SWICE: Towards Connection-Free Transmission in Wireless Distributed Edge Environment

Changkang Mo*, Yaodong Huang*†, Jiahui Luo*, Biying Kong*,
Hengzhi Wang*, Yu Liu‡, Fei Chen*, Laizhong Cui*
*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
‡Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China

***Abstract*—The rapid evolution of wireless edge computing faces fundamental limitations from connection-oriented protocols, particularly in dynamic scenarios with distributed edge environments. In this paper, we present SWICE, a novel transmission system with modified frame injection techniques that eliminates connection establishment overhead while ensuring reliable data delivery in wireless distributed edge environments. Our system introduces a connection-free measurement strategy that uses minimal packets to assess communication status while keeping device identities private. We formulate an edge node selection problem for efficiency and develop a heuristic algorithm for optimal data delivery. Additionally, we implement a reliability mechanism that combines adaptive retransmission to enhance communication stability. We conduct real-world experiments with commercially available Wi-Fi devices and modified wireless radios. The results show that SWICE achieves up to a 36.88% increase in goodput compared to conventional transmission methods, demonstrating its effectiveness through real-world experiments.**

I. Introduction

Wireless edge computing has experienced rapid growth in recent years, driven by the convergence of emerging technologies including the Internet of Things (IoT), advanced WLAN architectures (e.g., Wi-Fi 6/7), and industrial automation systems. In dynamic application scenarios such as smart transportation systems and mobile computing platforms, edge servers are often deployed in a distributed manner to provide broader coverage and more stable communication. This distributed approach enhances scalability, reliability, and efficiency, making it a key component for high-quality communication in such dynamic and wireless environments.

Fig. 1 depicts a wireless distributed edge computing scenario characterized by three core features: 1) inherently unstable wireless links between end devices and edge nodes, 2) frequent mobility of end devices (e.g., users entering/exiting the network coverage area), and 3) stable connectivity between pre-deployed edge nodes. Such architectures have become a fundamental part of next-generation applications requiring distributed intelligence.

However, this architecture introduces a significant challenge due to the limitations of traditional wireless communication protocols, which retain designs originally intended for wired networks at the network layer and above. While these adaptations are designed for protocol compatibility, their suitability for wireless transmission is compromised in such distributed communication environments. The omnidirectional nature of wireless transmission creates the potential for overhearing by all nodes in a signal coverage area, yet Wi-Fi and upper-layer protocols require proactive connection establishment before data transmission. This process introduces latency and overhead, particularly for applications requiring high-throughput and reliable transmission [1] in wireless distributed edge environments. Also, traditional connection-oriented methods rely on addresses across layers for data delivery and can reveal identities, which may introduce risks of malicious attacks [2].

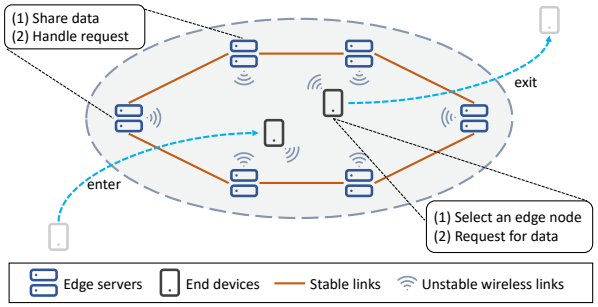


Fig. 1. A wireless distributed edge computing scenario with frequent mobility of end devices in a single edge domain

In such environments, adopting a connection-free transmission model and extending this concept to the underlying wireless communication layers emerges as a promising strategy. By eliminating explicit addressing and connection management overhead, this approach reduces latency while establishing communication frameworks optimized for transient network topologies. However, such architectural deconstruction necessitates new transmission system designs targeting dual objectives: reliable and high-throughput delivery. Such connection-free transmission focuses on data items themselves, offering a transparent and privacy-protected data delivery process to provide high Quality of Service (QoS). Based on these principles, we outline the following fundamental requirements for such a system.

First, ensuring both reliability and high-throughput data delivery in a connection-free system presents inherent complexities. Efficient data delivery requires dedicated data management linked to data identity. This requires a basic pattern for the data request and response mechanisms for data identifica-

†Corresponding author: Yaodong Huang (email: yd.huang@szu.edu.cn)

tion and transmission. Also, supplementary mechanisms (e.g., retransmission) are required to ensure transmission reliability.

Second, to support efficient data delivery in a distributed environment, the selection of the optimal edge node that is proximate to the end devices and has adequate capacity is vital. Achieving this requires dynamically making decisions on edge node selections based on information such as node communication status and capacities. Such processes are transparent to end devices without compromising the advantage of low-latency edge environments.

Third, the selection of edge nodes requires communication status information, necessitating an effective measurement mechanism. Measurements must account for data items, which requires frequent monitoring of network status for rapid communication adjustments. Also, as the identity of end nodes is hidden from the edge, measurement of data requests is required. Thus, edge nodes are responsible for measurement, with the ability to use minimal wireless probe packets between edge and end devices.

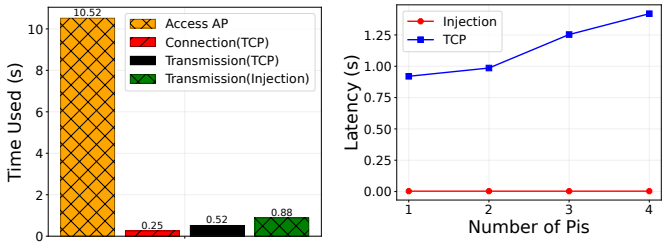
To address the challenges mentioned above, we design the Swift Wireless Infrastructure for Connection-free Edge (SWICE) transmission system. We build a wireless infrastructure testbed to offer connection-free transmission solutions in such distributed edge environments. To implement this design, from the bottom up, the system considers shifting the measurement to the edge node and requires only one probe packet for data delivery. The system selects the appropriate edge node for data delivery based on the measured information. Additionally, it provides reliable data transmission in a wireless network with connection-free features. We have implemented the system using real devices and tested it with a modified wireless radio supporting connection-free transmission. The radios can be easily deployed by embedding low-cost, commercially available Wi-Fi dongles. The results indicate that our proposed system can support transparent data delivery for end devices, having 36.88% higher goodput and 58.20% lower delay compared to traditional transmissions.

The main contributions of this paper are as follows.

