electronics

MDPI

Article

Use of QUIC for Mobile-Oriented Future Internet (Q-MOFI)

Muhammad Tauqeer [1,2], Moneeb Gohar [3,*], Seok Joo Koh [4] and Hani Alquhayz [5,*]

[1] Department of Computer Science, University of Swabi, Swabi 23430, Pakistan; tauqeer@uoswabi.edu.pk
[2] Department of Computer Science, Bahria University, E-8 Campus, Islamabad 44000, Pakistan
[3] Department of Computer Science, Bahria University, H-11 Campus, Islamabad 44000, Pakistan
[4] School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, Republic of Korea; sjkoh@knu.ac.kr
[5] Department of Computer Science and Information, College of Science in Zulfi, Majmaah University, Al-Majmaah 11952, Saudi Arabia
* Correspondence: mgohar.buic@bahria.edu.pk (M.G.); h.alquhayz@mu.edu.sa (H.A.)

Abstract: With the proliferation of mobile devices and various mobile services, ensuring smooth mobility for users has become a major challenge. The future internet is expected to be more mobile-friendly, with advancing technologies that will transform internet management in the coming decades. These technological advancements will help address mobility issues and provide a better internet experience for mobile devices and users. The transport layer plays a crucial role in the internet protocol suite by enabling communication between applications running on different servers. However, the widely used protocols, TCP and UDP, have several limitations, such as unreliability and slow performance due to three-way handshakes. To tackle these issues, Google introduced quick UDP internet connections (QUIC). QUIC aims to enhance latency, delay, and data transmission reliability. Q-MOFI, a future internet architecture focused on mobile devices and based on QUIC, strives to achieve these goals. Moreover, it enhances throughput by implementing multiplexing. Q-MOFI outperforms traditional UDP-based MOFI in terms of throughput gains, minimizing packet loss, and reducing binding operation latency, even when the number of hosts increases. The efficiency of this model has been validated through experimental testing.

Keywords: UDP; QUIC; MOFI; q-MOFI; LMC

Citation: Tauqeer, M.; Gohar, M.; Koh, S.J.; Alquhayz, H. Use of QUIC for Mobile-Oriented Future Internet (Q-MOFI). *Electronics* **2024**, *13*, 431. https://doi.org/10.3390/electronics13020431

Academic Editors: Jain-Shing Liu and Chunhung Richard Lin

Received: 11 December 2023
Revised: 11 January 2024
Accepted: 12 January 2024
Published: 19 January 2024

1. Introduction

With the increasing use of smartphones, ensuring seamless mobility for a diverse range of mobile devices and users has become a critical challenge for the future internet. The future of the internet is projected to embrace a mobile-oriented approach, signifying the intention to tailor the platform to accommodate the distinct requirements of mobile users. Advancements in technology are expected to transform internet management in the coming decades. These advancements will help address mobility issues and provide a more mobile-friendly internet architecture that is better suited to the needs of mobile devices and users [1]. Devices such as laptops and smartphones are mobile, requiring particular mobility management protocols to maintain IP mobility. Such protocols ensure that mobile device users have continuous access to mobile services while traveling between networks and staying connected [2]. The rapid growth of mobile devices, combined with the poor performance of standard networking protocols in high-mobility networks, has motivated researchers to create new protocols. The future internet architecture has mobility and trustworthiness as core design goals [3]. The future of mobile technology is expected to see a significant increase in the importance of identifier–locator (ID-LOC) mapping control and location monitoring, making the mobile environment a crucial factor in designing the future internet.

The transport layer, located between the application and network layers, is a significant part of the internet protocol suite, providing end-to-end communication services to

applications on different hosts. Its main functions include reliable data transfer, flow control, and error detection [4–6]. Application data are divided into smaller pieces (referred to as segments) by the transport layer, which is also in charge of reassembling the segments at the receiving end [7]. This layer provides flow control to prevent sender overload, and error control to ensure accurate and complete data reception [8,9]. TCP and UDP are widely used transport layer protocols, while quick UDP internet connections (QUIC), a recent protocol with the potential to enhance internet communication speed and security, has gained significant attention in recent years [10,11]. Furthermore, the transport layer provides essential services to apps, such as port numbers, which differentiate between different host apps and enable simultaneous operation of multiple applications on a single host, preventing data (mis)transmission [12,13]. The transmission control protocol (TCP) is a reliable transport protocol that enables end-to-end data packet delivery between applications. TCP employs several methods, including sequence numbering, acknowledgment, and flow control, to ensure proper packet delivery. TCP is widely used in various applications, such as file transfers, email, and web browsing [14–16]. TCP is a connection-oriented protocol that operates at the transport layer of the OSI model, ensuring reliable and efficient data transmission between two endpoints through methods like sequence numbering and acknowledgments [17,18]. TCP segments are assigned sequence numbers and received by the receiver, ensuring data delivery in the correct order and allowing the retransmission of missing segments. Congestion control is another crucial TCP mechanism, controlling data transmission based on network resources. TCP reduces transfer speed during busy networks to prevent congestion collapse [19–22]. TCP also offers flow control, which enables the receiver to regulate the pace of data transmission, and error detection and correction, which allow errors to be detected and corrected during transmission.

