OPSEL: Optimal Producer Selection under Data Redundancy in Wireless Edge Environments

Mohammed Elbadry, Fan Ye, Peter Milder
Stony Brook University
New York, USA
{mohammed.elbadry,fan.ye,peter.milder}@stonybrook.edu

ABSTRACT

In wireless edge environments, data redundancy among multiple neighboring nodes is common due to the need to support application performance, mitigate faults, or the intrinsic nature of applications (e.g., AR/VR, edge storage). Further, under data centric paradigms (e.g., Named Data Networking (NDN)), consumers that request the same data may leverage multicast so data are sent only once (e.g., VR games with data cached at multiple edge nodes). Naive strategies such as selecting a random neighbor or the prevailing wisdom of choosing the one with the strongest received signal strength (RSSI) cause more severe loss than other available producers. In this paper, we propose OPSEL, a single-hop dynamic producer(s) selection protocol that enables single and multiple consumers to continuously identify the optimal producer(s) (e.g., lowest loss) under constantly varying medium conditions. When Data is available single-hop, OPSEL's goal is to have the minimum number of producers sending to all consumers and meeting their performance needs without explicit coordination messages. Experiments on a real prototype show that OPSEL is 3% away in loss rate and has the same latency as the theoretical ideal, while naive timer methods can incur up to 60% more loss and 2-3× latency.

CCS CONCEPTS

• **Networks** → **Mobile ad hoc networks**.

KEYWORDS

Data centric MAC, Producer selection, Wireless medium access

ACM Reference Format:
Mohammed Elbadry, Fan Ye, Peter Milder. 2022. OPSEL: Optimal Producer Selection under Data Redundancy in Wireless Edge Environments. In *9th ACM Conference on Information-Centric Networking (ICN '22), September 19–21, 2022, Osaka, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3517212.3558090

1 INTRODUCTION

Data redundancy is common in many emerging wireless edge applications: a consumer node may have multiple *one-hop* neighbors possessing desired data. As long as the content matches the consumer's "Interests," any such neighbor can respond [1]. For example, in AR/VR [2, 3], the content produced by an edge server could be cached at and transmitted via multiple access points as the user moves around; in distributed edge storage [4], multiple nodes keep redundant data copies to mitigate the impact of faults [5, 6].

Such single hop redundancy creates the problem of *producer selection*: which producer should be selected to ensure the optimal reception (e.g., lowest loss) for all the consumers? Due to complex wireless propagation, attenuation and contention, each producer may generate different reception quality. If all producers respond, their concurrent transmissions cause severe contention, thus loss and latency. Choosing a random producer may result in inferior performance, and our experiments find that even some prevailing wisdom (e.g., the one with the strongest Received Signal Strength Indicator (RSSI)) leads to much worse results than the optimal one. Further, more than one concurrent producer might be needed to meet all the consumers' performance requirements.

The data-centric NDN paradigm introduces filtering and routing flexibility by describing the content instead of nodes' identities, enabling a richer set of wireless techniques by nature (e.g., multicast, overhearing Data of Interest). NDN's Network Forwarding Daemon (NFD) [1] is a mature, widely adopted data-centric network layer; it handles multihop producer selection, routing, and filtering based on *datanames*. However, the network layer selection does not have access to wireless reception quality information from the MAC layer, and its decisions incur significant delay passing across layers, making it too slow to respond to fast-varying wireless medium.

Existing work does not address single hop optimal producer selection in wireless edge environments. A selection mechanism is needed to choose the optimal producer throughout the whole duration of a possibly long data transfer that consists of pipelined Interests. For multihop, NFD has each producer that receives the Interest start a random timer. The producer whose timer runs out first responds [7]. However, such approach works when the producers are multihop. In single hop, each producer may generate a different reception quality (i.e., loss, latency) at the consumer-end. A random selection results in mostly mediocre and occasionally poor reception quality. Another intuitive approach is to select the one with the strongest Received Signal Strength (RSSI). We find that this can cause more than 50% higher loss compared to the optimal producer, because RSSI does not correlate strongly with loss rate.

We face multiple challenges in optimal producer selection design: *i)* Selecting a producer on per packet granularity causes high overhead; a larger, flexible granularity that continues to use a qualifying producer over multiple packets is preferred for pipelined Interests. *ii)* In Named Data Networking (NDN), nodes do not have

ICN '22, September 19–21, 2022, Osaka, Japan

ACM ISBN 978-1-4503-9257-0/22/09…$15.00
https://doi.org/10.1145/3517212.3558090

any identity and rely on *datanames* for filtering and routing; yet a more persistent method to identify and thus select neighboring producers beyond packet granularity is needed. *iii)* The selection must be quick and efficient even for an active transmission to all consumers, ideally minimizing explicit coordination messages.

In this paper, we design OPSEL, a dynamic optimal producer(s) selection protocol for both single and multiple consumers: we introduce the concept of *stream* which is an application semantic granularity that contains pipelined Data packets, during whose transmission the reception remains relatively stable. We also design an *identification beacon* that producers use to announce their transient IDs and the Data streams they can provide within one hop neighborhood. Our optimal producer(s) selection protocol leverages the initial Interest packets to sample different producers to explicitly measure their performance, then pick the optimal producer(s).

We make the following contributions:

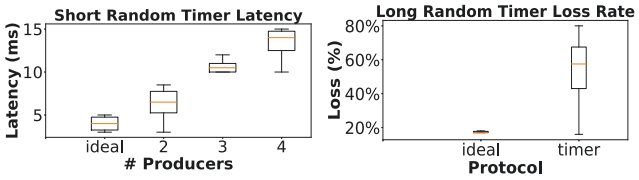
- We propose *stream*, an application level data granularity that allows optimal producer(s) selection on longer time scales than individual packets while retaining NDN's policy of One-Interest-One-Data transmission.
- We design an identification beacon that allows streams that need optimal producer selection in one-hop wireless medium be known. A consumer selects by indicating a producer's transient ID, eliminating redundant data transmissions. OPSEL supports both single and multiple concurrent consumers selecting optimal producers to meet all their performance requirements.
- We implement the prototype on an existing data-centric wireless stack. Experiments show that our prototype cuts loss rates by up to 60% and latency by 2–3× compared to the common wisdom of using a timer.

To the best of our knowledge, we are the first to point out the seriousness and impact of the single hop producer selection problem under data redundancy in wireless edge environments, and the first to propose an efficient, scalable optimal selection design for wireless communication.

2 BACKGROUND

Named Data Networking Forwarding Daemon (NFD) [1] is a data-centric network forwarder and core component of the Named Data Networking (NDN) platform. NDN network packets are *Interest* and *Data*. NFD uses a Pending Interest Table (PIT) to route and filter Data packets accordingly. It uses 1:1 mapping between Interest and Data packets. For a consumer to receive Data, the consumer has to send an Interest specifying the *dataname* of the Data it needs. Once the Interest is received by a producer for the Data (i.e. has a Data packet with matching *dataname*), it responds to the Interest packet by sending the Data packet matching the *dataname*.