- We design SWICE, a data transmission system optimized for distributed edge nodes in wireless environments. The system employs a wireless infrastructure with modified frame injection to achieve high-throughput transmission, along with dedicated reliability mechanisms to handle unstable connectivity scenarios. The system enables efficient connection-free communication between mobile end devices and the edge infrastructure.
- We propose a novel measurement strategy for estimating the communication status from end devices to edge nodes. This strategy enables the edge nodes to measure the connection status with a minimal number of packets. The end devices only respond to one packet for connection measurement without providing any identity information.
- We propose an algorithm to select the best edge nodes for data delivery. We formulate the node selection problem as a binary linear programming problem, which supports multiple data requests from multiple end devices simultaneously. We prove the problem to be NP-complete and propose a heuristic algorithm for edge node selection.
- We implement the proposed system on real devices and test them using commercially available wireless dongles. The results from extensive experiments over real devices show that our proposed system can achieve 36.88% higher goodput and 58.20% lower file transfer time compared to traditional transmissions, as well as up to 10.46% lower file transfer time compared to other baseline algorithms.

II. Motivation

In wireless edge computing, research and applications based on traditional Wi-Fi have become relatively mature. However, communication technologies based on Wi-Fi typically require two preliminary steps before actual data transmission can take place. The first step involves accessing the network (i.e., Access Point), and the second step requires establishing an application-layer connection with the target edge server. These steps are a result of the link-layer design of the Wi-Fi protocol and its compatibility with traditional wired protocols. Yet, in scenarios where end devices experience significant dynamic changes, these steps can introduce substantial latency and signaling overhead. We perform an initial comparison of the time used to transmit 5MB of data using native TCP and native frame injection (i.e., connection-free style). As illustrated in Fig. 2a, the traditional Wi-Fi approach shows a noticeable delay during the initial establishment phase when compared to frame injection. Note that frame injection does not incur connection delay, but the trade-off is that throughput may be reduced to some extent, resulting in slightly longer data transmission times. This is also the motivation behind our goal to design an efficient and reliable transmission system.



(a) Time used for data transmission (b) Latency of node selection

Fig. 2. Comparison of traditional TCP-based approaches and raw frame injection for: (a) overall data transmission time, highlighting the connection overhead of TCP, and (b) node selection latency

Furthermore, device mobility frequently alters the wireless network topology. To maintain optimal communication performance, it is essential to continuously monitor the network communication status between end devices and distributed edge servers. A common approach involves sending information to all edge nodes and evaluating communication quality through RTT [3], [4]. However, these measurements often require seconds, which is unacceptable in scenarios where end

devices are frequently moving and only have brief opportunities to transmit data. We conduct a preliminary comparison between the frame injection prototype and the traditional RTT method. As shown in Fig. 2b, the measurement time is reduced to milliseconds using raw frame injection, as we only need to broadcast a single query to obtain the result. The RTT method requires querying multiple edge servers, resulting in longer delays as the number of servers increases. Based on this, we need to design an efficient and accurate measurement system to quickly assess communication quality before data transmission in our system.

III. System Overview

In this section, we present the architecture of the proposed system, which aims to offer connection-free communication to wireless end devices in edge environments. The system is designed to offer transparent processes for end devices to obtain data, without requiring information about edge nodes. The components and their interactions are shown in Fig. 3.

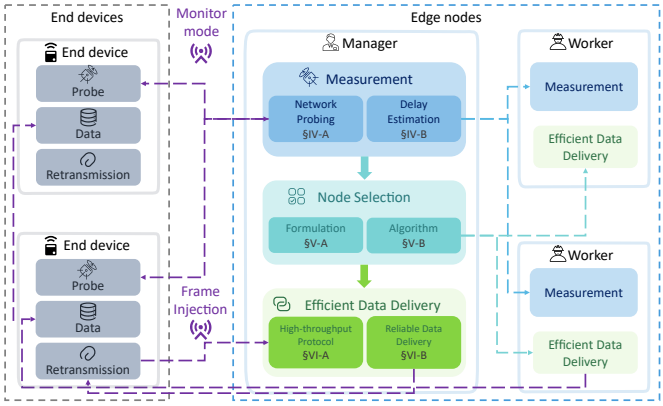


Fig. 3. The SWICE system architecture

A. End devices and Edge nodes

In the system, there are two major roles: end devices and edge nodes. End devices are the data consumers. They send requests to edge nodes and receive corresponding data. End devices primarily focus on obtaining data efficiently and are responsible for three functions: data requests, probe responses, and retransmission requests. A data request is the primary mechanism for obtaining the required data. A probe response is used by edge nodes to estimate communication status. Retransmission requests are used to retrieve lost data packets during transmission.

Edge nodes serve as data sellers or brokers, responsible for efficiently delivering data to the end devices. They are often established by certain organizations, referred to as edge providers, that are interested in offering fast local services to end devices. Edge nodes are further divided into two roles. One node serves as the manager, while the other nodes serve as workers. The choice of the first manager can be random or controlled by edge providers. The manager is responsible for organizing the edge nodes and collecting information from both end devices and workers for decision-making. The role of the manager can be switched to other edge nodes based on communication status. Workers mainly focus on the data delivery process, following the directives of the manager. Note that edge nodes within a given area are often part of the same organization, implying a higher level of trust among them.

B. Components

As shown in Fig. 3, three components in edge nodes correspond to the three main focuses of the system. The first is communication measurement, which estimates network quality, mainly related to delay, in wireless transmission scenarios. The manager node is responsible for conducting measurement processes and collecting information from end devices and workers. The measurement consists of two parts. Network probing is responsible for testing communication with end devices to obtain the network quality status. Delay estimation is responsible for collecting data from other workers and calculating the delay between end devices and edge nodes.

After estimating the delay among edge nodes and end devices, this information is sent to the node selection component. In this component, nodes that are the most suitable for data delivery are selected. We first formulate the problem as a binary linear programming problem. Then, based on the formulation, we propose an algorithm using the greedy strategy to make quick decisions based on the collected information. The manager then sends the decision to the corresponding edge node which then prepares data delivery.

The data delivery in our system mainly focuses on reliable transmission, ensuring that end devices can receive correct data in unstable, wireless situations. This part primarily focuses on mitigating data loss and its effects through retransmissions. From the perspective of end devices, they need to detect which data fragments are lost and request the missing data packets promptly. From the perspective of edge nodes, they need to identify the required lost data and efficiently send it back.