UDP is a transport protocol that facilitates connectionless communication between two endpoints, offering basic error detection and correction but lacking congestion control, flow control, or reliable data transmission. It is commonly used in low-latency applications like real-time video streaming, online gaming, and VoIP [23,24]. In the context of low-latency applications, the loss of certain data packets is deemed acceptable as long as it does not result in a substantial decline in overall performance. UDP disintegrates data into smaller units, known as datagrams, and transmits them to the intended device without the need for an initial connection establishment. The receiving device subsequently reassembles the datagrams into the original message and forwards it to the application layer [25,26]. Unlike TCP, UDP does not perform error checking or retransmission of lost packets, which can lead to packet losses and lost data. However, this drawback enables faster communication and lower network overhead when compared to TCP [27].

QUIC, a Google-developed transport protocol, aims to provide low-latency, secure, and reliable internet connections, improving network performance and overcoming TCP limitations [28,29]. The Cisco Visual Networking Index predicts a 23% CAGR in global IP traffic from 2021 to 2026, reaching 2.3 zettabytes per year by 2026. 'Video' will remain the dominant application category, accounting for 87% of global IP traffic by 2026. HTTP/3, based on QUIC, is predicted to become the dominant web content delivery protocol in the future [30]. QUIC (quick UDP internet connections) uses UDP (user datagram protocol) and implements its own congestion control [31] and error correction mechanisms, as opposed to traditional TCP, which relies on a three-way handshake and a reliable, ordered stream of data. This enables QUIC to reduce connection latency and recover from lost packets more efficiently, resulting in faster and more reliable data transfers [32,33].

QUIC establishes a connection between a client and server through a handshake process, ensuring quick and efficient data transfer and a better user experience. During this process, the client and server exchange information about encryption capabilities, session keys, and other secure communication parameters [34]. The client typically sends an initial packet to the server, which contains a cryptographic nonce used to prevent replay attacks [35,36]. The server responds with its own cryptographic nonce and other connection-related parameters. Once the handshake is complete, data can be transmitted between the

client and server using the stream-based protocol of QUIC. Data are divided into streams that can be transmitted and received independently of one another, allowing for more efficient use of network resources and better handling of lost or delayed packets [37–39].

QUIC is a popular choice for modern web applications due to its support for multiplexing and multi-pathing, enabling the transmission and reception of multiple data streams over a single connection [40]. This means that a web page can load resources like images and scripts in parallel, which reduces overall load time [41]. Multi-path QUIC allows a single connection to use multiple network paths simultaneously for data transmission, improving reliability and performance when a single path is congested or unavailable. The protocol dynamically selects the best path for each packet based on network conditions [42]. Second, QUIC supports connection migration, which means that if a client's IP address changes, it can keep its existing connection instead of establishing a new one [43,44]. Third, QUIC employs encryption by default, ensuring the security of all data transmitted over the connection [45].

The comparison between UDP and QUIC protocols presented in Table 1 highlights their differences. QUIC is a connection-oriented protocol that uses a handshake to establish a connection and provides reliable data transfer, while UDP does not. It supports multi-streaming and connection migration, has congestion control and flow control algorithms, and solves head-on-line blocking problems. UDP is unreliable, with no guarantee of delivery or ordering, while QUIC provides in-order delivery and congestion control. It also has built-in encryption and security features, unlike UDP's lack of these features. In summary, QUIC offers better reliability, in-order delivery, and security features compared to UDP.

Table 1. Feature comparison between UDP and QUIC.

Features	UDP	QUIC
Connection Setup	No	Yes
Reliability	No	Yes
Retransmission	No	Yes
Congestion Control	No	Yes
Security	No	Yes
Connection Migration	No	Yes
Multi-streaming	No	Yes

The host identity protocol (HIP) and locator/identifier separation protocol (LISP) for ID-LOC separation in the network layer have limitations due to their reliance on a centralized mapping system. To address this, mobile-oriented future internet (MOFI) has been developed, which includes features like global host ID and local locator (GILL), query-first data delivery (QFDD), and a distributed ID-LOC mapping system. GILL allows hosts to have a globally unique host ID and use local or private IP addresses as locators for packet routing. QFDD allows signaling operations for LOC queries before data transmission, selecting the best path. In the distributed ID-LOC mapping system, each access router manages the mappings for hosts in a decentralized manner.

Since its creation, the internet has experienced a steady increase in its user base and an even more significant traffic growth. This rapid growth demands various infrastructure, protocol, and application improvements [46]. The most commonly used stack for HTTPS involves TCP for carrying HTTP and TLS for securing it. This network model has been widely accepted and will continue to be deployed for many years [47,48]. However, there is potential to increase the speed with which a client and server establish a connection and deliver data between them. QUIC offers a possible alternative to the existing internet stack. QUIC's approach integrates the transport and security layers into a single layer, providing more functionality than TCP/TLS [49,50]. QUIC offers enhanced security features like encryption, authentication, multiplexing, connection migration, and congestion control, making it a more reliable and secure transport layer protocol for network communication.