NFD serves as a network layer handling multi-hop routing, filtering and reliability. Underneath the network layer, a Medium Access Control (MAC) is usually deployed to handle the physical medium. V-MAC [8] is a wireless MAC layer designed to support the NDN network layer [1] to provide a full data-centric stack. It uses *Interest* and *Data* frames for requests and responses, and filters incoming frames based on *encoding*, which is a hash of the network layer *dataname* representing the content. A network layer packet



(a) Short Timer Latency **(b) Long Timer Loss Rate**

Figure 1: (a) shows latency increase using short timer durations which lead to multiple redundant transmissions increasing latency by up to 5x. (b) shows long random timer duration results in 1 producer sending the data with high loss rate variations.

contains one *dataname* and corresponds to one hash of *encoding* that may be fragmented into multiple MAC frames. V-MAC uses a *Lingering Encoding Table* (LET) to store the encoding of any Interest sent by the node with a *timeout* duration. A node filters a received Data frame's *encoding* against entries in the LET: the frame is passed to the network layer if a match is found; otherwise it is dropped because no application requested that data in the preceding timeout duration. Besides name-based filtering in MAC, V-MAC offers a multicast robustness protocol that enables consumers to request lost frames through DACK frames without redundant Data frame retransmissions where it reduces loss rates from 50–90% to under 3% with slight latency increase.

NFD is a network stack that handles network layer protocols agnostic to the medium used, and V-MAC is wireless medium-dependent layer designed on top of 802.11 PHY. In this paper, we design wireless one-hop producer selection within the MAC layer because it needs to access information available only in the MAC, and only decisions made within the MAC are executed fast enough to adapt to varying wireless medium.

3 WHY RANDOM TIMERS FAIL

In this section, we demonstrate the insufficiency of a random back-off timer in wireless networks: each producer waits a random time; if one hears another transmitting, then it does not send data. We divide the experiments into two categories: *short random timer range* and *long random timer range*. Our experiments show that using a short random timer leads to multiple producers transmitting the same data (thus more overhead, contention, and latency). A long random timer ensures only one producer sends, but frequently a suboptimal one with serious loss is chosen.

Setup. We set up experiments outdoors with no active background traffic (confirmed through wireshark, a wireless monitoring tool). We have 5 Raspberry Pi's as 4 producers and 1 consumer. All Pi's are running V-MAC [8], a data-centric MAC layer that uses encodings (hash of datanames) to filter wireless frames. The consumer sends an Interest requesting a Data content of 37.5KB at the application layer, which translates into 25 Data frames (1500 bytes each) at the Link layer. The Interests are sent at 6.5Mbps data rate[1]

[1]The data rate is selected arbitrarily to be above baseline and provide reasonable goodput to support general applications (e.g., high resolution video real-time transmission, time-sensitive Data, etc.)

and producers send data at 65Mbps. If a producer hears a single Data frame of the same *dataname* (every frame carries encoding, a hash of the name) transmitted, it cancels its timer and does not send. We define an 'ideal' producer as the one giving the lowest loss rate and latency.

Short random timer (up to 2ms) increases latency. Our experiments show that as the number of producers increases, the backoff timer may not work. By the time a producer overhears another producer sending, its timer may have already expired and it has started sending. Figure 1a shows as more producers' redundant transmissions cause contention, latency increases from 3ms up to 15ms (5×). Such latency increase comes from an observed 2–3x data redundancy, wasting medium bandwidth. Therefore, a short random timer does not work well in real systems due to the medium (propagation time and contention), and internal system (TX/RX queues) latency.

Long random timer (up to 5ms) increases loss rate latency. We have 4 producers with varying loss rates (20%, 35%, 67%, and 80% respectively). Figure 1b shows the performance of a random producer sending based on a long backoff timer. A random producer creates widely varying loss rates ranging 10%–80%. It is hard to compensate for such variations even with application level mechanisms (e.g., redundant coding like FEC). The latency is 6–7 ms, about 2x latency of ideal due to long random timer range. This shows that a random backoff timer may avoid redundant transmission, but with no input from the consumer, it results in mostly mediocre performance and occasionally the worst producer. There is a large space of improvement to approach the ideal performance.

Besides the above issues, there are more challenges in single hop wireless systems: *i)* the timer length should adapt to the level of contention for appropriate waiting. This is difficult to do on the fly under dynamic medium and hardware processing latency; *ii)* we find a timer backoff for every Interest has high overhead and reduces the system's goodput up to half at high data rates. Longer waiting (2 to 5 ms) will further reduce the system's goodput multifold.

Therefore, an approach of selecting the producer dynamically based on producers' sampling and elimination of per-Interest selection to increase the system goodput is needed.

4 RELATED WORK

Existing work can be divided into four categories: *i) wired server selection, ii) data distribution on edge, iii) base station selection for cellular networks, and iv) Interest Flooding Suppression.*

Wired server selection. Similar work has been done on selecting suitable servers in the cloud (e.g., anycast) [9–11] based on different requirement parameters (e.g., energy utilization, latency). Other recent work done by Song et al. [12] focuses on selecting the best data storage replicate with the least delay. These works differ from our work completely as ours is in the wireless environment. Wired networks do not have to deal with sharing medium nor can send Data once and have all peers receive it. Wired network loss rates are constant based on the physical link while wireless varies based on the environmental condition and the number of users which is uncontrollable and unstable. Our work deals with the wireless environment where dramatically varying losses and latencies are common.

Data distribution on edge. Data redundancy and distribution on edge is a common approach to provide low latency for applications such as AR/VR [2, 3]. There are algorithms to optimize the distribution of data among edge nodes to reduce latency and improve resilience of data access [4]. Such works do not handle selecting the optimal replica server but only care about the ideal data distribution. This complements our system, which is designed to obtain the data from the best among multiple producers who already have data distributed on them.

Base station selection for cellular networks. Some work uses SINR and other metrics to select the proper cellular base station for communication [13–15]. Such works are similar to producer selection (assuming a base station is a producer). However, our work is designed specifically for peer based wireless communication on edge, which has different ranges and characteristics from cellular networks.

Interest Flooding Suppression. Other works focus on interest packet suppression where Interest packets can flood networks, increasing content retrieval latency and lowering the performance. Existing works [16–18] deploy different strategies (distance based, latency, energy efficient, etc.) that work on multi-hop calculations to improve the system's performance. This differs from our system in two main ways: *i)* our goal is to suppress redundant Data packets and *ii)* select the best producer within a single hop wireless neighborhood.

5 DESIGN

5.1 Goals and Assumptions

Our goals can be summarized as follows:

- Select the optimal producer(s) at a higher granularity than an Interest packet. A Data packet can be as small as one frame or tens of frames. Selection per single Data packet has a high overhead, and selection at higher, semantically meaningful granularity is needed.
- Since there are no identifiers for individual nodes as routing and filtering is done based on Data frames, a mechanism that allows identifying producers and what Data they can provide with low overhead is necessary.
- Select the optimal producer(s) for one or multiple consumers with no coordination messages among the consumers. Explicit coordination messages among consumers are susceptible to loss and incur overhead in wireless environments. Thus a mechanism that allows multiple consumers to select the optimal producer without explicit coordination messages among consumers is needed.