C. Communications

The communication between end devices and edge nodes is organized through wireless frame injection in monitor mode, using custom packet formats inspired by Information-centric Networking (ICN) naming principles. Each packet embeds a unique data name before its payload section, enabling receiver-side filtering without requiring address-based routing. The packets can be sent directly through wireless raw frame injection. In such cases, when requesting or responding, the end device does not need to care about the identity of the receiver of these messages. This has two main advantages. First, it helps to hide the identity from both parties, i.e., edge nodes do not know the identity of end devices and vice versa, yet the data delivery can still be fast and reliable. Second, with the nature of wireless frame injection, multicast plays an important role in the system, allowing various packets to be sent to multiple entities at the same time.

Note that in this communication model, since the edge nodes and end devices do not know each other, from the

perspective of edge nodes, they only serve requests based on data names, not on the addresses or identities of end devices. All behaviors of edge nodes are oriented towards data, not the devices. With the aforementioned design, end devices can obtain data efficiently without any information of edge nodes. Edge devices can also finish data delivery without knowing the identity of end devices, thus ensuring privacy for both parties in such scenarios.

IV. Communication Measurement Mechanism

In this part, we introduce the measurement strategy. We first introduce the process for conducting the network probing and discuss how the measurement is conducted among edge nodes and how the delay is calculated.

A. Network Probing

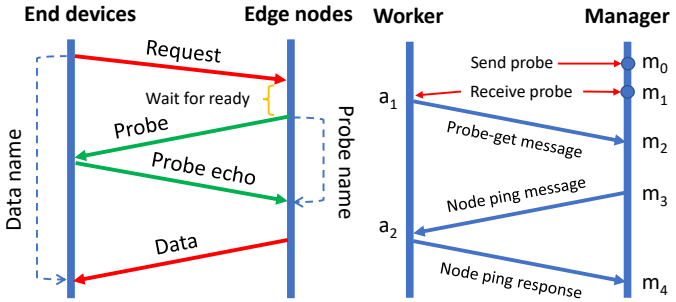
1) Request Packet: Fig. 4 (a) depicts the process for measurement and packet delivery. The end devices first send a request packet for the required data with the data name. We set a timeout feature for requests so that the request packets need to be retransmitted if no response is received. For the edge nodes, they maintain a request table that records the pending requests.

In such connection-free communication environments, neither edge nodes nor end devices initially possess surrounding information. From the perspective of end devices, they send request packets to edge nodes via frame injection, without knowing which particular node will receive the packet. From the perspective of edge nodes, they are aware that a request packet has arrived from an end device, but they do not know the identity of the originating end device, nor are they aware if other nearby edge nodes have also received the same request.

Consequently, after an edge node receives a data request, a brief waiting period is essential. This period serves two main purposes: 1) allowing time for the requested data to be ready (e.g., fetched from a sensor data source) and 2) coordinating with other nodes to aggregate the necessary information. During this waiting period, if other requests are received for the same data, the edge nodes will not immediately respond to them. Such a request might be a retransmitted packet or originate from other end devices. Either situation will be processed collectively in the subsequent step.

2) Probe Packet: Once the data is ready, the edge nodes start to send a probe packet. There are two main objectives for the probe packet. First, it detects the connection status of the network. Second, it ascertains the number of active requests. The probe packet can identify how many requests are still valid and waiting for data delivery.

The probe packet is sent by one of the edge nodes (i.e., manager). Since multiple workers, including the manager, might receive requests from an end device, when a worker receives a request, it notifies the initial manager. The initial manager then extracts the name of the request and records which node first informed it. The initial manager instructs the first node to send a probe packet and hands over the manager role to this node.



(a) Network probing and data delivery (b) Delay estimation

Fig. 4. Communication processes among different nodes in the two phases of measurement mechanism

3) Probe Response: End devices maintain a collection of data requests they have sent. After receiving a probe packet, end devices check if the name specified in the probe packet is in their set of requests. If it is, they return a probe response through an announcement packet, carrying a random number to inform the edge nodes that there is an end device here requesting data.

After receiving the probe response, edge nodes coordinate to select the most suitable node to send out data. The manager node is responsible for aggregating the information and making the selection decision. All nodes receiving the probe response packet will send information to the manager for further aggregation. Then, the manager determines the node responsible for the data delivery of each request.

4) Data: The node selected by the manager begins sending data to the end devices that are requesting. The data name should correspond to the name of the request packets. The data name is to differentiate different data items, and sending out data will not reveal the identity of the edge node. End devices receiving the data do not care which node is sending, and cannot infer the information about the senders.

Due to the characteristics of wireless networks, reliable data transmission is required in such scenarios. Thus, we also design a reliable transmission scheme that allows end devices to actively request missing data packets. These requests are treated as new requests and can be responded to by another edge node. We discuss the details in Section VI.

B. Delay Estimation

Now we discuss the measurement methodology for communications between different edge nodes and the end devices under a specific request. The main goal is to estimate the round trip time (RTT) between each edge node and a specific device using only one packet. We use the clock on the manager as the standard and calculate the relative timings among different edge nodes. Fig. 4(b) illustrates the delay estimation processes.

First, the new manager, selected to send out the probe packet to the end device, records the sending time as m_0. The end device then sends back the probe response as an announcement packet. Due to the broadcast feature of frame injection, most edge nodes within reach will receive it. Upon receiving this

packet, each edge node records the reception time as a_1, as denoted in Fig. 4(b). Upon receiving this response packet, every worker immediately sends a new message to inform the manager that it has received the probe response. The manager also may receive the response packet at time m_1. When the manager receives a new message from another node at time m_2, it sends out a ping packet to that node at time m_3. Then, the worker receives the ping at time a_2 and sends a response. The manager receives the response at m_4. At this point, the estimated RTT from this worker can be obtained.

We make three assumptions. First, the travel time among edge nodes is stable, as they typically use superior communication media, e.g., wired transmission. Second, the identity of an edge node is known to other edge nodes but not to end devices. Third, the processing time between a ping and a ping response is negligible. Thus, $m_2 - a_1 \approx a_2 - m_3 \approx m_4 - a_2 \approx \frac{1}{2}(m_4 - m_3)$. Then, the time difference of receiving the probe response is

$$m_1 - a_1 \approx \frac{1}{2}(m_4 - m_3) - (m_2 - m_1). \tag{1}$$

From (1), we can induce that

$$a_1 \approx m_2 - \frac{1}{2}(m_4 - m_3). \tag{2}$$

We can now calculate the RTTs between each edge node and end devices. For the manager node, the time is directly given by

$$u_m = m_1 - m_0.$$

For other workers, the RTT $a_1 - m_0$ using the clock of the manager from (2) is as

$$u_a \approx m_2 - \frac{1}{2}(m_4 - m_3) - m_0,$$

which can be used for the parameters in the next section.