The mobile-oriented future internet (MOFI) architecture uses UDP in the transport layer and the QUIC protocol to improve communication performance. This reduces signaling delay, data loss, congestion control, and data flow control. The integration of QUIC and MOFI enhances internet connectivity and mobility. AI-driven algorithms dynamically modify routing decisions based on historical data, network conditions, and host behavior, improving efficiency, security, and energy efficiency. Q-MOFI improves data transfer speed by determining target host locations. Future evaluation should focus on optimizing MOFI signaling capabilities for dynamic mobile network environments.

The contribution of the study is presented in various sections. Section 2 briefly describes the literature of the proposed study. The models of other researchers are presented in Section 3. Our proposed method is explained in Section 4. The results are discussed in Section 5, and Section 6 concludes the overall study.

2. Literature Review

In [51], Kim et al. introduced MOFI, a new architecture for the future internet, aimed at addressing the limitations of current architecture and better serving mobile users' needs. MOFI architecture supports high mobility, heterogeneity, and scalability, and addresses issues like bandwidth restrictions, high latency, and frequent disconnections. It features a global host ID and local locator (GILL), query-first data delivery, and a distributed ID-LOC mapping system. MOFI uses a layered architecture similar to TCP/IP, with the network layer divided into two sub-layers: communication and delivery. Security is a major concern in MOFI architecture, but it performs better in almost every aspect compared to HID and LISP protocols.

In [52], Arash et al. explored the evolution of transport protocols, highlighting the drawbacks of TCP and the benefits of the QUIC Protocol. They developed a methodology for evaluation and testing, including a testbed for controlled experiments under different network conditions. They compared QUIC and TCP protocols in various scenarios to determine which performed better under different conditions. TLS is used for secure cryptographic handshake [53,54]. The study tested page load time, multiple requests, bandwidth usage, and connection establishment time in mobile and desktop environments. The results showed that QUIC outperformed TCP in most aspects except for small requests. In video streaming, QUIC outperformed TCP for high-resolution videos. However, performance issues were found with window size, packet reordering, and handling large objects. QUIC's performance decreased when used on mobile devices or over cellular networks.

In [55], Viernickel et al. introduce a new architecture for multi-path QUIC (MPQUIC), allowing mobile devices to use multiple network interfaces like LTE and Wi-Fi. MPQUIC reduces download times compared to traditional QUIC, TCP, and MPTCP by increasing throughput through multipath utilization. The authors address issues with MPTCP by introducing new features like sub-flow establishment, packet numbers, stream offsets, and packet control. They also discuss packet scheduling and implementation ideas to enhance MPQUIC beyond traditional QUIC. The authors compare MPQUIC's performance with QUIC, TCP, and MPTCP in terms of file download speed and page load time.

In [56], Firmansyah et al. developed an adaptive MP-QUIC transmission method for IoT environments to overcome challenges like limited resources and heterogeneous network conditions. The method uses a proxy server to adjust MP-QUIC transmission parameters based on available network resources and IoT application requirements. The server uses an adaptive algorithm to modify the transmission rate and parallel channel number based on network capacity, latency, and packet loss rate. The proxy device sends data traffic via various backbone network paths while adjusting to changing network conditions. This approach improves MP-QUIC performance in IoT environments, especially in scenarios with limited resources and challenging network conditions.

In [57], M. Luglio et al. developed a QUIC-based proxy architecture for efficient hybrid backhaul transport, focusing on the virtual performance-enhancing proxy (vPEP). The vPEP aims to handle both switch-over and fail-over of end-to-end web traffic. It utilizes features

of the new QUIC protocol, including 0-RTT secure communication establishment and connection-oriented flow management across a UDP-based connection-less transport. The redirector (RD), proxy client (PC), and proxy server (PS) functions are dispersed over the backhaul link edges for smooth service. Future work aims to improve traffic classification systems and QUIC-based tunnel migration algorithms.

In [58], Bo Li et al. developed a new multimedia transfer (MMT) system using Google's QUIC transport protocol. MMT, a multimedia transport protocol, is designed for heterogeneous network transport and offers advantages over existing protocols like MPEG2-TS and RTP. The new MMT system combines MMT and QUIC characteristics, outperforming existing MMT systems. QUIC is claimed to offer better transport performance than HTTP, with experiments showing that in large delays and lossy networks, QUIC outperforms HTTP, while in other cases, it may achieve almost the same performance. Therefore, QUIC is a better choice for media transfer in the MMT system compared to HTTP.

