, ..., and */alice/videoA/N*. We measure the duplicate counts of individual packets for this video, and compute two suppression times, one for all the Interests under */alice/videoA* and one for all the Data packets under */alice/videoA*. One may compute a suppression time for the shorter prefix */alice*, but there can be many videos under */alice*, and the suppression time for one may not be applicable to the others.

[1]Propagation delay is the time for a packet to travel from one node to another, including time spent in any layer-2 network and receiver logic.

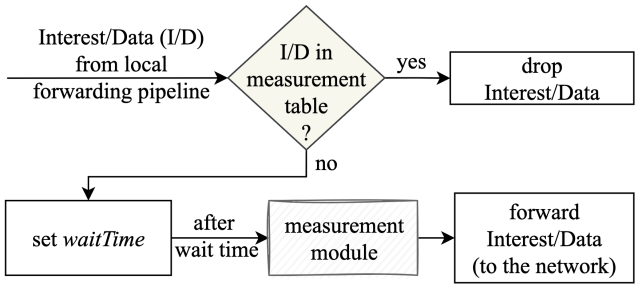

Figure 3: Outgoing Interest/Data Pipeline

Determining the appropriate prefix for computing the suppression time is an open issue [15]. In the current implementation, we use the granularity of the full name minus the last component.

3.4 Incoming Packet Processing

When an NDN forwarder receives an Interest or Data packet from a multi-access network (Figure 2), ADS checks whether the same packet is already scheduled for transmission. If so, the scheduled transmission is canceled (i.e., suppressing a duplicate). Then ADS checks whether a record of the packet already exists in the measurement table. If not, a new record is added to the measurement table. If there is already a record for the packet, ADS determines whether the packet is an Interest or Data packet. For an Interest, the corresponding duplicate count is increased. For a Data packet, ADS performs the following actions; i) calculate the Exponentially Weighted Moving Average (EWMA) of the matching Interest's duplicate count (Section 3.6), ii) remove the matching Interest record from the table, iii) update the suppression time of the Interest name prefix, and iv) increment the data duplicate count. After the above processing steps, the Interest/Data packet is sent to the forwarder's local forwarding pipeline for further processing.

3.5 Outgoing Packet Processing

After a forwarder receives a packet from a local application and determines that the packet should be forwarded to a multicast face (Figure 3), ADS first checks the measurement table for a corresponding record. If a record exists, which indicates that this is a duplicate packet, the packet is dropped. However, if there is no matching record, ADS waits for a period of time (referred to as *waitTime*) before forwarding the packet (Figure 3). If the same Interest or Data packet is overheard during this time, the scheduled forwarding is dropped. Once the wait timer expires, the packet is sent to the measurement module (Figure 2) and eventually forwarded to the network. The *waitTime* is computed using the mechanism described in Section 3.6.

3.6 Adaptive Suppression Time

To maintain the Interest and Data duplicate counts below a user-specified duplicate threshold (e.g., 1.5), which is set based on the environment's loss rate, ADS dynamically adjusts the Adaptive Suppression Time. When a record is removed from the measurement table, ADS calculates the Exponential Weighted Moving Average

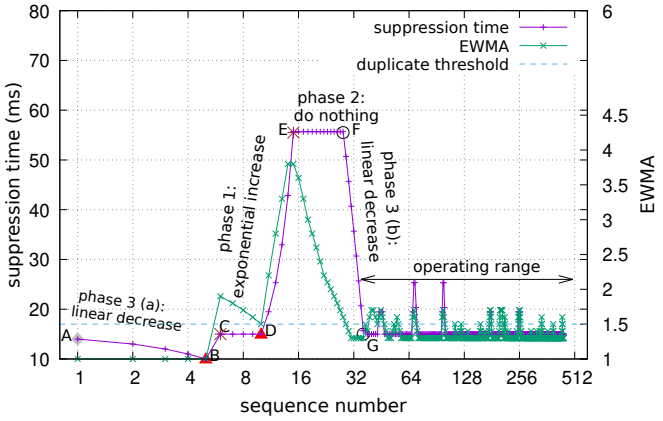

Figure 4: *Phases of Suppression Time*. The graph is drawn using data from one of our experiments with 1 producer and 7 consumers.

(EWMA) [13] of the duplicate count (c) for the corresponding Interest or Data name prefix. The EWMA (e) is calculated using the following formula (i denotes a new instance of EWMA computation):

$$e_i = \begin{cases} c_1 & i = 1 \\ a * c_i + (1 - a) * e_{i-1} & i > 1 \end{cases} \quad (1)$$

where a is a *smoothing factor* between 0 and 1. We use $a = 0.125$ in our implementation.

ADS starts with a *minimumSuppressionTime*, which is set to the maximum propagation delay between two nodes in the network in our experiments. Note that *minimumSuppressionTime* can be set to a small value, such as 1ms, but setting it to a higher value equivalent to the propagation delay helps the algorithm detect early duplicates and converge faster.