Note that the estimation reflects the network connection quality. Accuracy in this estimation is important. Also, we believe our assumptions are reasonable based on certain real-world scenarios. First, edge nodes in one area are often deployed by the same company, and the transmission is much more stable than that between edge nodes and end devices, sometimes even using wired connections among nodes. This leads to much more stable communication, where transmission in two directions takes nearly the same time. Second, as mentioned above, knowing the identity of nodes from the same company should be acceptable, and for node selection, an entity (e.g., the manager) must possess the identification of the nodes. Third, the process time between receiving a ping and sending a response is very small compared to transmission. Even if the processing time needs to be considered, there is a similar processing time between the worker receiving the probe response and sending the first message to the manager. Thus, we neglect the processing time for a simpler estimation.

V. Formulation and Solution for Node Selection

In this section, we present the node selection problem of our system. The goal of selection is to identify the specific node to deliver data for a certain request.

A. Node Selection

Node selection is crucial for efficient data delivery. In wireless distributed edge environments, the data delivery process cannot rely on stable connections; thus, selecting a node that has better communication capabilities with end devices can greatly improve data delivery performance. Meanwhile, frame injection also provides some benefits over wireless data delivery. Due to the omnidirectional nature of injection, multiple requests for the same data from different end devices can be satisfied by sending the data only once. Thus, the problem formulation takes this into account, supporting simultaneous delivery to multiple end devices by having one designated edge node send the data only once.

B. Problem Formulation

We formulate the node selection problem as a binary linear programming problem, using parameters including the delay obtained from the measurement mechanism, the bandwidth capacity of the edge nodes and the requirements of data items.

The goal of the problem is to minimize the total delay for data access, denoted as the objective function (3). u_{rsd} denotes the accessing delay for request r to get data item d from edge node s. The delay is obtained from the measurement mechanism in the previous section. x_{rs} is the decision variable. If $x_{rs} = 1$, it means that request r will be served by s. p_{rd} encodes the prior knowledge of data requirement of request r. Specifically, $p_{rd} = 1$ indicates that request r explicitly demands data item d. y_{sd} indicates whether edge node s is designated as the provider for data item d. If $y_{sd} = 1$, it denotes that edge node s is this designated provider and will be the one to respond with data item d to relevant requests.

$$\min_{\mathbf{x}} \sum_r \sum_s \sum_d u_{rsd} x_{rs} p_{rd} \tag{3}$$

$$\text{s.t.} \quad x_{rs} p_{rd} \leqslant y_{sd}, \qquad (\forall r \in \mathcal{R}, s \in \mathcal{S}, d \in \mathcal{D}) \tag{4}$$

$$\sum_d y_{sd} q_d \leqslant b_s, \qquad (\forall s \in \mathcal{S}) \tag{5}$$

$$\sum_s y_{sd} = 1, \qquad (\forall d \in \mathcal{D}) \tag{6}$$

$$\sum_s x_{rs} = 1, \qquad (\forall r \in \mathcal{R}) \tag{7}$$

$$x_{rs}, y_{sd} \in \{0, 1\}. \qquad (\forall r \in \mathcal{R}, s \in \mathcal{S}, d \in \mathcal{D}) \tag{8}$$

Constraint (4) ensures that if request r for data item d is served by edge node s, then node s must be the designated provider for data item d. Constraint (5) represents the maximum available bandwidth for different edge nodes. This constraint is specifically designed to fit the scenarios of the wireless distributed edge environments. We denote q_d as the bandwidth requirement for data item d, and denote b_s as the bandwidth of edge node s. It takes into account the scenario where the same request from different end devices can be satisfied by a single data delivery, utilizing bandwidth for only one data item. Thus, the summation is only for

data item d, but not request r. Constraint (6) ensures single-source delivery in connection-free environments. For data item d, only one edge node is selected to respond, even when multiple devices send concurrent requests r for d. This design eliminates redundant transmissions while leveraging the nature of wireless channels, meaning all devices can overhear the unique response. Constraint (7) ensures the request must be satisfied. This constraint shows that any request needs to be satisfied by at least one node for a specific data item. Constraint (8) is the domain of the variables.

This problem is NP-complete. We provide a simple proof here. First, search for set $\mathcal{D}$ to obtain the minimum q_d, select one $s \in \mathcal{S}$, and let corresponding $y_{sd} = 1$ in (5) . For the same s, search for $\mathcal{R}$ to find $x_{rs} = 1$ that satisfies (4). This solution is a feasible solution. Thus, it takes polynomial time to validate a solution. Second, the problem can be reduced to the bin-packing problem [5]. From (4) and (5), we have $\sum_d x_{rs} p_{rd} q_d \leqslant \sum_d y_{sd} q_d \leqslant b_s$. Considering a special case with only one server, the inequality can then be transformed into $\sum_d x_r p_{rd} q_d \leqslant b y_d$, since in this case $y_d = 1$. Thus, this special case of our problem is equivalent to the original bin-packing problem itself, i.e., our problem is at least as computationally hard as the bin-packing problem. Given that we can validate a solution in polynomial time and our problem has at least the same complexity as the bin-packing problem, we conclude that the proposed problem is also NP-complete.

C. Edge Node Selection Algorithm

To solve the above problem, we propose an algorithm to quickly select the corresponding node for data delivery. The algorithm uses a greedy process to select the node that satisfies all the constraints with the minimum cost. Algorithm 1 shows the procedures.

Algorithm 1 Edge Node Selection Algorithm

Input: Request set $\mathcal{R}$, edge node set $\mathcal{S}$, delay u_{rsd}, bandwidth requirements $\{q_d\}$, node capacities $\{b_s\}$, prior knowledge of data requirement p_{rd}
Output: Request-node set map $< \mathcal{R}, \mathcal{S}^* >$, r sent out by s^*

1: **for** each requests $r \in \mathcal{R}$ **do**
2: $d \leftarrow$ data requested by r
3: **if** $\exists s' \in \mathcal{S}$ s.t. $y_{s'd} = 1$ **then**
4: $s^* \leftarrow s'$ ▷ Data item d already served by node s'
5: **continue**
6: **end if**
7: $\mathcal{S}' = \mathcal{S}$
8: **while** $\mathcal{S}' \neq \emptyset$ **do**
9: $s \leftarrow \arg\min_s u_{rsd}$ ▷ Select node with lowest cost
10: **if** $q_d \leq b_s$ **then** ▷ Node s has enough bandwidth
11: $y_{sd} \leftarrow 1$ ▷ Select node s to provide data item d
12: $b_s \leftarrow b_s - q_d$
13: $s^* \leftarrow s$
14: **break**
15: **else**
16: $u_{rsd} \leftarrow \infty$ ▷ Mark node s unavailable for request r
17: $\mathcal{S}' = \mathcal{S}' \backslash s$
18: **end if**
19: **end while**
20: **end for**