In [59], V. Tong et al. proposed a new technique for traffic classification based on a convolutional neural network that combines feature extraction and classification into one system. To improve performance, this technique makes use of flow and packet-based features. Traffic may be classified using three different methods: payload-based, flow-static-based, and port-based. Traditional traffic categorization techniques, namely payload-based algorithms and ports, have become obsolete with the advent of QUIC. In order to identify various QUIC-based services, a novel flow static-based technique based on convolutional neural networks (CNNs) has been developed. This proposed method can accurately detect five different types of QUIC-based services. The use of flow-based features leads the processing and classification runtime to increase. We use all packets in the flows, thus when the number of packets in the flows is large, this can cause some issues.

In [60], L. Basyoni et al. developed QuicTor, a datagram-based design that enhances Tor's real-time communication capabilities. They use the QUIC transport protocol, known for its low latency and reliability, to improve Tor's performance. QuicTor offers a better user experience in real-time applications like video conferencing and online gaming. However, the authors do not discuss potential security implications, as the evaluation is conducted under specific conditions. The paper emphasizes the importance of improving Tor's real-time communication capabilities for user privacy and security.

In [61], Shreedhar et al. evaluated the performance of the QUIC protocol for web browsing, cloud storage, and video streaming. They compared it to traditional transport protocols like TCP and UDP using metrics like throughput, latency, and packet loss. The results showed that QUIC outperforms TCP for web and cloud storage workloads, but performs similarly or worse than UDP for video streaming. However, the study highlighted challenges in connection establishment, congestion control mechanisms, and handling packet losses. The results may not be generalizable to other scenarios or real-world applications.

In [62], B. Feng et al. introduced LISP, a new architecture model that separates IP address location and identity functions, adding a locator for device discovery and an ID for service identification. LISP offers advantages such as increased scalability, mobility, and security. It also improves routing efficiency and reduces traffic overhead. However, LISP relies on a centralized ID-LOC mapping system, Map Server, which has the disadvantage of concentrating traffic on a single server. Simulation results were used to compare LISP's performance with current internet architecture and other proposed architectures.

In [63], M.H. Mazhar et al. developed a new method to measure the quality of experience (QoE) of video streaming services using HTTPS and QUIC protocols. The method combines passive and active monitoring techniques, collecting network-level data like packet loss and delay, and application-level data like video bandwidth and buffering events. The authors introduce the video quality indicator (VQI), a new metric that combines network-level and application-level metrics to provide a single metric for video QoE. The results show that the proposed method accurately measures video QoE for both protocols and correlates well with user subjective ratings.

In [64], Cook et al. compared the performance of QUIC and TCP in different network scenarios, measuring throughput and latency under different levels of loss and delay. The study also examined the impact of different transport protocols on video streaming services quality. The results showed that QUIC generally performed better than TCP in terms of throughput and latency, with a notable performance gain when packet loss was high. TCP outperformed QUIC in certain scenarios [65]. The paper provides a thorough analysis and comparison of QUIC and TCP performance in various network scenarios, highlighting QUIC's suitability for video streaming applications. However, it only considers a limited set of scenarios and does not consider security or compatibility with existing infrastructure.

In [66], De Coninck et al. introduced multipath QUIC (MP-QUIC), a protocol that enables the use of multiple network paths simultaneously. They propose two methods for multipath support in QUIC: using a separate QUIC connection for each path or using a single connection with multiple paths. Both methods significantly outperform single-path QUIC in terms of throughput, especially in high packet loss rates. However, the second method, using a single connection with multiple paths, generally outperforms the first approach, with up to a 2.5× improvement in throughput in certain scenarios. MP-QUIC is compatible with existing systems and offers a simple and flexible method for multipath support.

In [67], Kharat et al. compared the performance of the QUIC protocol in wireless networks under various conditions, including distances, signal strengths, and network congestion levels. They used a Raspberry Pi testbed to compare QUIC's performance to the traditional TCP protocol. The results showed that QUIC outperformed TCP in terms of throughput, with up to a 70% improvement in certain scenarios. The authors concluded that QUIC is a promising protocol for wireless networks, but the higher packet loss rate in some scenarios may be a limitation. Further research is needed to optimize QUIC's performance and investigate the impact of different parameters.

3. Existing Model

In order to compare the outcomes of the contemporary pattern with our suggested scheme, MOFI-based UDP is implemented to evaluate the results of the present scheme. The MOFI architecture is built with three primary elements in the current scheme. Distributed ID-LOC mapping system, query first data delivery, global ID, and local LOC.

Figure 1 depicts the MOFI architecture model, which consists of two hosts—the Host 1 and the Host 2—and an AR. In this model, the mapping system is distributed, with each AR performing the mapping control operation using a local mapping controller (LMC) and a hash table. End-to-end communication between two hosts is performed with host ID, and location identifiers (LOCs) are used for data delivery in the network. To provide an optimal path, the LOC query operation is performed first to determine the LOC of the mobile host (MH) before data transmission. Each AR has an HID-LOC register (HLR) for mapping control. When the client wants to initiate communication with the server, it first looks up the RLOC of the destination host using its HID. This lookup is performed using the mapping control components of the MOFI protocol, which is responsible for managing and distributing the mapping information. Once the RLOC of the destination server is known, the client can use it to send packets to the destination. The packets are then forwarded by the routers in the network based on their RLOCs until they reach the destination server.