Next, we use the EWMA of the duplicate count to dynamically adjust the suppression time for a name prefix in three different phases (Figure 4) to achieve an optimal operating range.

Phase 1: Exponential Increase The suppression logic enters Phase 1 (B-C and D-E in Figure 4) when two conditions are met: a) e is above the duplicate threshold and is increasing, and b) this node has contributed to duplicating the packet. The fulfillment of both conditions indicates that the suppression time is not large enough to prevent nodes from sending the packet too quickly. Consequently, we multiply the suppression time (t) by a constant factor m, i.e.

$$t_i = t_{i-1} * m,$$

to discourage premature forwarding and mitigate the aggressive duplication of the same Interest or Data packet.

Phase 2: Do Nothing If e satisfies one of the following two conditions: a) e is below the duplicate threshold but increasing, and b) e is above the threshold but decreasing or remains constant, the suppression logic enters the Do Nothing phase (C-D and E-F in Figure 4). As the name suggests, the suppression time is carried forward from the previous instance. i.e.

$$t_i = t_{i-1}.$$

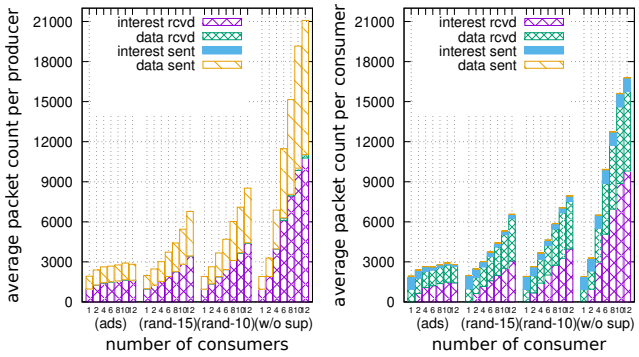

Figure 5: Average packet count for ADS, rand-15, rand-10, and no duplicate suppression (w/o sup) at the producer (left) and at each consumer (right) when using IEEE802.11g. The number of consumers is 1, 2, 4, 6, 8, 10, and 12.

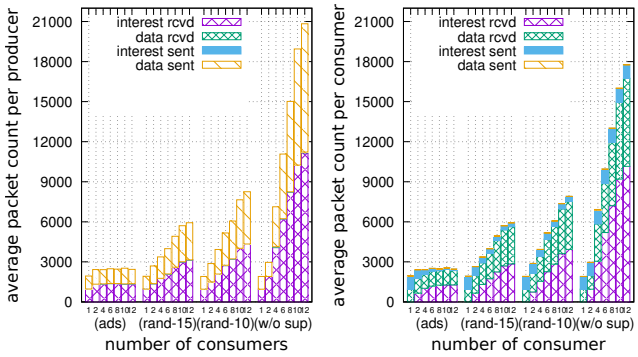

Figure 6: Average packet count for ADS, rand-15, rand-10, and no duplicate suppression (w/o sup) at the producer (left) and at each consumer (right) when using IEEE802.11b. The number of consumers is 1, 2, 4, 6, 8, 10, and 12.

The rationale to carry forward the suppression time in condition (b) is that even though e is decreasing, the goal of maintaining the duplicate count below the threshold is still not achieved.

Phase 3: Linear Decrease This phase is divided into two sub-phases 3(a) and 3(b). The suppression logic enters 3(a) in the initial stage. If a node does not receive any duplicate packets from other nodes, it decreases the suppression time by 1ms until it reaches 0 (A-B in Figure 4). In this case, this node is the only sender of the packet, so duplicate suppression is not required. However, once the node records the first duplicate, the suppression time is reset to the *minSuppressionTime* (B-C in Figure 4). The occurrence of the first duplicate hints at the existence of other nodes in the network that are also interested in the same data and hence can cause duplication. The suppression logic enters phase 3(b) (F-G in Figure 4) if e is below the threshold and decreasing. This phase happens when the computed suppression time is sufficiently large for nodes not to generate duplicates but it may not be optimal. Thus, we decrease the suppression time by a constant n, i.e.

$$t_i = t_{i-1} - n \ (n \text{ is typically larger than 1ms}).$$

The suppression logic switches among the three phases to dynamically adjust the suppression time, eventually reaching an optimal operating range for a given network condition. For example, in Figure 4, we can see that after sequence number 37, the suppression time oscillated between 15ms and 25.35ms throughout the rest of the experiment. Thus, 15 – 25.35 is the operating range discovered by ADS for the given setup.

Finally, we compute the *waitTime* (w_i) using the suppression time (t_i) as follows:

$$w_i = rand(0, 2 * t_i) \quad (2)$$

4 IMPLEMENTATION

We implemented ADS in the NDN Forwarding Daemon (NFD) [19] using the master branch of both ndn-cxx [17] and NFD. It is implemented in the Face class of NFD rather than the forwarding pipeline for two reasons. First, duplicate suppression is applicable only to multicast-capable faces, so putting this functionality in the Face class and enabling it only for multicast-capable faces means that it will not unnecessarily delay the packets going to non-multicast faces. Second, this placement will not require any changes to NFD's forwarding pipeline or the adaptation layer code.