We now explain the algorithm in detail. The algorithm begins by checking whether data item d is already being delivered by any edge node (lines 3-6). If such a node s' exists (i.e., $y_{s'd} = 1$), we select s' as the output and do not consume additional bandwidth. When no prior allocation exists, the algorithm iteratively selects the edge node s with the lowest delay for request r (lines 7-9) and evaluates two scenarios. First, if the selected node s has sufficient bandwidth to accommodate requirement q_d of d, it is assigned as the provider by updating $y_{sd} = 1$, thereby committing both the resource allocation and delivery responsibility (lines 10-14). Second, if s lacks adequate bandwidth, its delay metric u_{rsd} is permanently invalidated by setting it to ∞ (line 15-17), which automatically excludes it from future iterations.

We now analyze the complexity of the algorithm. We denote the number of requests as $\mathbb{R}$ and the number of edge nodes is a constant. It is easy to see that the process must be done for all requests once (line 1). All other operations in the loop are only branches and single operations, which do not increase the complexity. Thus, the worst-case complexity can be obtained from line 7, which is no more than $\mathcal{O}(\mathbb{R})$.

VI. Efficient Data Delivery Mechanism

In this section, we present the design of efficient data delivery of the system. As outlined above, end devices initiate communication by measuring the surrounding link quality of edge nodes, subsequently selecting an optimal node for dedicated data transfer. We explain how the system achieves efficient communication, focusing on both high-throughput delivery through MAC-layer enhancement and reliable transmission mechanisms.

A. High-Throughput Protocol Design

The high-throughput mechanism employs two designs. At the MAC layer, we adjust the transmission rate of frame injection to align with 802.11ac TCP throughput. Also, we implement frame filtering based on the data name. This fundamental request-response architecture enables receivers to discard non-matching data frames at the MAC layer, significantly reducing processing overhead.

The datagram protocol features a streamlined structure comprising three elements: a fixed 16-byte header, variable-length data name (4-255 bytes), and application payload. The header contains essential control fields including segment length, optional checksum, packet type flags, name length indicator, and fragmentation parameters (identifier/offset). The payload has minimal limitations and can support several application layer protocols.

B. Reliable Data Delivery Mechanism

1) Retransmission Request: To control the granularity of data delivery, the data item is fragmented, and a unique offset is assigned to each fragment. When data items start to be sent, the fragments are encapsulated in packets and sent out sequentially. In connection-free situations (i.e., no ACK), the responsibility for handling lost packets falls on the end

devices requiring the data. This is because the end devices have knowledge about the reception of data packets so that they can decide how much and which packet to request again. Therefore, for reliable data transmission, the end devices need to know which data fragments they are currently missing, how to formulate a request for the lost information, and when to trigger a request for retransmission.

We design two different scenarios for the end devices to initiate retransmission requests. The first scenario is based on the packet loss rate. We track the received packets and fill the pending data item according to the packet offset. In this way, we can check the number of holes in the data whenever a packet is received. If this number reaches a certain threshold, we consider it appropriate to initiate a retransmission request. In our experiments, we test how many holes are suitable before making an aggregated retransmission request. The second scenario is based on the time interval between two received packets, which we call packet spacing time. As frame injection lacks RTT measurement, we estimate it by using the time interval between two received packets. Upon receiving a data packet, if the packet is not sequential, we start to wait for the missing packet based on packet spacing time (PST) multiplied by a ratio β ($\beta > 1$). We test β in our experiments. When the timer expires, we start to send the retransmission request packet.

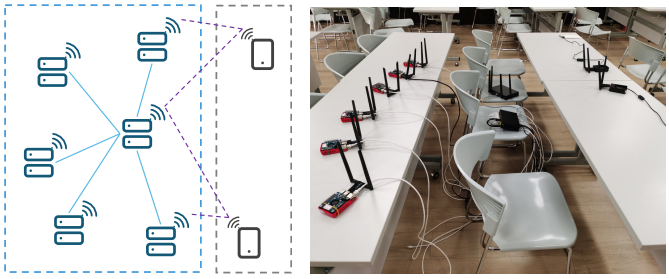
2) Retransmission Response: To achieve the goal of efficient transmission, we also need to simplify the retransmission handling of edge nodes. Specifically, to avoid request flooding and data redundancy in the wireless environment, the manager aggregates the information of lost packets before making a retransmission response. For the retransmission, each retransmission request is seen as a new request for new data, which means that a missing part of the data is also considered new data. Edge nodes still need to handle it like a normal data request, employing measurement mechanisms to ensure that only a single copy of the data is transmitted, rather than responding to every request. This design is entirely centered around data requests, eliminating the need for various identity information while maintaining efficiency.

VII. Evaluation

In this section, we present the performance of the proposed system, SWICE, using experiments conducted with real devices. We first test SWICE with the traditional protocol TCP for data transmission. We then test the efficiency of SWICE compared with certain baselines and further test the parameters that are best suited for the experiment environments.

A. Implementation

We implement key functionalities of SWICE, including measurement processes, node selection, system communication framework, and protocol compatibility with frame injection over the MAC layer. The system development is carried out using the Go programming language. To carry out the measurement process, we use the Google protobuf [6] library to efficiently serialize data. As for the frame injection, we refer to the V-MAC system [7] and modify the kernel driver. Specifically, we modify the transmission channel of network packets so that network data is directly transferred to our custom kernel driver after being received by the wireless dongle. Moreover, we fix the injection rate by setting the Modulation and Coding Scheme (MCS), channel bandwidth, and spatial streams. We design an intermediary protocol layer that connects our proposed system with the kernel driver. The primary function of the intermediary protocol is to maintain two concurrent read and write buffers: the read buffer is used to acquire data from the driver and waits for the upper layer system to retrieve it, while the write buffer queues the data to be sent to the driver sequentially. The communication of this protocol with the driver is carried out through the Netlink protocol [8] for reading and writing messages.