Optimal Producer(s) Definition. Optimal producer(s) is the minimum number of producers transmitting the same Data (ideally one) to and meeting the performance needs of all consumers that requested the Data. Each consumer may have a different subset of qualifying producers, and these subsets may partially overlap.

Assumptions. We make the following assumptions to scope this work. *i)* The medium volatility [19] can cause frequent frame losses (e.g., a request is received by some but not all producers). *ii)* The size of the content the producer wants to send is reasonably large

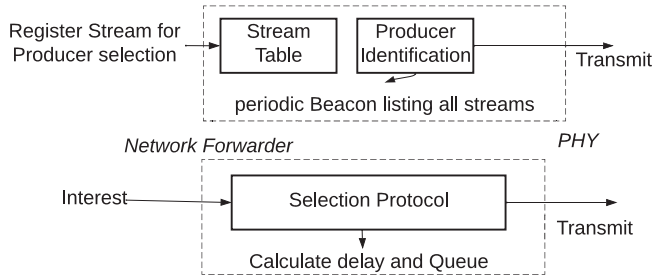


Figure 2: The *Stream Table* stores entries for all stream names a node can produce and their hash (stream ID). *Producer Identification* handles sending periodic announcements (beacons) of what streams a node can provide for nearby consumers. *Selection Protocol (Performance Table and Backoff Algorithm)* handles selecting the optimal producer(s) for all consumers interested in a stream.

(e.g., ∼ 1MB, large files or continuous streams). Such content can be segmented into Data packets at the network layer then Data frames at the link layer. *iii)* Due to link asymmetry, a producer's reception quality cannot be used to predict that of the consumer [20, 21]. *iv)* There exist naming conventions that allow applications to come up with globally or locally unique names, which is a common assumption in data-centric networks. Existing work has proposed designs and approaches for fast name lookup [1, 22]. *v)* The system is to run on commodity WiFi dongles (i.e., CSMA) to be widely adoptable. *vi)* We assume that all consumers can hear each other in the design because if the consumers cannot hear each other then the performance will be based on the overhead of redundancy. *vii)* There is no mobility among the nodes as most of our current applications need low to no mobility. Mobility may add requirements to the problem but will not change the design.

5.2 Cross-Stack Interface

In this section we explain the experience of NDN developers and application developers using producer selection for their Data. OPSEL is not needed where there are small amount of Data or no redundant copies.

When an application developer wants to use OPSEL for data possibly redundant within single hop, they need to specify a *stream name* that is unique and corresponds to the content (pipelined Data packets that are semantically meaningful to the application). *Stream name* is to be defined by the application and is producer node independent. A consumer requesting the Data (using the agreed upon *stream name* and *dataname*) can specify its *performance requirements* on thresholds of tolerable loss rate and latency, used in finding qualifying producers.

With application-level granularity *stream name* and *performance requirements*, OPSEL is able to select the optimal producer(s) for single or multiple concurrent consumers. Below we present an overview of our design components and describe their functionality in more detail afterwards.

Scenario	*Video Transmission*	*Audio*	*File Transfer*
Goodput	10 Mbps	1 Mbps	40 Mbps
Loss Rate	20%	2%	10%

Table 1: Stream Parameter Configuration Examples

5.3 Design Overview

Figure 2 shows an overview of our producer selection design. Our design consists of three main components: *Stream Table*, *Producer Identification*, and *Selection Protocol*.

Stream Table keeps a list of all the streams the node can be a producer for. *Producer Identification* announces through a beacon periodically what streams a node can provide. *Producer Identification* retains a list of all producers and what streams they can publish, created from the neighbors' beacons. *Selection Protocol* selects the optimal producer(s) for all consumers interested in the same stream. It uses an *Ordered Round-Robin* sampling strategy where it sends different Interests within a stream to different producers to measure their performance, and select the first qualifying producer possible.

Performance Table is used to store the consumers' stream requirements (loss rate and latency thresholds provided by application developers), and overheard information (e.g., whether a stream is already being transmitted; which producers are popular among existing consumers). Examples of stream requirements are shown in Table 1: if the stream is audio, low goodput is sufficient but it needs low loss rate, while video transmission can tolerate more loss but needs higher goodput. With file transfer, more loss than audio can be tolerated because lost data can be requested again, but much higher goodput is preferred.

A *Backoff Algorithm* allows consumers to cooperate without explicit coordination messages in selecting a common set of qualifying producers. Consumers with smaller subsets of qualifying producers announce their selections earlier, thus other consumers with larger subsets can reuse the selected ones that are common qualifying producers.

Below, we present our design for *Stream Table*, *Producer Identification*, and *Selection Protocol*.

5.4 Stream Table

We use a table to store a list of the Data streams for which the node is a producer. All such streams are announced periodically. We discuss the details of producer announcement in section 5.5. Every stream identifier is unique and known among all nodes for the same application (e.g., using naming conventions [23, 24]). When the application registers a stream, the MAC stores its stream ID.

Stream Table falls inline with Moiseenko's ICN flow classification [25] which introduces grouping Data packets for rate control and other network layer benefits. Stream Table continues on such approach and provides the needs and benefits of grouping Data packets in the MAC for producer selection. The stream ID must be unique to each Data stream and remains the same across nodes (i.e. nodes requesting the same Data must use the same stream ID). It must be unique to avoid mixing up Data. Unlike NFD [1], the stream name needs to be unique spatially (single hop) and temporally (duration of the transmission), not globally. Thus, a reasonable length

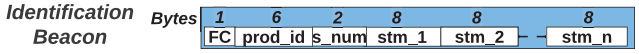


Figure 3: Frame structure of producer identification through beacon. The method allows each producer to aggregate multiple streams they can support in one frame and announce periodically.

(say 64-bits) can balance between the need for uniqueness and memory overhead. Each stream is generally tied to a globally unique label/name; such names can be hashed into IDs without conflict (e.g., by following naming conventions similar to NDN [23, 24]).

We give an example of how stream ID can be generated and used: A video file from a camera after being processed for a hall is replicated among edge nodes for accessibility. Such file can be of size 100MB or more for a few minutes at high quality. An example naming of the file can be *us/tx/austin .....ProcessingAB.mp4*, where the file describes the location, date of recording and how it was processed. Such a unique name can be used as the stream name and *datanames* can be generated by dividing the file into smaller reasonable lengths for Data packets and numbered sequentially.

5.5 Identification

A MAC-level stream identification method is necessary to know which producers nearby can fulfill which requests. We design *Identification Beacon*, a mechanism for consumers interested in a stream to identify and differentiate their nearby producers at the MAC level. Producers announce their supported streams in such beacons.