We added ADS to the Link Service module, instead of the Transport Module, in NFD's Face System. The Transport module is supposed to provide a simple transmission functionality so it is more suitable for the Link Service module to make the decision on whether to send a packet.

We have implemented a name tree to store the suppression time. When choosing suppression time for a packet before forwarding, ADS traverses the tree for an exact match of the packet's name prefix and gets the corresponding suppression time.

5 EVALUATION

We compared ADS against the current NFD (i.e., no duplicate suppression) and two random suppression strategies, one with a 10ms maximum suppression time (rand-10) and the other with a 15ms maximum suppression time (rand-15), in Mini-NDN [16].

Before selecting suppression times of 10ms and 15ms for random suppression, we conducted a series of ping experiments between nodes to estimate the propagation delay between them. These experiments yielded propagation delays of approximately 2.5 ms and a round-trip time (RTT) of 5 ms for our network topologies. Subsequently, we experimented with various suppression time values for random suppression, ranging from 5ms (equivalent to the RTT) to 4 times the RTT (i.e., 20ms), which are adequate for nodes to hear each other in a lossless environment. We chose 10ms and 15ms because they performed better than lower or higher suppression times in our setup.

With the random suppression strategy, a node will wait for a random time period between 0 and the maximum value before forwarding an Interest or Data packet. While waiting, it will drop the scheduled transmission if the same Interest or Data packet is received.

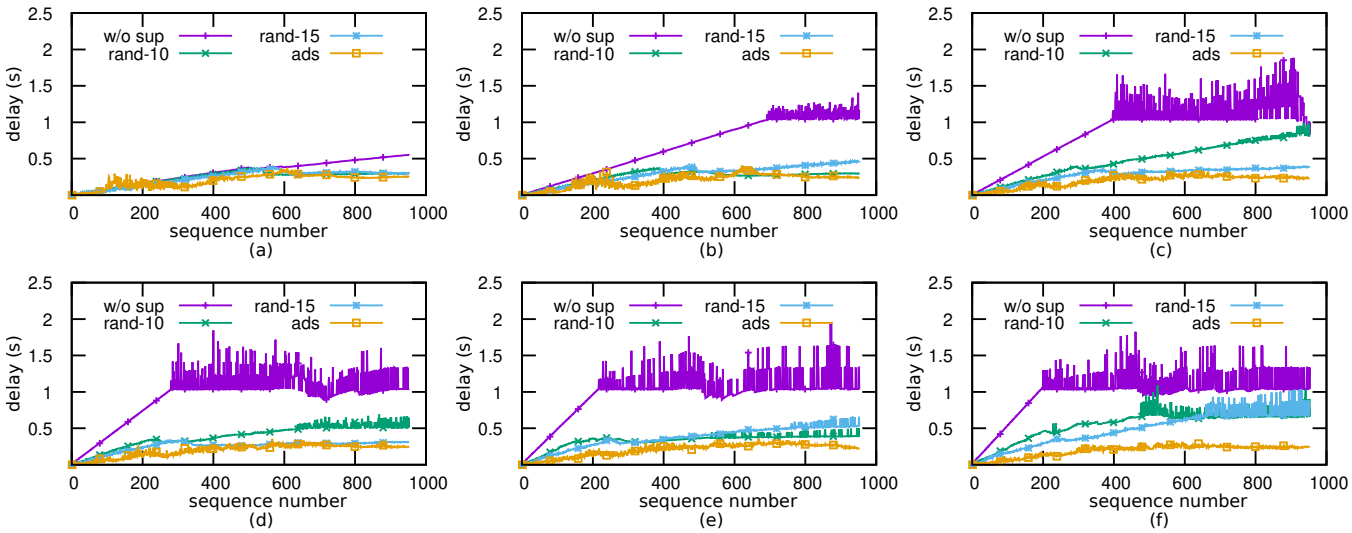

Figure 7: Data fetching delay with an increasing number of consumers ((a) two consumers, (b) four consumers, (c) six consumers, (d) eight consumers, (e) ten consumers, (f) twelve consumers) when using IEEE802.11g