(a) Node topology (b) Experiment testbed

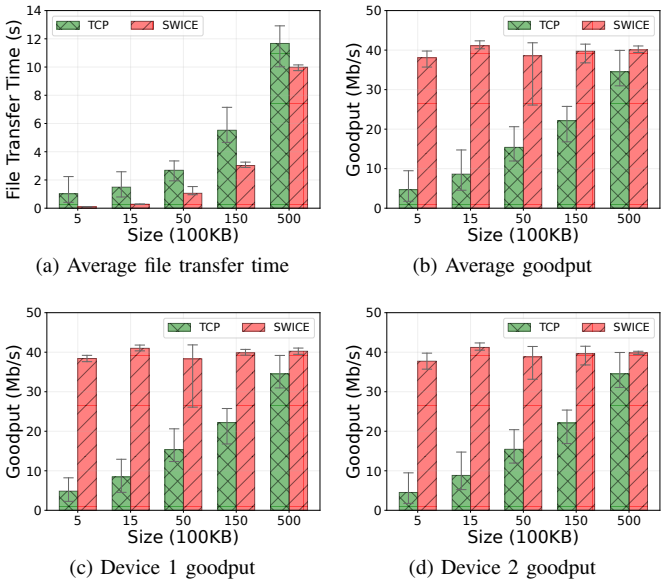
Fig. 5. Illustration on node topology and experiment testbed

B. Evaluation Settings

The devices we use are Raspberry Pi 4B and Raspberry Pi 5. We use the RTL8812AU USB Wi-Fi dongle, which supports the IEEE 802.11ac Wi-Fi standard for wireless transmission. The frequency bandwidth we use for the dongles is 40 MHz, and the MCS we use is 7. To simulate the scenario where server performance is superior to that of end devices in real-life applications, we deploy two end devices using Raspberry Pi 4B and six edge nodes using Raspberry Pi 5 shown in Fig. 5. We deploy our system on the Raspberry Pi OS with Linux kernel version 6.1. We test different sizes of chunks. Chunk sizes used in the experiments are 500 KB, 1.5 MB, 5 MB, 15 MB, and 50 MB. The reason we use chunks for testing is that they represent transient transmission moments in the dynamic operation of the proposed system, during which we make decisions on node selection and aim for the fastest possible transmission. We plan to conduct experiments with the complete dynamic system in our future work.

C. Comparison with Traditional Protocol

At first, we conduct a set of comparative experiments between our proposed SWICE and the commonly employed TCP protocol. This comparative analysis aims to highlight the efficiency of SWICE in contrast to TCP. The effectiveness of the system can be demonstrated by comparing the goodput and file transfer time.



(a) Average file transfer time (b) Average goodput

(c) Device 1 goodput (d) Device 2 goodput

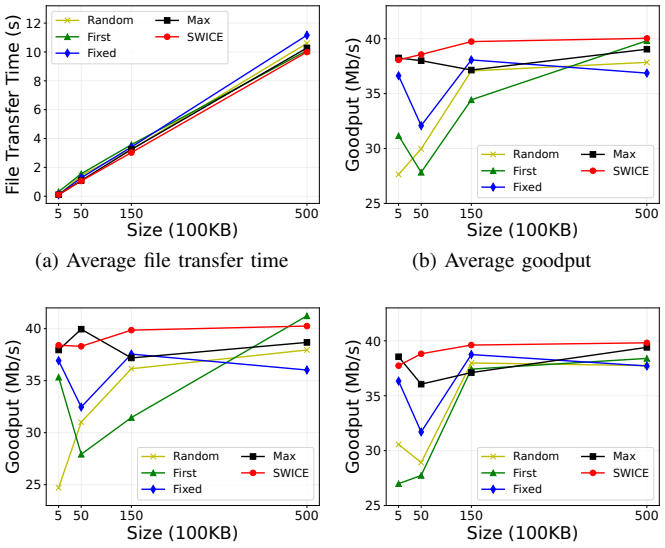
Fig. 6. Performance analysis of proposed system on average file transfer time and goodput compared to TCP, and performance analysis over different end devices

Fig. 6(a) shows the average file transfer time of SWICE and TCP with different chunk sizes for transmission, and Fig. 6(b) shows the goodput comparison between two systems while transferring chunks of diverse sizes. The overall file transfer time increases as the chunk sizes increase since a bigger chunk would take a longer transmission time. As for the goodput, it increases in TCP scenarios as the chunk size increases. This is typical for TCP as transferring larger chunks benefits more from longer connections. However, the goodput of SWICE remains stable for all chunk sizes, indicating SWICE can offer great performance for both large and small chunks.

Fig. 6(c) and Fig. 6(d) also confirm the viewpoint from the perspective of goodput. They respectively illustrate that we achieve a higher goodput for each of the end devices. This explains the TCP behavior, where transmitting data to multiple end devices or from multiple edge segments typically requires establishing multiple connections and transmitting multiple data copies. In contrast, SWICE utilizes the potential of injection for obtaining data, which allows SWICE to maintain a stable file transfer time in different devices. Our experiment results show that SWICE can reach a lower file transfer time, around 58.20% less than that of TCP on average while achieving a higher goodput of 36.88% compared with that of TCP on average. When the chunk size is set to 0.5 MB, the goodput of SWICE is 7 times higher than that of TCP. These jointly show that SWICE has a better performance in both file transfer time and goodput consistently.

D. Comparison with Different Algorithms

We then conduct a comprehensive comparison between the proposed algorithm and four baselines. “Random” algorithm selects an edge node as the sender randomly. “First” algorithm, chooses the sender that receives the data request first. “Fixed” algorithm uses a fixed edge node as the sender. “Max” algorithm selects the sender which is available and has the highest bandwidth.