Data-centric paradigms do not necessarily have node identifiers. For identification purposes, some form of transient ID by hashing an input of globally or locally unique content (e.g., MAC address, *stream_id* augmented with some random seeds) can be used. Such a transient ID can expire after some time (e.g., say every 10 seconds) to avoid permanent identity association for better privacy if necessary. Such transient ID is used in Interest frames to indicate which producer should respond, and in V-MAC's data-oriented multicast acknowledgment protocol DACK [8] frames to prevent multiple active producers retransmitting the same Data. It is also used in every Data frame to indicate which producer sent which Data. Thus, only source ID is added in Data frames, and destination ID is added in Interest and DACK frames. We discuss the implications of having ID and how this differs from address-based communication further in section 7.

5.5.1 Proactive Beacon vs On-Demand Probe. Nodes periodically announce a list of streams they can provide to nearby consumers. Figure 3 shows the frame structure of an *Identification Beacon* where prod_id is any unique form of ID in data-centric communication (e.g., by unique ID generating techniques); *s_num* represents the number of streams listed in the frame, then the list of stream IDs it can support.

Each node that hears the beacon stores the producer ID to *Performance Table* entries of respective streams (or refreshes if already added). If multiple nodes announce the same stream, their producer IDs are appended to or refreshed in respective lists. Then, the *Selection Protocol* (discussed in section 5.6) decides which producer to request the Data from for a stream.

We have considered an on-demand discovery strategy where nearby producers are discovered right before sending an Interest. We found that its performance was very poor (discussed in section 6.3). Below we describe how we achieve multiple-consumer producer selection without explicit coordination messages leveraging *Producer Identification* and *Stream Table* mechanisms.

5.6 Selection Protocol

In this section, we discuss how OPSEL selects the optimal producer(s) across multiple consumers. We start through a simple scenario and build up to encompass more complex ones.

For one consumer selecting among multiple producers, it sends pipelined Interests of that stream to different producers to measure their performance, and then selects the optimal one that meets the *performance requirements*. If another consumer wants to join that ongoing stream, it can overhear the first consumer's selections, sample the chosen producer if it qualifies, and otherwise sample others. Consumers need to overhear medium transmissions and rank producers based on popularity (number of times selected) in *Performance Table* (details in *Order Round-Robin Sampling* below).

Another scenario is where multiple consumers are sampling producers concurrently and have to decide at similar times the optimal producer(s). OPSEL lets the consumer with the least number of choices announce its selection first. Other consumers will try to reuse announced selections if they qualify. We design a *Backoff Algorithm* to handle concurrent consumers actively selecting from the qualifying producer(s).

Regardless of how the selection happens (i.e. consumers competing or joining an ongoing selected producer), it can occur that the selected qualifying producer performance degrades and thus is not a qualifying producer anymore. The re-selection process starts again on the next Interest being sent by consumers where they start sampling different producers until another qualifying producer is selected.

5.6.1 Performance Table. We use a table that logs overheard active transmissions of Interests and stores their information (producer ID, stream ID, and encoding) to know which producers are being selected in the medium. It also stores performance (loss rate and latency) of producers selected previously by the consumer. Thus a consumer knows which producers are sending what and can be sampled first to see if they can meet its requirements.

In *Performance Table*, we prioritize those producers that are actively transmitting a stream or selected by more consumers to probe first. Such prioritization allows selected producers to be reused, thus minimizing the total number of selected producers for less transmission overhead.

The *Performance Table* information (counter per producer of number of times they were selected by consumers through Interests, the streams they were selected for, and loss rates and latency metrics for producers the consumer selected itself) is held for a certain duration and then it expires for the following reasons: *i)* a producer that was selected a lot some time ago (say 5 minutes) may no longer be a good choice due to varying wireless reception; *ii)* holding the

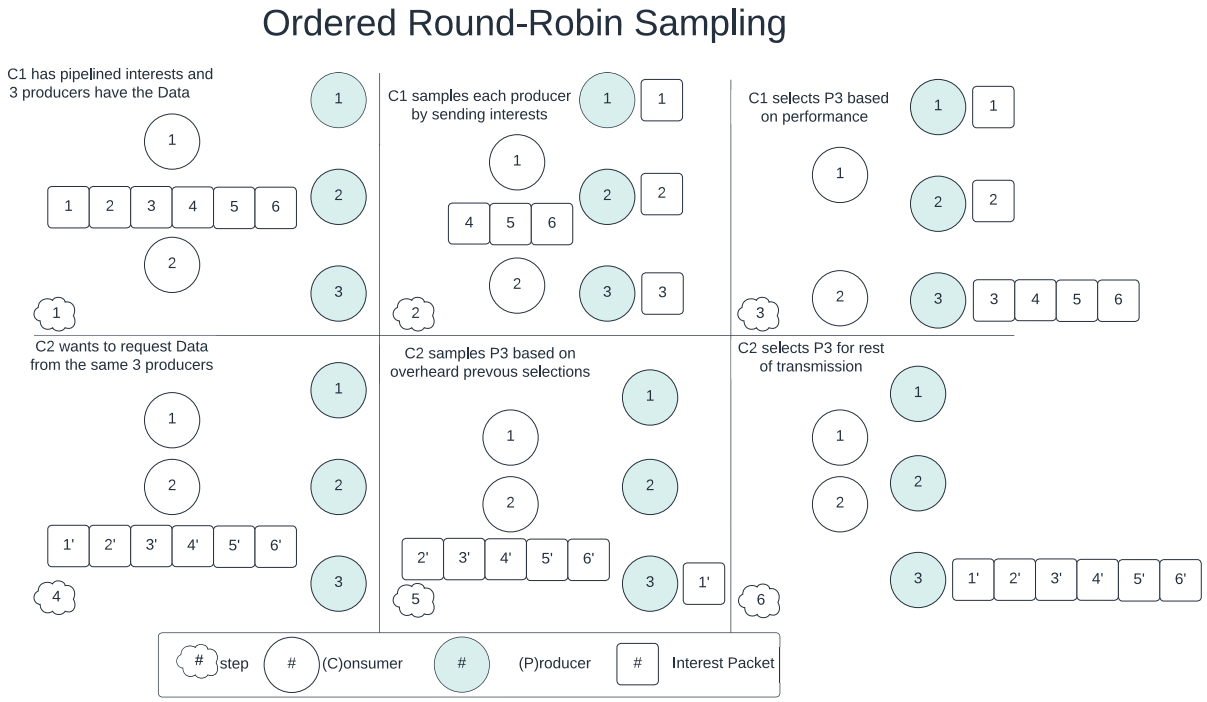


Figure 4: Illustration of 2 consumers each requesting data of 6 (possibly different) pipelined Interests within the same stream, for which 3 producers have the data. Step 2 shows a consumer sampling each producer by sending one Interest to each, waiting for Data frames to return (approximately 25 Data frames per Interest), and measuring respective performance. In step 3, the consumer selects Producer 3 as the optimal producer and sends the rest of its Interests to it accordingly. With the second consumer overhearing such interactions, it starts by sampling Producer 3 to see if it meets its stream's performance requirements. After verifying that is the case, it selects Producer 3 for its remaining Interests.

information for too long will require large sizes of memory and is unnecessary. Therefore, we set an empirical window time of one second, after which the information is removed.