The transport layer is responsible for ensuring the reliable delivery of data between end-users and provides services such as segmentation of data into smaller units, flow control to prevent data overflow, and error checking to ensure the integrity of the data. MOFI uses the UDP protocol to transmit data across the network. UDP is a connection-less protocol, which means it does not establish a reliable connection between two devices. Instead, it directly sends data to the end-users without any guarantee of delivery of data packets.

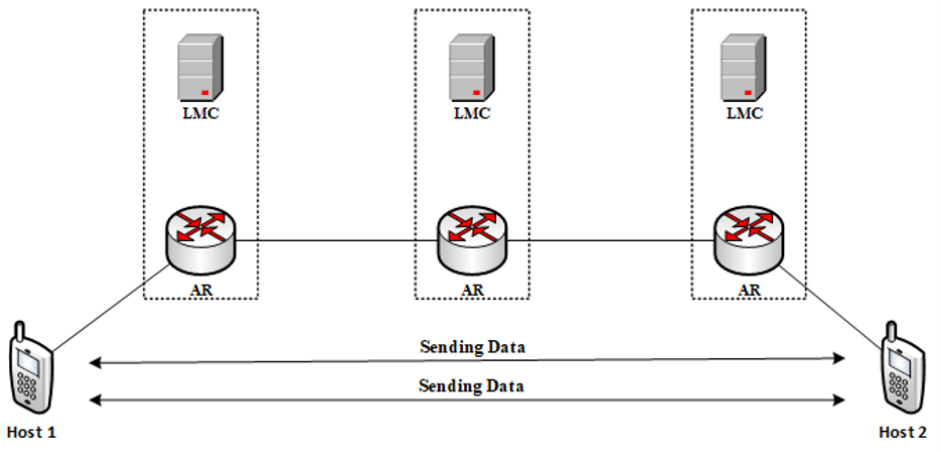

Figure 1. UDP-based MOFI.

When a host enters a network, a HID-LOC binding operation is performed to register the HID and LOC of the host in the mapping system. This is achieved by exchanging HBR and HBA messages, as shown in Figure 2. When a new host is attached to the network, the first HBR message is exchanged with AR. Then, the AR checks whether the HID and LOC of the host are present in the hash table. Since there will be no such addresses for the new host, AR forwards the HBR message to HLR. HLR then updates its register by saving the HID and LOC of the host and sends an HBA message to AR, which in turn forwards it to the host. After this binding operation, the HID and LOC of the host are registered in the hash table.

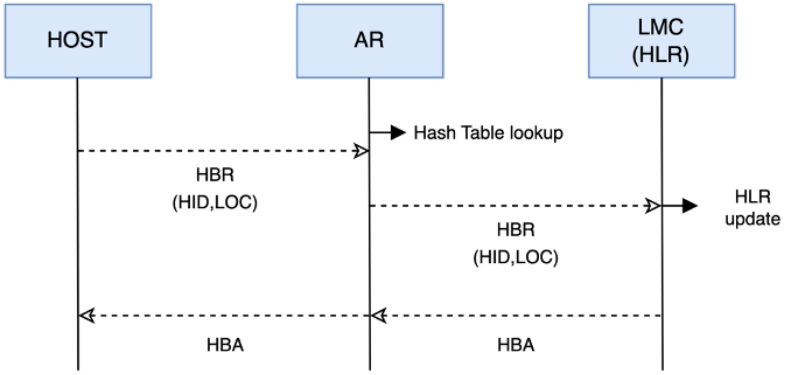

Figure 2. HID-LOC binding operation.

Initially, we will set up the MOFI to start simulation by creating nodes such as the client and server and assigning IP addresses to them. Once IP addresses are assigned, the MOFI routing system will map the location of the host to the AR using a mapping technique. If MOFI is configured successfully, all nodes will be in the idle state and the listening state, waiting for communication to start. As UDP is used in the transport layer, there is no connection between the client and server. Therefore, the client does not establish a connection with the server to send a datagram, but instead, it directly sends the datagram without establishing a connection. Similarly, the server just waits for datagrams to come without having to accept a connection. The datagram received by the server has the address of the sender, which is used by the server to send the data to the correct client.

To initiate the UDP client, it first creates the UDP socket and then sends a message to the server. The UDP client and server socket are created, and the client sends a message to the server, where the server waits until the datagram arrives from the client. The datagram is then processed by the server, which replies to the client and goes back to waiting until

the datagram arrives from the client. The client processes the reply given by the server and, if required, sends a message again to the server. This process continues until they exchange messages with each other. After completing the communication between the client and server, the socket will be closed, and the program will exit. The flowchart for the UDP-based MOFI is shown in Figure 3.