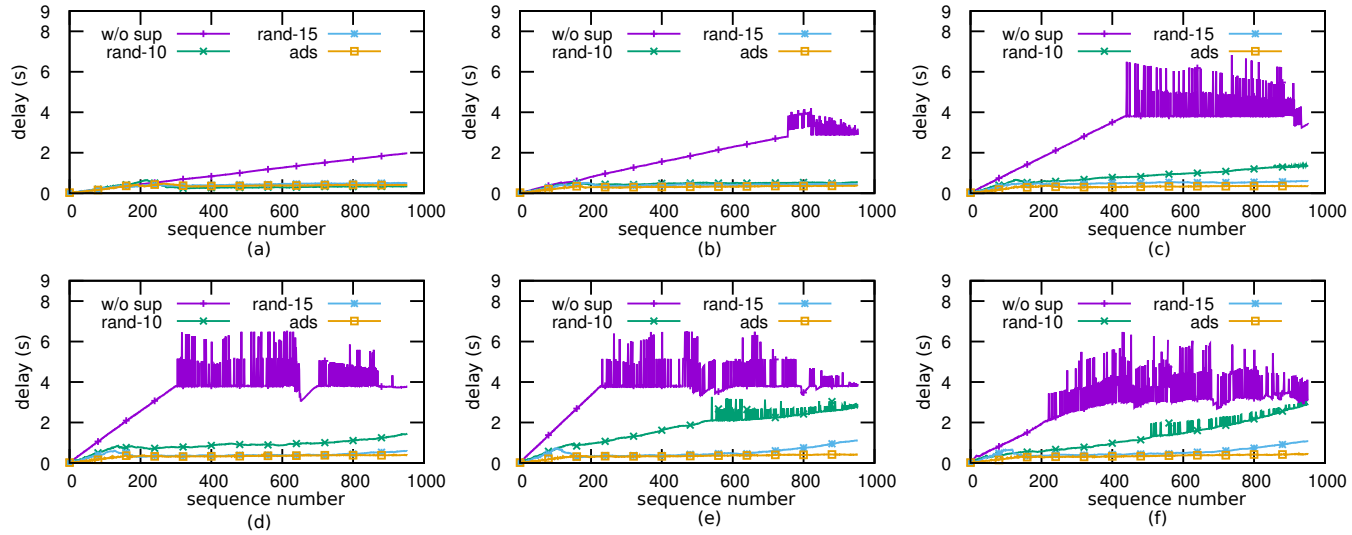

Figure 8: Data fetching delay with an increasing number of consumers ((a) two consumers, (b) four consumers, (c) six consumers, (d) eight consumers, (e) ten consumers, (f) twelve consumers) when using IEEE802.11b

5.1 Emulation Setup

Our experiment topologies consist of 1 producer and varying number of consumers connected to a WiFi access point. On every node, NFD automatically creates a UDP multicast face on the WiFi interface. In all the experiments, the producer publishes a 1MB file using `putchunks` [18], and the consumers fetch the file over their multicast face using `catchunks` [18]. The producer (`putchunks`) divides the 1MB file into 1100-byte data segments, so a total of 956 unique NDN Data packets are fetched by each consumer. We use 1100 bytes instead of the default segment size of 8800 bytes because of a known problem – excessive re-transmissions and timeouts in `catchunks` when using the default size [20].

The actual number of Interest and Data packets sent and received by each node may be much higher than 956 due to duplicates and retransmissions after packet losses. Note that we did not enable NFD to store unsolicited data. If NFD were to store unsolicited data, the duplicate packet problem would be even worse, as more nodes would reply to an Interest with the same data. We used a duplicate threshold of 1.3, and the parameters $a = 0.125$, $m = 1.3$, and $n = 5$ in adjusting the adaptive suppression time (Section 3.6). We use

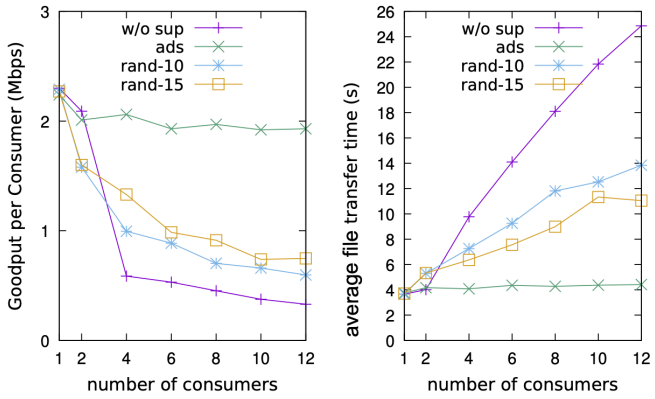

Figure 9: Goodput (left) and file transfer time (right) vs number of consumers when using IEEE802.11b

IEEE802.11g and IEEE802.11b, which support up to 54Mbps and 11Mbps, respectively, for our experiments.

We performed two different types of experiments: i) simultaneous fetch and ii) delayed fetch.

- *Simultaneous fetch:* in this experiment, the producer first publishes the file, and after a few seconds, all the consumers fetch the file simultaneously. This will introduce a large number of duplicate Interest packets from the consumers, as they tend to fetch the same segment at the same time. The main goal of this experiment is to evaluate *Interest* suppression performance. We performed simultaneous fetch experiments for ***IEEE802.11g*** and ***IEEE802.11b***.
- *Delayed fetch:* in this experiment, the producer first publishes the file. After two seconds, $N-1$ (all but one) consumers fetch the file simultaneously. After four seconds, the N^{th} consumer starts fetching. The N^{th} consumer's Interest is likely to be answered by both the producer and the previous $N-1$ consumers, as they already have the corresponding data in their content store. This will introduce a large number of duplicate Data packets into the network. Hence, the main goal of this experiment is to evaluate *Data* suppression performance. We conducted delayed fetch experiments only for ***IEEE802.11g***.