Order Round-Robin Sampling. With *Performance Table*, a consumer overhears and orders nearby producers for a stream based on their active transmission and popularity (i.e. number of times heard being selected). If a producer nearby is actively transmitting a stream needed by a consumer, it gets ordered to the top. Popularity is used when there is a tie between more than one producer actively transmitting, the tie is broken by how many times they have been heard being selected (i.e. through Interests).

Through such overhearing and ordering, consumers are able to join an ongoing stream by selecting an active producer right from the first Interest and continues to use it; if it does not meet the required performance, the consumer then samples other producers to find a qualifying one. Figure 4 demonstrates the selection process where a consumer with pipelined requests sample different producers that have announced they have data for such stream through *Proactive Beacon* (Section 5.5). After sampling each producer, the consumer selects Producer 3 as its respective producer for the rest of the Data. When another consumer wants to request Data that is available at the producers, the consumer immediately samples the chosen producer by the first consumer, and continues using it after confirming its performance requirements are met. This shows how consumers can overhear and leverage each other's sampling results to identify optimal producers without coordination messages.

However, it is possible for multiple consumers to start sampling at the same time when no producer has yet been selected. We design a backoff algorithm that is agnostic to how many consumers are actively sampling; it allows for consumers to converge on optimal producer(s) without explicit messages among consumers when they are requesting the same Data (i.e. one stream).

We set the size of sampling per producer to be 25 Data frames empirically based on extensive experiments. Such number gives a balanced trade off between sampling producers and estimating a qualifying producer's performance. The performance of the producer is evaluated based on loss rates and latency. Loss rate is calculated by dividing number of lost frames over highest sequence number for the Data frame received. Latency is measured by calculating the duration from when the Interest was sent until the last Data frame belonging to that Interest was received. Experiments showing the justification behind the number of sampling Data frames per producer are shown in section 6.4.

5.6.2 Backoff Algorithm. We design a backoff algorithm that adds some delay before sending an Interest selecting a producer so that *Performance Table* can revise its selection if needed. For the backoff algorithm to work, a common trigger needs to happen among the consumers where they start their timers concurrently. Throughout the transmission, the end of previous Data packet and link layer receiving the next Interest packet is the common trigger of starting the calculation. At the very first time, each producer waits a short duration to see if any other producer is sending an Interest for the same Data to select their selected producer as well to sample, otherwise they proceed with their selection. Such mechanism ensures that all producers align in their backoff implicitly from the second Interest packet. Not doing a backoff algorithm can incur high medium overhead and Data performance degradation (high latency) due to redundant transmissions. Each consumer waits proportionally to the number of qualifying producers it has. Thus the more producers a consumer has, the longer it waits, and the larger chance it can reuse some earlier announced selections. The backoff time is:

$$T = \alpha \cdot \lambda \tag{1}$$

We use λ to denote the number of qualifying producers for the consumer. The more qualifying producers, the longer it waits. This ensures that consumers with fewer choices send their Interests first and announce their selections first, and those with more choices may reuse announced ones if they also qualify. We use α to represent the slot size at which multiple nodes should wait. The slot size is based on the medium contention (i.e. larger slot sizes for busier medium). It is determined based on the duration between receptions of two back to back Data frames of consecutive sequence numbers. The slot size enables the algorithm to adapt to different medium background traffic and understand how long it should assign per slot (e.g., wait 3ms in busy background and only 0.5ms in empty medium).

This equation does not break all ties (e.g., two consumers with the same number of choices). Upon such cases, CSMA in MAC helps reduce collisions by another small random backoff before transmitting. The main purpose of the equation is to maximize producer reuse for less transmission redundancy.

6 EVALUATION

6.1 Implementation

We build an implementation on top of V-MAC's source code. The main challenges are: *i)* the current Data and Interest frames sent from V-MAC do not indicate any node identity, however our design requires producer IDs to operate. We need to create such unique producer IDs; *ii)* proper way to announce available streams on each node. *iii)* cross-stack communication needs to pass information required for producer selection (e.g., *performance requirements*, stream ID, supported streams a node can be a producer for, etc.). Below we discuss how we resolve the above issues in the implementation.

For producer IDs, we create a transient ID by adding random values (8 bits) to the 48-bit MAC address. Such IDs can be regenerated periodically before the node announces what streams it can offer Data. (Further discussion on producer ID can be found in section 7.) We further modify V-MAC's frame header format by adding *prod_id* in Interest frames so that producers know to which ones they need to respond.

Upon receiving a Data frame with *prod_id*, an entry pairing the prod_id with its *stream_id* (obtained from the mapping from encoding to stream_id) is stored (or refreshed) in the *Streams Table* to record candidate producers for this application. We also add *prod_id* within DACK frames in case there are different producers transmitting the same Data for different consumers, thus preventing redundancy. To support producer identification, we add a new frame subheader under control frame type to create Beacon Identification frames. Regarding cross-stack communication, we add more to the communication Application Binary Interface (ABI) between V-MAC's kernel module and its userspace library to support passing of an Interest's stream ID, and *performance requirements*.

Data frame payload Size	**1500 bytes**
Interest frame payload size	**300 bytes**
Number of frames per Interest	**25 frames**
TX Power	**20 dbm**
Transmission Rate	**65Mbps**

Table 2: Experiment parameter configurations

6.2 Experiment Setup

We evaluate our producer selection design using nine nodes each consisting of a Raspberry Pi 4 and an Alfa AWUS036NHA WiFi dongle. Table 2 shows parameters used.

We first evaluate outdoors with no background transmission[2] There are fine-grained intricacies that impact the wireless medium. Other work, Aletheia [26] sheds light on what wireless behaviors exist; then indoors with high background transmission (40–50%) to test the system's robustness and resilience. We also vary the data rates during Data transfer to see how well OPSEL can still choose the best producer. We test consumers joining an ongoing transmission one at a time and multiple joining at the same time to test our protocol's robustness.

6.3 Stream and Identification

In this section we evaluate the mechanisms within *Stream Table* and *Producer Identification*.

6.3.1 Stream Table. We test OPSEL by adding over 5000 streams at the same time and find that searching for one stream within the table results has 0.1 μs latency, which is the same as lower numbers of streams. *Stream Table* leverages hashtable structure which eliminates high latencies and demonstrates its support of high numbers of entries.

In section 5.5 we discussed the proactive beacon method and an on-demand discovery method. We evaluate both mechanisms and discuss their tradeoffs in latency and robustness.

Proactive Beacon. Identification Beacon adds no latency overhead for the consumer request because it is sent proactively and independently from the request. However, periodic beacons add

[2]We used Wireshark to monitor the medium and find 0% medium transmission (wireless meters are needed to ensure no wireless interference).

to medium utilization overhead. We found that one producer announcing up to 70 streams using one frame caused no observable latency on consumer requests. We also observe that periodic beacon interval of 100ms is reasonable for both low overhead and fast rediscovery (300ms from stream announcement at worst and average at 100ms).