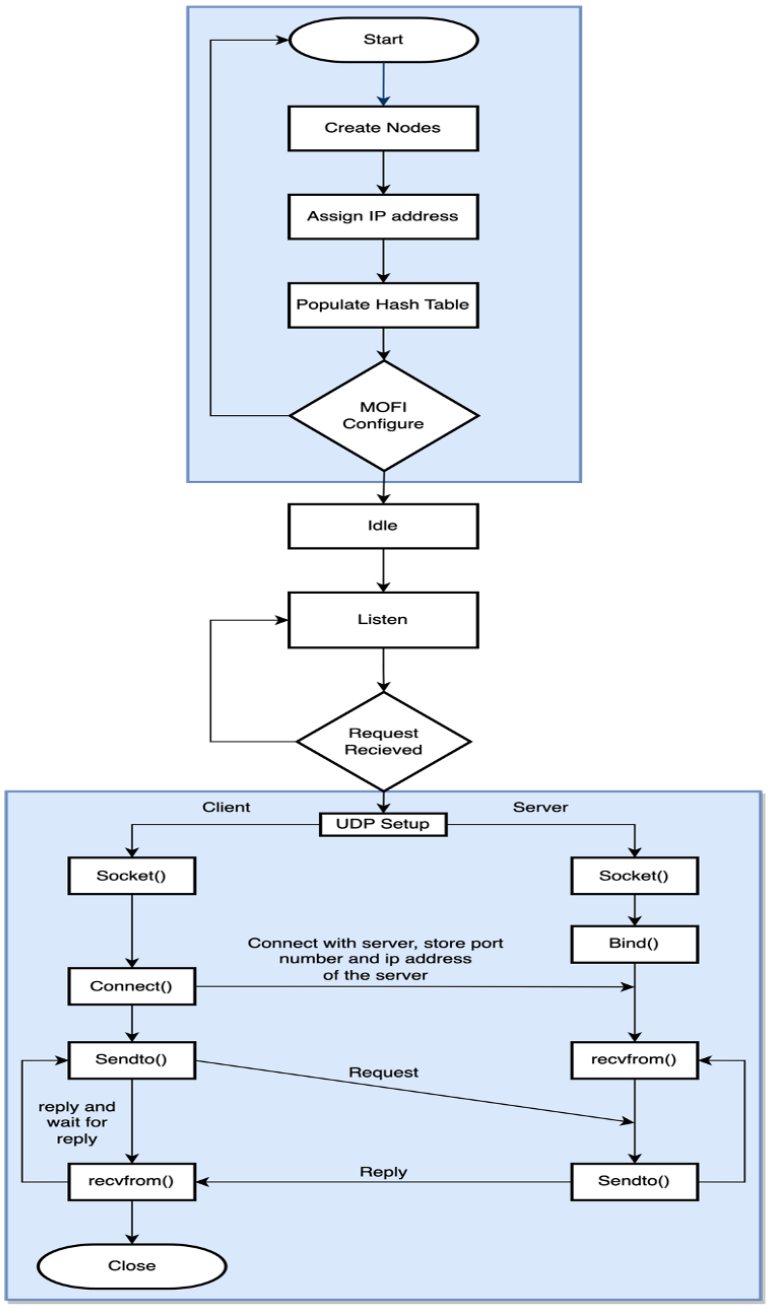

Figure 3. Flowchart representing MOFI Architecture.

4. Proposed Model

Our proposed scheme is to implement MOFI using the QUIC protocol instead of UDP. There are limitations associated with UDP such as it being a connectionless protocol that does not provide any delivery or reliability guarantees, which may result in lost or delayed packets and negatively impact the user experience. Additionally, UDP lacks a mechanism for retransmitting lost packets and congestion control. In the absence of congestion control, a sender may continue to transmit data at full speed even when the network is congested,

leading to dropped packets and degraded performance. To address these limitations, QUIC builds on the strengths of UDP while adding features such as reliability, security, multiplexing, establishing the connection before data are transmitted, and congestion control. The protocol stack for the QUIC is shown in Figure 4.

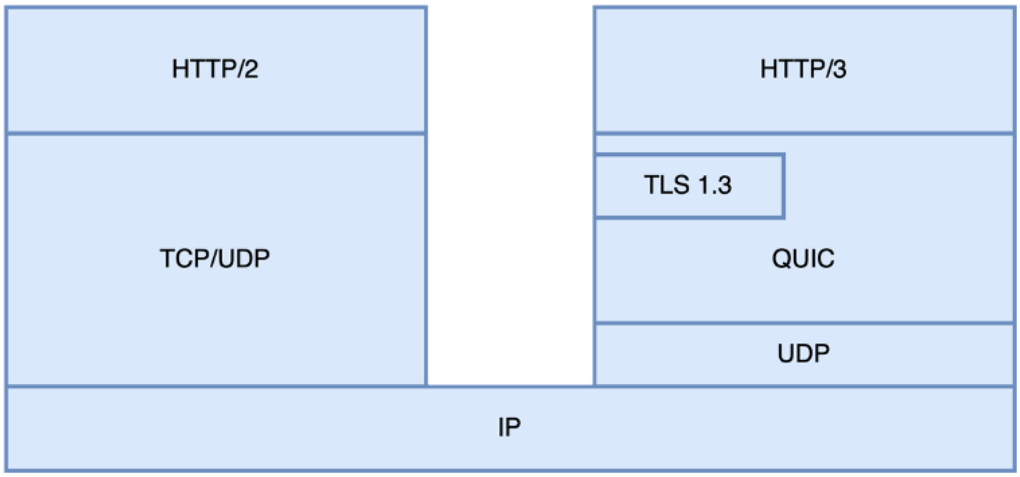

Figure 4. Protocol stack of Q-MOFI.