We repeated each experiment 10 times and computed the average of the results.

5.2 Performance Metrics

We use the following metrics in our evaluation:

- **File transfer time** is the time for a consumer to fetch a complete file.
- **Goodput** is the transmitted file size divided by the file transfer time.
- **Average packet count** is the average number of packets recorded at a node, including any transmitted or received Interest or Data packets.
- **Data fetch delay** is the time for a consumer to fetch an individual Data packet. In the event of retransmission, the delay is the duration from the initial attempt until the data is eventually received.
- **Aggregate duplicate count** refers to the total number of packets transmitted by the nodes in the network minus the expected number of packets.
- **Duplicate packet suppression** represents the percentage of duplicate packets suppressed by the nodes.

5.3 Results

We present our evaluation results in the rest of this section.

5.3.1 Average Packet Count. Figure 5 and 6 show the average packet count for **IEEE802.11g** and **IEEE802.11b**, respectively. As we can see from the figures, **ADS reduces the average packet count substantially at the producer and consumers**, and the reduction increases with the number of consumers. When there are 12 consumers in an ***IEEE802.11g*** network, with ADS duplicate suppression, the producer receives 54% fewer Interest packets from the consumers (1595 vs 3446) and sends out 64% fewer Data packets (1195 vs 3331) compared with the second best strategy (i.e., **rand-15**). Similarly, on average each consumer receives 53% fewer Interest packets (1432 vs 3071) and 62% fewer Data packets (1225 vs 3191), and sends out 81% fewer Interest packets (159 vs 858) compared with rand-15. The numbers are very similar for **IEEE802.11b**.

5.3.2 Data Fetch Delay. Figure 7 and 8 show the data fetch delay for IEEE802.11g and IEEE802.11b, respectively. For two consumers, the performance across all schemes is similar, with the exception of the slightly inferior performance observed in the absence of suppression. Additionally, for the initial 100 segments, the delay appears to be volatile for the ADS scheme, as it strives to determine an optimal operating range.

With an increasing number of consumers, ADS consistently achieves the lowest data fetch delay among all the schemes, and this delay remains stable even with a higher number of consumers. While the delays of ADS and rand-15 may appear comparable for 4, 6, and 8 consumers in Figure 7 and 8, a closer examination of Figure 9 and 10 reveals that ADS outperforms rand-15 significantly in terms of both goodput and file transfer time (see the next section).

5.3.3 Goodput and File Transfer Time. In Figure 9 and 10, we observe that the goodput and file transfer time are almost the same for all the schemes when there are 2 consumers. However, as the number of consumers increases, ADS outperforms the other schemes significantly. For example, with 12 consumers and IEEE802.11g, the goodput is 6.55, 2.51, 2.01, and 0.83 Mbps for ADS, rand-15, rand-10, and no duplicate suppression, respectively. The file transfer time is 1.25, 3.35, 4.31, and 10.11 seconds for those schemes. The goodput of ADS is almost 8 times that of no suppression and 2-3 times that of random suppression. It is important to note that the available bandwidth is only 54 Mbps with IEEE802.11g, which is shared among all consumers. This means that each consumer should theoretically achieve a maximum goodput of 4.5 Mbps if the bandwidth is divided among them equally. However, as we can see from the results, consumers using ADS were able to achieve 6.55 Mbps. This is because one Data packet can satisfy multiple consumers in NDN.

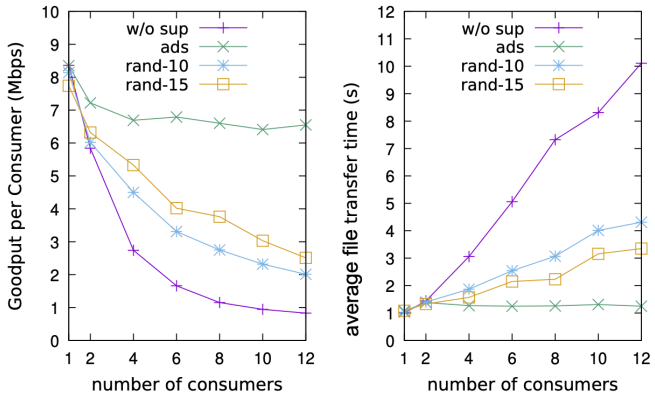

Figure 10: Goodput (left) and file transfer time (right) vs number of consumers in simultaneous fetch when using IEEE802.11g

Table 1: Aggregate goodput with constrained link bandwidth

Link BW	Aggregate Goodput (Mbps)			
(Mbps)	w/o sup	rand-10	rand-15	ads
1	0.21	0.37	0.44	**0.60**
3	0.63	1.18	1.41	**2.24**
6	1.24	2.32	2.82	**4.26**