We have also tested the impact of Identification Beacon on medium and transmission performance as the number of producers who have streams to announce increases. We tested up to 10 producers that can hear each other and have over 60 different streams per node (i.e. each producer announces 70 different stream IDs). We do not foresee much more than 10 producers in a single hop environment announcing different content and needing Identification Beacon. We found no latency increase (nor loss) when nodes requested Data from the producers that was observable at the microsecond scale. This shows that the Identification Beacon can handle scalability without hindering the performance.

On Demand Discovery. We find that on demand discovery adds significant latency to every consumer request (200–400ms) per Interest: *i)* The discovery is only sent when a request from the upper layer application is received at the MAC, and it takes time for neighbors to hear and respond to the probe. This adds the latency of 200–400ms per Interest or frequently even if responses are cached for a duration providing intermittent performance (unlike *Proactive Beacon* which incurs the latency only once when the Data is announced and does not incur any overhead further); *ii)* The probe is sent at the much slower base data rate (6Mbps) to maximize its chance of reception by all neighbors. *iii)* The consumer has to wait a while to receive multiple responses before selecting. We also find on numerous occasions the discovery frame was not received by any candidate producer, thus no response came back.

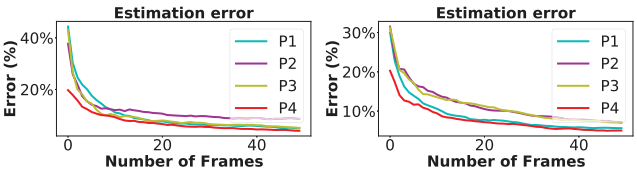
However, with periodic identification beacons, even if one beacon is lost, another will be sent shortly (within 100ms); each producer sends its beacons independently, increasing the chances of successful reception significantly. Thus we find that Identification Beacon is a suitable approach for producer and application (node and content) discovery, even in dense producer environments.

6.4 Effectiveness of Loss Prediction

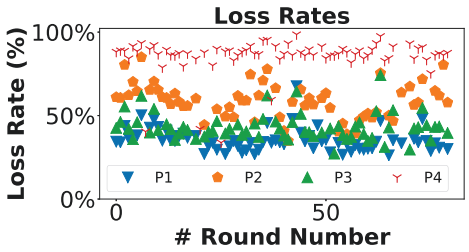
In this section, we set up experiments to answer a series of questions on the effectiveness of loss prediction through initial data frame probing: *i)* Can we use a small initial window of probing frames to achieve a good estimation of loss rate in a much larger subsequent window, and what is a proper probing window size? *ii)* How do varying data rates impact producers' performance? *iii)* How do different frame sizes impact loss rate and producers' performance?

Setup. We run experiments of broad ranges of loss rates and data rates to test OPSEL's robustness. Our setup is four producers and one consumer. To understand how OPSEL performs under different wireless environments, we vary the data rates at 6.5Mbps and 65Mbps[3] and loss rates between 35% and 100% among the 4 producers. We run these experiments indoors for over 5 days (producers send 500 Data frames every 10 seconds) where producers send Data and analyze the data at the consumer end. We repeated

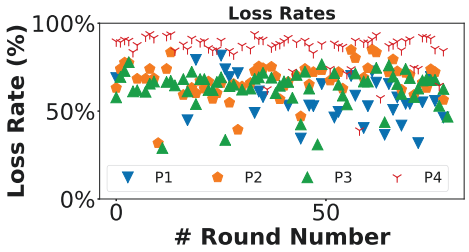
[3]We stop at 65Mbps due to hardware dongle limitations: the goodput does not increase with higher data rates.



(a) 6.5 Mbps loss estimation error (b) 65 Mbps loss estimation error



(c) 6.5 Mbps loss rates



(d) 65 Mbps Loss rates

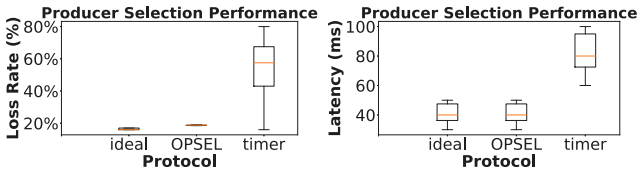
Figure 5: (a) and (b) show the estimation error in loss rate between a small initial probing window (up to 45 frames) and the whole round (2500 frames). (c) and (d) display individual rounds' loss rates from each producer (P1–P4).

the 5 day experiments in 4 different environments to reach our conclusion.

Proper probing window size. We analyze how close the loss in the initial probing window is to that of the whole round (i.e., 2500 frames). We define an error metric using the absolute difference between the loss rate of the probing frames and that of the round.

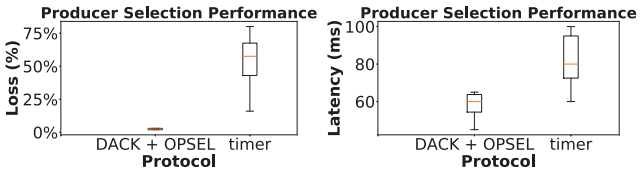
Figure 5a shows that with 6.5Mbps, a 10 frame window has a loss within 10% of that for the whole round, and this holds true for all producers. With 20 probing frames, the error drops down to around 5%, and further drop is marginal beyond 20 frames. Figure 5b shows 65Mbps prediction results and one can see that at 10 frames the error is 15% and at 20 frames it is 8%. The increase in error is because frames transmitted at higher data rates are less resilient and more prone to corruption due to background traffic and medium volatility. We decide that 25 probing frames provide a reasonable balance between accuracy and overhead for different data rates.

Optimal producer does not vary over different data rates. Figure 5c shows four producers' loss rates per round (P1–P4) at 6.5Mbps. We observe that for most of the rounds, the lowest loss producer alternates between P1 and P3, and both are good choices with maximum loss-rate difference at about 3%. Figure 5d shows 65Mbps results where such difference can be up to 15% between the top



(a) Loss Rates (b) Latency

Figure 6: (a) shows loss rates of our producer selection algorithm and long timer approach. (b) shows latency of our producer selection protocol and long timer approach. Both results (latency and loss rates) of our protocol are close to ideal, almost identical.



(a) Loss Rates with DACK (b) Latency with DACK

Figure 7: (a) loss rates and (b) shows latency. Both figures show performance of our producer selection algorithm with DACK and a long timer approach. Both results (latency and loss rates) of our protocol are better with our system showing low variability in loss rates and lower latency performance.

two choices. This enables OPSEL to select the correct producer (P3) which remains optimal throughout the transmission. This shows that regardless of the data rate used, the optimal producer remains relatively stable.

Various frame sizes. We try various frame sizes (100–1800 bytes) and find that we still select the optimal one under different frame sizes. We observe up to 15% loss as the frame size increases across all producers. This is because longer frames take more transmission time thus are more prone to collision with background traffic, leading to more losses. Since frame sizes affect all producers similarly, the chosen producer remains the optimal one.