(a) Average file transfer time (b) Average goodput

(c) Device 1 goodput (d) Device 2 goodput

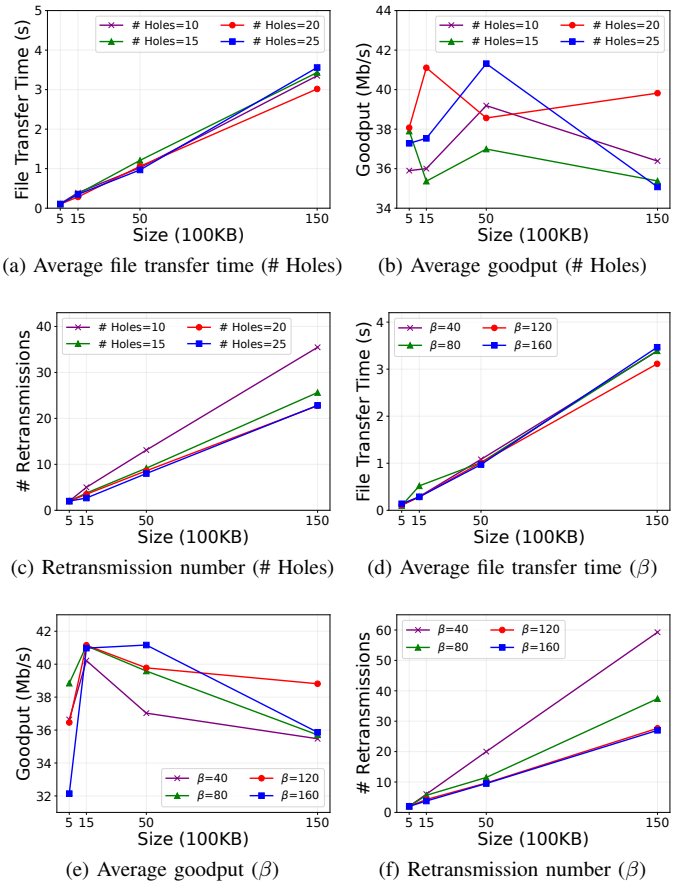
Fig. 7. Performance analysis of proposed algorithm compared with baseline algorithms, and performance analysis over different end devices

Fig. 7(a) illustrates the average file transfer time of different performances of algorithms while transmitting varying sizes of chunks. Overall, the file transfer time increases linearly. SWICE has better performance, resulting in the lowest file transfer time. This is because SWICE is designed to find the lower file transfer time between edge nodes and end devices. In all, SWICE outperforms Random, First, Fixed, and Max algorithms by 5.71%, 1.0%, 10.46%, and 2.77% in terms of file transfer time. Fig. 7(b) shows the average goodput of different algorithms while transmitting different sizes of chunks. The goodput fluctuates in a relatively smaller range when transmitting small chunks. However, as the chunk size increases, the goodput tends to be stable. In comparison to the other algorithms, SWICE outperforms Random, First, Fixed, and Max algorithms by 5.78%, 0.55%, 8.95%, and 2.54% respectively.

Fig. 7(c)(d) show the goodput of different performance of algorithms while transmitting various sizes of the chunk of two distinct end devices respectively. Fig. 7(c)(d) show that First algorithm exhibits superior performance when the chunk size is 500 KB, outperforming SWICE by 2.38%. However, this is because SWICE focuses on the average latencies from all nodes, aiming for overall optimality rather than the performance of a single node. In total, SWICE is demonstrated to be the most effective overall.

E. System Parameters Test

In this section, we evaluate how SWICE performs and functions by varying two important system parameters. The number of holes (denoted as # Holes in figures) parameter



(a) Average file transfer time (# Holes) (b) Average goodput (# Holes)

(c) Retransmission number (# Holes) (d) Average file transfer time (β)

(e) Average goodput (β) (f) Retransmission number (β)

Fig. 8. Parameter settings for the number of holes and waiting intervals ($\beta\times$ PST)

refers to the number of lost packets before triggering the retransmission mechanism. β refers how long as $\beta\times$PST before triggering the retransmission mechanism.

Specifically, the number of holes parameter is assessed at four distinct levels: 10, 15, 20, and 25. Fig. 8(a) shows the file transfer time with different numbers of holes for transmitting the different sizes of chunks. The overall trend in Fig. 8(a) increases linearly. We found that when the number of holes is set at 20 SWICE performs the best because the smaller value of the number of holes leads to more frequent retransmission which may cause extra cost of data transmission. However, a larger threshold leads to poor performance due to delayed retransmissions. Therefore, the setting of 20 is the proper value in our environment. Compared to the settings of 10, 15, and 25, the file transfer time is reduced by 9.97%, 12.12%, and 15.23%, respectively. Fig. 8(b) shows the goodput with different numbers of holes for transmitting the different sizes of chunks. The overall trend rises first and falls a bit while transmitting a larger chunk due to network stuttering. When the number of holes is set at 20, SWICE performs the best.

Fig. 8(c) shows the number of retransmissions with different number of holes settings for transmitting various sizes of chunks. In Fig. 8(a), the general trend grows linearly. When the number of holes is set at 20, SWICE performs the best. Compared to the number of holes as 10, 15, and 25, the performance is reduced by 35.85%, 11.22%, and 0.36% respectively.

Then, we explore the effects of β set at 40, 80, 120, and 160. Fig. 8(d) shows the file transfer time with different β settings for transmitting chunks of various sizes. We observed that the overall trend increases linearly when the β is set at 120, because the smaller value of β results in retransmitting data more often. But the performance gets worse while not retransmitting data in time. Therefore, the setting of 120 of β is more suitable for SWICE. Compared to β settings of 40, 80, and 160, the file transfer time is reduced by 8.0%, 8.14%, and 10.15%. Fig. 8(e) shows the goodput with different β settings for transmitting various sizes of chunks. The goodput rises at the start and is down a bit as the chunk size grows. While the β is set at 120, SWICE performs better overall. Fig. 8(f) shows the number of retransmissions with different β settings for transmitting different sizes of chunks. We observe that the overall trend increases linearly. SWICE performs better with the β setting at 160 and 120. Comprehensively the setting of 120 is more reasonable. Because a higher setting of β may not lead to retransmitting promptly. Compared to β settings of 40 and 80, the number of retransmissions is reduced by 53.31% and 26.02%.

VIII. Related Work

Many existing studies optimize wireless edge performance through computational task orchestration [9]–[12], inter-domain optimization [13], [14], and distributed learning acceleration [15], yet relies on TCP or other connection-oriented protocols. In contrast, our work fundamentally rethinks the communication substrate by eliminating connection-oriented protocol overhead, enabling high-efficiency data delivery that complements these upper-layer optimizations.

Prior works have explored the use of frame injection [16]–[18], primarily leveraging the fundamental rate of raw packet injection capabilities [19]–[21]. Our work systematically exploits the connection-free characteristic of frame injection to design a complete transmission system including network measurement and node selection, trying to exceed the traditional rate constraints.

As for the network measurement, most existing work requires the user to know the identities of edge nodes for node selection and data delivery [22]–[25]. Elbadry *et al.* [26] propose OPSEL, a protocol designed to select the most efficient data source in wireless edge environments. Related to our designs, there are also some traditional methods for RTT measurements [27]. Carra *et al.* [28] introduce an algorithm for passive RTT estimation using one-way traffic. Our system introduces a connection-free measurement strategy that eliminates identity disclosure requirements and achieves efficient node selection with minimal probing packets.