Table 2: Average file transfer time with constrained link bandwidth

Link BW	Average File Transfer Time (s)			
(Mbps)	w/o sup	rand-10	rand-15	ads
1	39.39	22.06	18.99	**15.8**
3	13	7.08	5.95	**3.78**
6	6.79	3.62	2.98	**1.97**

5.3.4 Constrained Bandwidth Experiment. In this experiment, we use the topology with 1 producer and 6 consumers but change the link bandwidth, both egress and ingress, at the producer to 6, 3, or 1 Mbps after 1 second into the file transfer. Note that we use IEEE802.11g which supports up to 54Mbps. Table 1 and 2 show that, with very constrained link bandwidth, ADS has much higher aggregate goodput at the consumers and lower file transfer time compared to the other schemes. For example, with a 1 Mbps link bandwidth, ADS achieved an aggregate goodput of 0.6 Mbps, almost 3 times the aggregate goodput without suppression, and only 40% of the file transfer time. This strongly suggests that a duplicate suppression module is a crucial component for multicasting in a multi-access environment, especially for networks with poor link conditions and multiple consumers.

Table 3: Number of duplicate packets suppressed by rand-10, rand-15, and ADS, respectively, in delayed start experiments

	dup interest suppressed			dup data suppressed		
topo	rand-10	rand-15	ADS	rand-10	rand-15	ADS
1p4c	63.90%	**74.97%**	66.05%	1.23%	7.69%	**21.55%**
1p6c	69.94%	80.16%	**86.20%**	1.39%	8.13%	**50.54%**
1p8c	67.61%	79.89%	**93.34%**	2.26%	3.58%	**63.87%**
1p10c	68.06%	79.40%	**94.71%**	1.95%	3.12%	**85.60%**

5.4 Delayed Start and Data Suppression

To demonstrate the *Data* suppression capabilities of ADS, we performed a series of delayed start experiments with IEEE802.11g. The experiment results are shown in Table 3.

In terms of *Interest* suppression, with 4 consumers, rand-15 was slightly better than ADS. As the number of consumers grows, however, ADS reacts to the growth in duplicate count by increasing the suppression time more aggressively, resulting in a significant reduction of the Interest duplicates. For example, when there are 10 consumers, 94.71% of the duplicate Interests were suppressed by ADS compared to only 79.4% suppressed by rand-15.

In terms of *Data* suppression, ADS performs significantly better than the other schemes, up to 26 times better than the second-best scheme (rand-15). For example, with 1 producer and 10 consumers, ADS suppressed 85.60% of duplicate data, whereas rand-10 and rand-15 suppressed 1.95% and 3.12%, respectively. The main reason is that when there is severe congestion, a fixed maximum value for the random waiting timer is insufficient for nodes to hear from each other before they send their own packets. The problem with using random timer in data suppression has also been highlighted by Elbadry et. al. [9]. With ADS, when the duplicate count of Data packets recorded by a node increases above the target threshold, the suppression time value will be increased exponentially. This is why we observe that ADS effectively suppresses duplicate Data packets.

6 RELATED WORK

Previous research efforts [1–3, 11] have proposed suppression schemes to maximize bandwidth utilization. These schemes utilize the concept of a defer timer to handle Interest flooding scenarios. The schemes proposed by Amadeo et. al. [1–3] do not provide an adaptive solution and only consider Interest suppression. Gao et. al. [11] sets the defer timer based on node transmission distance and residual energy for sensor networks. However, this solution does not have an adaptive defer timer for Data suppression.

Li et. al. proposed a leader-based Interest suppression scheme [14]. The access point keeps track of the order of Interests received from the clients. It chooses a client, whose Interest is received first, as the leader to send future Interests before the others. All the other clients, known as followers, delay or suppress duplicate Interests to reduce contention. However, their approach has several limitations: i) it does not support Data suppression, ii) the leader can be a bottleneck, and the other nodes may not get a fair chance to send Interests, and iii) the scheme may fail to adapt to various

Table 4: Impact of exponential increase factor m on suppression time and duplicate count (the linear decrease factor n is 5 and the duplicate threshold is 1.5 for all the experiments)

m (exponential increase factor)	average computed suppression time (ms)	average duplicate count
1.1	23.95	1.90
1.2	27.04	1.82
1.3	34.15	1.63
1.4	56.95	1.58
1.5	66.71	1.4
1.6	154.18	1.33

network conditions. For example, in a lossy network, if the leader's Interests are frequently lost, the AP may have to select new leaders constantly.

In [21], the authors proposed suppressing duplicate Interests by waiting for a random amount of time between 0 and a preconfigured maximum value, and if the same Interest is overheard during the wait, the node cancels its own transmission.. As we discussed in Section 7, this scheme cannot automatically adapt to changes in the number of consumers and producers.