QUIC improves the speed, security, and reliability of the internet by combining the low latency of UDP with the reliability and security of TCP. It provides encryption, congestion control, multiplexing, and flow control. QUIC's multiplexing feature allows for multiple streams of data to be sent over a single connection, improving efficiency and reducing latency. Additionally, QUIC includes multi-homing support, which allows a client to maintain multiple network connections to a server simultaneously, improving redundancy and reducing the impact of network failures. QUIC also supports stream prioritization, enabling clients to prioritize certain streams over others, and connection migration, allowing clients to seamlessly switch between different network connections without disrupting ongoing transfers. Additionally, QUIC uses a built-in congestion control mechanism to prevent network congestion and ensure efficient use of network resources, and uses a retransmission mechanism to ensure reliability in the transport of data over the internet.

Our proposed model, Q-MOFI, which is based on the QUIC protocol, is shown in Figure 5; it consists of Hosts 1 and 2, along with an AR (Access Router). When Host 1 wants to communicate with Host 2, they establish a connection with Host 2 before transmitting any data. To establish the connection, they perform a 1-RTT handshake. Host 1 generates a unique identifier (CID) to identify the connection and sends an Initial Packet to Host 2. Host 2 receives the initial packet, generates its own CID to identify the connection, and sends a server handshake packet back to Host 1 with its CID. The server handshake packet includes information such as Host 2's transport parameters and cryptographic keys. Host 1 receives the server handshake packet and responds with a client handshake packet, which includes information such as Host 1's transport parameters and cryptographic keys. QUIC also supports 0-RTT, which is the fastest way to establish a QUIC connection. It enables Host 1 to send data in the first packet it sends to Host 2, without waiting for a response from Host 2. This eliminates the need for an RTT (round-trip time) and reduces the overall time required to establish the connection.

To start the simulation, we first set up the MOFI. We begin by creating nodes, such as the client and server, and assigning IP addresses to them. Once IP addresses have been assigned, the MOFI routing system maps the location of the hosts to their respective access routers (AR) using a mapping technique. If MOFI has been configured successfully, all nodes are in the idle state and are waiting in the listen state for communication to begin.

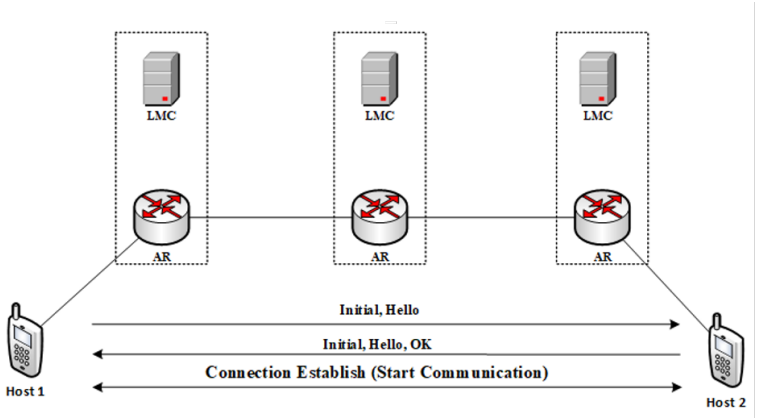

Figure 5. QUIC-based MOFI.

In the transport layer, QUIC is used instead of UDP. When a request for communication is received, the nodes first check if there is an established connection with the intended recipient. If a connection has already been established, the data are stored and the communication can proceed directly. If no connection has been established, the nodes initiate a three-way handshake. The client sends a SYN sent message along with the initial message and a unique identifier (CID). The receiver host responds with a SYN-ACK message, which includes parameters required for the connection establishment. The client sends an ACK message in response to the SYN-ACK message, completing the three-way handshake and establishing the connection. Once the connection has been established, the nodes can start communicating. The flowchart for the Q-MOFI is illustrated in Figure 6.