IX. Conclusion

In this paper, we have presented a new system that brings the connection-free model to wireless distributed edge computing, enabling users to access data in edge scenarios without requiring knowledge of the identity, capacity, or location information of edge nodes. We have introduced a novel measurement approach to assess communication quality without revealing the identities of both end devices and edge nodes. We have also formulated the data access problem based on delay considerations and proposed an algorithm that selects the edge nodes for data delivery. Furthermore, we have developed a reliable wireless transmission for data access using frame injection in monitor mode without maintaining connections. We have implemented this system on real hardware with actual wireless connections. Results from extensive experiments have shown that it supports efficient and transparent wireless communication suitable for dynamic scenarios with brief interaction times.

Acknowledgements

This work has been supported in part by the National Natural Science Foundation of China under Grant No. 62202309, No. U23B2026, No. 62372305, No. 62272318, and No. 62302314; Guangdong Basic and Applied Basic Research Foundation under Grant No. 2024B1515040012; Shenzhen Science and Technology Program under Grant No. RCBS20231211090523043 and RCYX20200714114645048O; and Research Team Cultivation Program of Shenzhen University under Grant No. 2023QNT015.

References

[1] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[2] M. B. Kamel, P. Ligeti, A. Nagy, and C. Reich, "Distributed address table (dat): A decentralized model for end-to-end communication in iot," *Peer-to-peer networking and applications*, pp. 1–16, 2022.

[3] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. ACM, 2003, p. 134–146.

[4] R. Ye, A. Boukerche, H. Wang, X. Zhou, and B. Yan, "E^3tx: an energy-efficient expected transmission count routing decision strategy for wireless sensor networks," *Wireless Networks*, vol. 24, no. 7, pp. 2483–2496, Oct 2018.

[5] M. R. Garey and D. S. Johnson, "Approximation algorithms for bin packing problems: A survey," in *Analysis and design of algorithms in combinatorial optimization*. Springer, 1981, pp. 147–172.

[6] J. Feng and J. Li, "Google protocol buffers research and application in online game," in *IEEE conference anthology*. IEEE, 2013, pp. 1–4.

[7] M. Elbadry, F. Ye, P. Milder, and Y. 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)*. IEEE, 2020, pp. 257–270.

[8] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "Rfc3549: Linux netlink as an ip services protocol," 2003.

[9] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient parallel split learning over resource-constrained wireless edge networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9224–9239, 2024.

[10] N. H. Motlagh, L. Lovén, J. Cao, X. Liu, P. Nurmi, S. Dustdar, S. Tarkoma, and X. Su, "Edge computing: The computing infrastructure for the smart megacities of the future," *Computer*, vol. 55, no. 12, pp. 54–64, 2022.

[11] D. Zeng, G. Min, Q. He, and S. Guo, "Convergence of edge computing and next generation networking," *Peer-to-Peer Networking and Applications*, vol. 14, no. 6, pp. 3891–3894, 2021.

[12] Y. Qin, D. Guo, L. Luo, and M. Xu, "A joint orchestration of security and functionality services at network edge," *Computer Networks*, vol. 212, p. 108951, 2022.

[13] Y. Wang, H. Liu, H. Song, S. Huang, S. Qin, Y. Liu, Q. Xiang, L. Kong, G. Li, J. Shu *et al.*, "Peering the edge: Enabling low-latency interdomain edge communication via collaborative transmission," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–6.

[14] C. Cicconetti, E. Carlini, and A. Paradell, "Edgeless project: On the road to serverless edge ai," in *Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge*, 2023, pp. 41–43.

[15] Z. Lin, G. Zhu, Y. Deng, X. Chen, Y. Gao, K. Huang, and Y. Fang, "Efficient parallel split learning over resource-constrained wireless edge networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 10, pp. 9224–9239, 2024.

[16] H. Yin, M. Ramanujam, J. Schaefer, S. Adermann, S. Narlanka, P. Lea, R. Netravali, and K. Chintalapudi, "{ADR-X}:{ANN-Assisted} wireless link rate adaptation for {Compute-Constrained} embedded gaming devices," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 1331–1349.

[17] L. You, S. Liu, W. Xie, Z. Wang, Y. Tan, and S. C. Liew, "Improving cooperative wi-fi broadcast with fine-grained channel estimation," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, 2024, pp. 1–10.

[18] X. Gu, W. Wu, Y. Zhou, A. Song, M. Yang, Z. Ling, and J. Luo, "Cqp-rffi: Injecting a communication-quality preserving rf fingerprint for wi-fi device identification," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[19] K. Qian, C. Wu, Z. Yang, Y. Liu, and K. Jamieson, "Widar: Decimeter-level passive tracking via velocity monitoring with commodity wi-fi," in *Proceedings of the 18th ACM international symposium on mobile ad hoc networking and computing*, 2017, pp. 1–10.

[20] M. Vanhoef, X. Jiao, W. Liu, and I. Moerman, "Testing and improving the correctness of wi-fi frame injection," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 287–292.

[21] S. M. Günther, M. Leclaire, J. Michaelis, and G. Carle, "Analysis of injection capabilities and media access of ieee 802.11 hardware in monitor mode," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–9.

[22] F. Wu, W. Yang, M. Sun, J. Ren, and F. Lyu, "Multi-path selection and congestion control for ndn: An online learning approach," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1977–1989, 2020.

[23] A. B. Rahman, P. Charatsaris, M. S. Siraj, and E. E. Tsiropoulou, "Symbiotic content caching in next-generation information-centric networking," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2023, pp. 414–421.

[24] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu, "A transport protocol for content-centric networking with explicit congestion control," in *2014 23rd international conference on computer communication and networks (ICCCN)*. IEEE, 2014, pp. 1–8.

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

[26] M. Elbadry, F. Ye, and P. Milder, "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.

[27] T. Li, K. Zheng, K. Xu, R. A. Jadhav, T. Xiong, K. Winstein, and K. Tan, "Tack: Improving wireless transport performance by taming acknowledgments," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 15–30.

[28] D. Carra, K. Avrachenkov, S. Alouf, A. Blanc, P. Nain, and G. Post, "Passive online rtt estimation for flow-aware routers using one-way traffic," in *NETWORKING 2010: 9th International IFIP TC 6 Networking Conference, Chennai, India, May 11-15, 2010. Proceedings 9*. Springer, 2010, pp. 109–121.