In [10], the authors achieve Data suppression in a conferencing architecture using a delay timer (t), which is set based on the distance from the control server and conference participants. They use *Interest Lifetime* to approximate the distance, and each intermediate router reduces the *Interest Lifetime* by its processing time. Once the timer goes off, the node sends back the response along with a multicast suppression Interest message. Others suppress their responses upon receiving the suppression message. This scheme introduces a new message type and alters the processing of Interest Lifetime. In contrast, our scheme does not require any new messages or modifications to the processing of existing messages.

Elbadry et al. proposed OPSEL [9], an optimal producer selection mechanism for wireless edge environments with multiple producers providing data redundancy. OPSEL allows single and multiple consumers to consistently identify and select the best producer(s) according to predefined performance criteria in dynamically changing network conditions. Their work focuses on suppressing duplicate Data packets, but our work addresses both Interest and Data suppression using a single scheme.

7 DISCUSSION

7.1 Parameter Tuning

Our evaluation demonstrates that ADS can automatically adapt to different number of consumers, varying link bandwidth settings, and delayed start scenarios. However, we recognize that more work is needed to understand how to select appropriate ADS parameters for different networks.

First, the duplicate threshold is an important parameter in our algorithm. Its value depends on application requirements and link loss rate. If the application is delay sensitive, a higher duplicate threshold is preferable to reduce the suppression time. Moreover, a more lossy network may also use a higher duplicate threshold to compensate for the link losses.

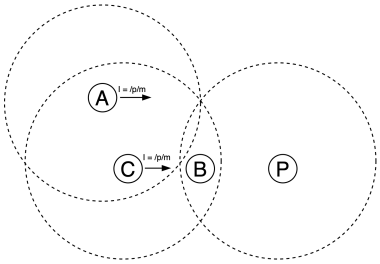

Figure 11: Multi-hop Duplicate Suppression Scenario

Second, it is crucial to identify a pair of exponential increase factor m and linear decrease factor n that can quickly establish a stable and effective operating range for the suppression time, so as to minimize duplicate packets without introducing much additional delay. Choosing a larger value for m can rapidly increase the suppression time when necessary, but it also runs the risk of overestimation. Conversely, opting for a smaller m may slow down the iterative process to reach the stable operating range but reduce the likelihood of overestimating the suppression time. Similar tradeoffs exist for the value of n. We conducted several experiments to understand the effect of different m and n values on duplicate suppression using a network of one producer and three consumers. The duplicate threshold is set to 1.5. Table 4 shows that, as we increase the exponential increase factor m from 1.1 to 1.6 while keeping the linear decrease factor n constant at 5, the average suppression time increases from 23.95ms to 154.18ms and the average duplicate count for each Interest/Data packet decreases from 1.9 to 1.33. Since the target duplicate threshold is 1.5, an m of 1.5 (along with an n of 5) may be chosen for this emulated network given that it leads to an average duplicate count of 1.4. In this set of experiments, we fixed the value of n, but ideally one should evaluate different combinations of m and n to find the best combination.

The straightforward approach to selecting the above parameters involves protocol designers and users conducting tests to determine the appropriate parameters for different network and application scenarios (as we did above). However, a more promising approach is for our algorithm to autonomously adjust the parameters based on the observed delay introduced by the algorithm, observed duplicate count, and estimated link loss rate. This represents an area for future research.

It is important to note that even with a fixed set of parameters, ADS can automatically adapt to changes in the number of consumers and producers, as well as variations in link bandwidth. This adaptability arises from our algorithm's dynamic adjustment of suppression time based on observed duplicate counts. This offers a distinct advantage over the random timer approach, which requires manual tuning even for minor network changes, such as an increase in the number of consumers.

7.2 Multi-hop Duplicate Suppression

Our algorithm can be used as a basis for multi-hop duplicate suppression. However, in a multi-hop scenario, a node may need to

send an Interest even if the Interest has been overheard before, as the first Interest may not be able to reach the producer due to connectivity issues. For example, in Figure 11, A and C both have an Interest I to send. A sends I first, but it happens that there is no route from A to the producer P. C then overhears I. If it cancels its own transmission, it will not receive the corresponding data as the original Interest will not reach the producer. Instead, since C has a route through B to reach P, it should still sends its Interest. In other words, nodes with a better chance to reach the producer should not cancel their transmissions. The challenge lies in the fact that individual nodes often lack certainty regarding whether a specific Interest, be it their own or someone else's, will successfully reach the producer. To make informed suppression decisions, they may need to assess their location and network connectivity to the name prefix in comparison to the original Interest sender [4, 5].

8 CONCLUSION AND FUTURE WORK

NDN needs an effective duplicate suppression mechanism in multi-access networks in order to improve packet delivery performance. We have proposed ADS to reduce redundant NDN Interest and Data packets in multi-access networks. ADS relies solely on observation of the packets received, without introducing additional protocol messages among network nodes. Our evaluation shows that ADS can reduce redundant network traffic significantly in different network conditions, resulting in much higher application goodput and lower file transfer time.