6.4.1 RSSI is unreliable for Loss Prediction. We designed multiple experiments where 2 consumers receive data from 3 producers concurrently, and recorded received RSSI and loss rates for data frames over the whole transmission. We observed that RSSI values stayed consistent across hundreds of frames (varying by 1–2 dbm) and changed over hours and longer distances. We found that RSSI is weakly correlated to loss. For example, we had many consumers reporting strong RSSI values for a producer with high loss rates, and strong RSSI for another with low loss rates. This confirms previous observations [27] showing that RSSI is not a reliable indicator of loss rates.

6.5 Selection Algorithm Performance

We perform experiments outdoors in a location where there is no other background traffic to repeat the experiments and ensure validity. Then, we proceed to indoor experiments to validate the system's robustness.

Setup. Our setup is similar to section 3, with 4 producers and 4 consumers, each a Raspberry Pi connected to an Alfa AWUS036NHA dongle. We ensure there is no other background traffic happening in outdoor experiments and setup the 4 consumers near each other with producers distributed with the following loss rates: 15%, 30%, 60%, and 80%. We setup the system to test the algorithm where 3 consumers request the data concurrently and 1 consumer joins after 5 Interests and data packets are transmitted. We also compare our system's performance to a long timer approach which adds high latency, and to an ideal that always selects the optimal producer (the one with 15% loss rate). We set the data rate to 65Mbps. We define a stream to be of 10 data pipelined packets, each of size 37.5KB. Each data point represented in our evaluation was run at minimum of 10 times to ensure correctness and validity.

Producer Selection Performance. Figure 6a shows average loss rate performance of all consumers of all data they sent Interest for. OPSEL's loss rate is 18% while the ideal is 15% and long timer varies from 15–80% making it unreliable. This demonstrates that OPSEL was able to select the correct producer for multiple consumers concurrently. Figure 6b shows that our selection's protocol latency is 43μs while ideal is 40μs, meanwhile long timer latency is near 83μs (1.8x). This shows that OPSEL achieves near ideal loss rates (3% difference from ideal) without incurring any latency overhead.

Producer Selection with DACK. We test our producer selection performance with the DACK protocol where all consumers use DACK to request retransmission of missed frames from the selected producer. Figure 7a shows that our loss rates among all consumers drop below 5% while retaining lower latency than that of a long timer (shown in figure 7b). This demonstrates that with OPSEL and DACK (used as is from V-MAC [8]), our system meets the user needs and retains low latency and loss rates.

Changing Producers. We setup an experiment where the optimal producer is moved physically far from the consumers and thus the consumers have to find another producer. Once an Interest's data loss rates exceeds the *performance requirements*, the consumers started sampling other producers heard and selected the next producer that best met their requirements. The transition from one producer to another can cause a maximum loss of one Interest's data and no data loss, which occurs when the producer disconnects or the connection worsens. This demonstrates the system's capability to recover when the optimal producer changes when needed.

Consumers not having one ideal producer. We setup 3 consumers having the same optimal producer and 1 consumer far away where they cannot select the optimal producers of the 3 other consumers due to high loss rates above *performance requirements*. Our experiment performed as expected where the 4th consumer selects the 3-consumer optimal producer first, and after 1 Interest's data failing to meet the requirements, it sends the next Interest to another producer and ends up selecting that one as its optimal producer. At this point, we have 2 data transmissions for the same stream but that is what we need to satisfy all the consumers. This

demonstrates the system's flexibility in creating more than one stream to meet all consumers needed when necessary.

Indoor Results. We repeated all the experiments above indoor as well and obtained similar performance and robustness. While naive timer-based resulted in constant high loss rates 50–90%, OPSEL resulted in 30–50% loss rates through selecting the correct qualifying producer for the consumers given ongoing background traffic and medium condition. The results indoor varied based on background traffic (correlating with medium activity throughout the day). However, our indoor results are consistent with our outdoor results and show the resilience and system's robustness.

7 DISCUSSION

Limitations. We are aware that our design has several limitations: *i)* potentially high periodic beacon overheads when many nearby producers exist; *ii)* some efforts needed to set desired performance requirements for streams; *iii)* producer IDs annihilate the anonymity advantage in data centric networks; *iv)* performance parameter expiration can impact application performance. For *i)*, an adaptive beacon interval mechanism can be developed to loosely coordinate among producers to reduce overheads. For *ii)*, an analysis algorithm understanding the request content (e.g., video, audio, file transfer) can adjust the parameters without user intervention. As for *iii)* different random IDs can be generated over time so a producer cannot be tracked thus remaining anonymous. We leave the details to future investigation. Regarding *iv)*, the developer needs to follow some simple guidelines to set proper performance parameters (e.g., detailed in section 5.6), which we believe is quite easy to adopt.

Our design introduces multiple mechanisms (e.g., *Stream Table*, beacon and producer identification), which are essential for successful MAC layer producer selection in data-centric paradigms. E.g., *Stream Table* is needed to continue using a qualifying producer over multiple packets and avoid inefficient per-Interest selection.

The leading Consumer may select a non-optimal producer for all. It is possible that many consumers join an ongoing transmission where the first consumer has selected a non-optimal producer for most of these consumers. If the selected one does not qualify, these consumers will select their own optimal producer(s). The first consumer may stick to its selection, even though some later selections also qualify. A simple periodic sampling of popular producers can resolve the issue. The consumer sends its Interest to popular producer(s) to see if they qualify. We leave such evaluation for future work.

Learning-based Optimal Producer Selection. A learning based strategy is also possible. We decided not to pursue it for two reasons. First, the commodity hardware that is used limits the amount of useful information that could be exploited (e.g., frame loss feedback and RSSI). Second, we found that learning is unnecessary: our selection algorithm achieves nearly ideal results with only small overhead. In newer standards (802.11ax), learning might be able to take advantage of richer information such as Signal to Noise Ratio (SNR) sounding and Channel State Information (CSI) to guide the selection.

Small Streams. For applications with long durations of streams but small network layer packets, the producer selection strategy can operate without any issue. The first small packet (or a few) can be broken into a sufficient number of MAC frames for probing to measure producers' performance. However, for short streams (or small files), there may not be enough data for soliciting enough measurements from all producers. If short streams are frequent, the first few streams can be collectively used to ensure all producers are probed, thus ensuring optimal selection for subsequent streams.

High Mobility. OPSEL solves single-hop optimal producer selection for low-mobility and stationary environments. To solve that in highly-mobile environments, a speed-dependant adaptive parameter needs to be introduced to tune the *Proactive Beacon* frequency and performance parameter expiration time. Both are dependent on how often the network condition changes due to mobility, and thus a new set of producers need to be known and sampled before selecting.

Regarding missed Data when a consumer joins ongoing Stream in the middle. It can happen that a consumer wants a stream of Data that is already in the middle of transmission. A layer above MAC can observe what ongoing Interests are for the stream, and if other consumers are already in the middle of receiving that stream, it can join them for the ongoing transmission. That layer can then request the missed packets that were sent before its joining.

OPSEL needed only under one-hop redundancy. We are aware that Data may not be redundant over one-hop in wireless edge environments, thus OPSEL is not always needed. Such data can be referenced, transmitted and filtered using their *datanames*, and no stream name, stream table, or any producer identification information is necessary.

8 CONCLUSION

In this paper, we present a producer selection protocol for wireless edge environments where multiple nearby neighbors all have the desired data. The application specifies desired performance requirements and the selection protocol chooses suitable producer(s) efficiently, and quick reselection occurs once the performance degrades below thresholds. We compare different performance indicators and find ordered round-robin probing selection is an effective mechanism. We build a prototype with a completely new MAC layer using real WiFi (802.11/b/g/n) dongles for data-centric network stacks. Experiments show that OPSEL allows multiple consumers to converge selecting the optimal producer(s). OPSEL is 3% away in loss rate form the theoretical ideal and has the same latency as ideal, while naive timer methods can be up to 60% worse than ideal and incur 2-3× latency.

9 ACKNOWLEDGMENTS

We are grateful to our shepherd Prof. Alexander Afanasyev and the anonymous reviewers for their insightful comments and feedback, which have greatly improved the quality of this paper. In addition, this research was supported in part by the National Natural Science Foundation (Grants No.1730291, 1652276).

REFERENCES

[1] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[2] Charles C Byers. Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks. *IEEE Communications Magazine*, 55(8):14–20, 2017.

[3] Wei Wang, Yongli Zhao, Massimo Tornatore, Abhishek Gupta, Jie Zhang, and Biswanath Mukherjee. Virtual machine placement and workload assignment for mobile edge computing. In *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, pages 1–6. IEEE, 2017.

[4] Xiaoyu Xia, Feifei Chen, Qiang He, John C Grundy, Mohamed Abdelrazek, and Hai Jin. Cost-effective app data distribution in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):31–44, 2020.

[5] Zeinab Nezami, Kamran Zamanifar, Karim Djemame, and Evangelos Pournaras. Decentralized Edge-to-Cloud Load Balancing: Service Placement for the Internet of Things. *IEEE Access*, 9:64983–65000, 2021.

[6] Xiaoyu Xia, Feifei Chen, John Grundy, Mohamed Abdelrazek, Hai Jin, and Qiang He. Constrained app data caching over edge server graphs in edge computing environment. *IEEE Transactions on Services Computing*, 2021.

[7] Giulio Grassi, Davide Pesavento, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. VANET via named data networking. In *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 410–415. IEEE, 2014.

[8] Mohammed Elbadry, Fan Ye, Peter Milder, and Yuanyuan Yang. Pub/Sub in the Air: A Novel Data-centric Radio Supporting Robust Multicast in Edge Environments. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 257–270. IEEE, 2020.

[9] Patrick Wendell, Joe Wenjie Jiang, Michael J Freedman, and Jennifer Rexford. DONAR: decentralized server selection for cloud services. In *Proceedings of the ACM SIGCOMM 2010 conference*, pages 231–242, 2010.

[10] Hiroki Kataoka, Shigenari Nakamura, Dilawaer Duolikun, Tomoya Enokido, and Makoto Takizawa. Multi-level power consumption model and energy-aware server selection algorithm. *International Journal of Grid and Utility Computing*, 8(3):201–210, 2017.

[11] Hiroki Kataoka, Atsuhiro Sawada, Dilawaer Duolikun, Tomoya Enokido, and Makoto Takizawa. Energy-aware server selection algorithms in a scalable cluster. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 565–572. IEEE, 2016.

[12] Yaqin Song, Hong Ni, and Xiaoyong Zhu. An enhanced replica selection approach based on distance constraint in icn. *Electronics*, 10(4):490, 2021.

[13] Christodoulos Skouroumounis, Constantinos Psomas, and Ioannis Krikidis. Low-Complexity Base Station Selection Scheme in mmWave Cellular Networks. *IEEE Transactions on Communications*, 65(9):4049–4064, 2017.

[14] Jianpeng Ma, Shun Zhang, Hongyan Li, Nan Zhao, and Victor CM Leung. Base Station Selection for Massive MIMO Networks With Two-Stage Precoding. *IEEE Wireless Communications Letters*, 6(5):598–601, 2017.

[15] Shipra Kapoor, David Grace, and Tim Clarke. A base station selection scheme for handover in a mobility-aware ultra-dense small cell urban vehicular environment. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5. IEEE, 2017.

[16] Xiangshen Yu, Rodolfo WL Coutinho, Azzedine Boukerche, and Antonio AF Loureiro. A distance-based interest forwarding protocol for vehicular information-centric networks. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5. IEEE, 2017.

[17] Shuai Gao, Hongke Zhang, and Beichuan Zhang. Energy efficient interest forwarding in NDN-based wireless sensor networks. *Mobile Information Systems*, 2016, 2016.

[18] Menghan Li, Dan Pei, Xiaoping Zhang, Beichuan Zhang, and Ke Xu. Interest-suppression-based NDN live video broadcasting over wireless LAN. *Frontiers of Computer Science*, 11(4):675–687, 2017.

[19] Shafqat Ur Rehman, Thierry Turletti, and Walid Dabbous. Multicast video streaming over WiFi networks: Impact of multipath fading and interference. In *2011 IEEE Symposium on Computers and Communications (ISCC)*, pages 37–42. IEEE, 2011.

[20] Kyu-Han Kim and Kang G Shin. On accurate and asymmetry-aware measurement of link quality in wireless mesh networks. *IEEE/ACM Transactions on Networking*, 17(4):1172–1185, 2009.

[21] Seongwon Kim, Min Soo Sim, Chan-Byoung Chae, and Sunghyun Choi. Asymmetric simultaneous transmit and receive in WiFi networks. *IEEE Access*, 5:14079–14094, 2017.

[22] Chuwen Zhang, Yong Feng, Haoyu Song, Beichuan Zhang, Yi Wang, Ying Wan, Wenquan Xu, and Bin Liu. PBC: Effective Prefix Caching for Fast Name Lookups. In *2020 IFIP Networking Conference (Networking)*, pages 1–9. IEEE, 2020.

[23] Yingdi Yu, A Afanasyev, Z Zhu, and L Zhang. Ndn technical memo: Naming conventions. *NDN, NDN Memo, Technical Report NDN-0023*, 2014.

[24] Davide Pesavento, Giulio Grassi, Claudio E Palazzi, and Giovanni Pau. A naming scheme to represent geographic areas in NDN. In *2013 IFIP Wireless Days (WD)*, pages 1–3. IEEE, 2013.

[25] Ilya Moiseenko and David Oran. Flow classification in information centric networking. Technical report, Internet-Draft–work in progress 05, IETF, 2020.

[26] Mohammed Elbadry, Fan Ye, and Peter Milder. Aletheia: A lightweight tool for WiFi medium analysis on the edge. In *ICC 2021-IEEE International Conference on Communications*, pages 1–7. IEEE, 2021.

[27] Ana Bildea, Olivier Alphand, Franck Rousseau, and Andrzej Duda. Link quality metrics in large scale indoor wireless sensor networks. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1888–1892. IEEE, 2013.