In our future work, we will investigate ways to reduce the number of parameters employed in our algorithm. One potential approach is to study the correlations among the parameters. Furthermore, we will explore the feasibility of using machine learning to tune the parameters and implement the entire suppression logic. Finally, we plan to conduct experiments in single-hop wireless ad-hoc scenarios and eventually expand our work to encompass multi-hop scenarios.

ACKNOWLEDGMENT

We thank our shepherd Marc Mosko and the anonymous reviewers for their insightful comments and feedback which helped improve the quality of the paper. In addition, this work was supported by the National Science Foundation award 1629769.

REFERENCES

[1] Amadeo, M., Campolo, C., and Molinaro, A. Multi-source data retrieval in iot via Named Data Networking. In *Proceedings of the 1st ACM Conference on Information-Centric Networking* (2014), ACM, pp. 67–76.

[2] Amadeo, M., Campolo, C., and Molinaro, A. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications 50* (2015), 148–158.

[3] Amadeo, M., Campolo, C., Molinaro, A., and Mitton, N. Named Data Networking: A natural design for data collection in wireless sensor networks. In *2013 IFIP wireless days (WD)* (2013), IEEE, pp. 1–6.

[4] Bouk, S. H., Ahmed, S. H., Park, K.-J., and Eun, Y. Interest broadcast suppression scheme for named data wireless sensor networks. *IEEE Access 7* (2019), 51799–51809.

[5] Chowdhury, M., Khan, J. A., and Wang, L. Leveraging content connectivity and location awareness for adaptive forwarding in NDN-based mobile ad hoc networks. In *Proceedings of the 7th ACM Conference on Information-Centric Networking* (2020), pp. 59–69.

[6] Dulal, S. NDNSD: Service publishing and discovery in NDN. *University of Memphis Thesis* (2020).

[7] Dulal, S., Ali, N., Thieme, A. R., Yu, T., Liu, S., Regmi, S., Zhang, L., and Wang, L. Building a secure mHealth data sharing infrastructure over NDN. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022), pp. 59–69.

[8] Dulal, S., and Wang, L. Adaptive duplicate suppression for multicasting in a multi-access NDN network. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022), pp. 156–158.

[9] Elbadry, M., Ye, F., and Milder, P. OPSEL: optimal producer selection under data redundancy in wireless edge environments. In *Proceedings of the 9th ACM Conference on Information-Centric Networking* (2022), pp. 22–32.

[10] Fesehaye, D., and Wei, J. SNC: scalable NDN-based conferencing architecture. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)* (2014), IEEE, pp. 872–880.

[11] Gao, S., Zhang, H., and Zhang, B. Energy efficient interest forwarding in NDN-based wireless sensor networks. *Mobile Information Systems 2016* (2016).

[12] Gummalla, A. C. V., and Limb, J. O. Wireless medium access control protocols. *IEEE Communications Surveys & Tutorials 3*, 2 (2000), 2–15.

[13] Hunter, J. S. The exponentially weighted moving average. *Journal of quality technology 18*, 4 (1986), 203–210.

[14] Li, M., Pei, D., Zhang, X., Zhang, B., and Xu, K. Interest-suppression-based NDN live video broadcasting over wireless LAN. *Frontiers of Computer Science 11*, 4 (2017), 675–687.

[15] Liang, T., Shi, J., and Zhang, B. On the prefix granularity problem in NDN adaptive forwarding. In *Proceedings of the 7th ACM Conference on Information-Centric Networking* (2020), pp. 41–51.

[16] NDN Project Team. Mini-NDN – a lightweight NDN emulator. http://minindn.memphis.edu/. Accessed on 09/21/2023.

[17] NDN Project Team. ndn-cxx: NDN C++ library with eXperimental eXtensions. https://github.com/named-data/ndn-cxx. (Accessed on 09/21/2023).

[18] NDN Project Team. NDN Essential Tools. https://github.com/named-data/ndn-tools. (Accessed on 09/21/2023).

[19] NDN Project Team. NFD: Named Data Networking Forwarding Daemon. https://github.com/named-data/nfd. (Accessed on 09/21/2023).

[20] NDN Project Team. IP fragmentation causes ndncatchunks to time out indefinitely. https://redmine.named-data.net/issues/5035, 2020. Accessed on 09/21/2023.

[21] Podder, P., Gupta, S. D., Neishaboori, A., and Afanasyev, A. sv2pc: On scaling LTE-based vehicle-to-pedestrian communication using NDN. In *NDN Community Meeting, NDNComm* (2021), NIST.

[22] Yu, Y., Afanasyev, A., Clark, D., Jacobson, V., Zhang, L., et al. Schematizing trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking* (2015), ACM, pp. 177–186.

[23] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. Named Data Networking. *ACM SIGCOMM Computer Communication Review 44*, 3 (2014), 66–73.