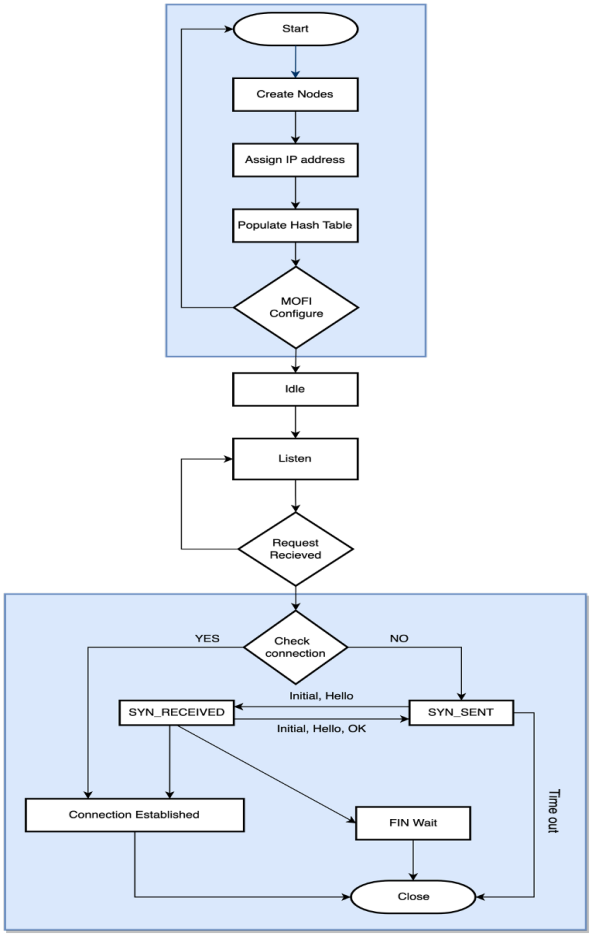

Figure 6. Flowchart representing Q-MOFI Architecture.

5. Experiments and Results

In this section, we describe the experiments we conducted to evaluate the performance of the QUIC protocol in a MOFI network using Golang for simulation. Golang has become a popular programming language for implementing QUIC. Golang was created by Google and is renowned for its efficiency, concurrency support, and simplicity, making it an excellent option for creating network applications. In order to implement QUIC, Golang also offers a standard library that has functions for establishing QUIC connections, transmitting and receiving packets, and handling failures. The development and testing of QUIC-based apps take less time and effort thanks to this library, which also makes QUIC implementation simpler. The experimental setup used to conduct the testing is illustrated in Figure 7. In this setup, two hosts (Host 1 and Host 2) and three ARs with LMCs (AR 1, AR 2, and AR 3) were employed. Host 1 acted as the sender, while Host 2 played the role of the receiver. Host 1 used the HID of 192.168.0.125, and Host 2 used the HID of 192.168.0.129. For the LOC, a private IPv4 address (172.20.0.1) was used as the A-LOC, while a public IPv4 address (216.58.194.x) was used as the B-LOC. Our goal was to measure the latency, delay, packet loss, and throughput of the Q-MOFI and compare it with the other existing model, MOFI, which uses UDP.

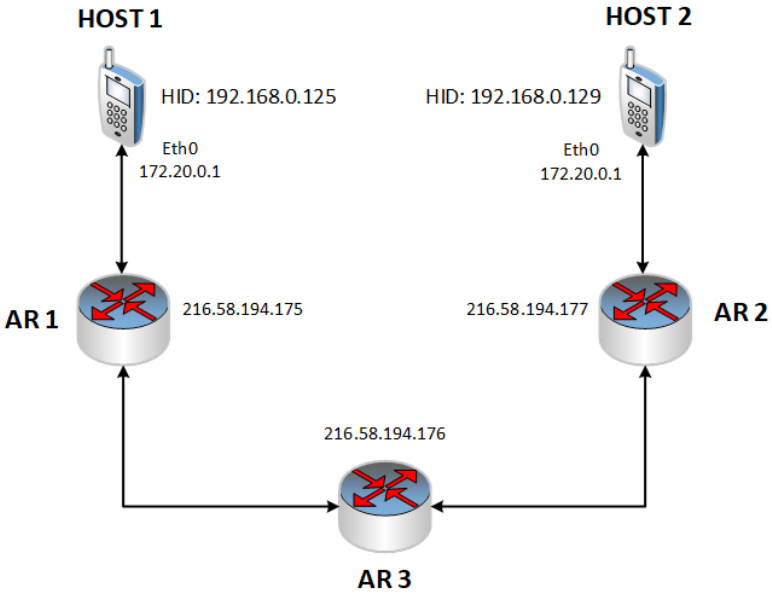

Figure 7. Experimental Setup.

5.1. Handshake and Connection Establishment

The user datagram protocol (UDP) does not utilize a handshake for establishing a connection between two endpoints. UDP is a connectionless protocol that does not require a dedicated connection between two endpoints for data transmission. It is faster and more efficient than TCP, but there's no guarantee of data reception or correct order. UDP does not use a handshake to establish a connection, so there's no need to establish a session before transmitting data. The source endpoint sends a datagram to the destination endpoint, and the destination endpoint receives it without any setup or coordination.

Figure 8 illustrates the handshake mechanism used by UDP and QUIC. UDP is connectionless, allowing fast and lightweight data transfer without an explicit handshake. QUIC, on the other hand, emphasizes reliability and security through a robust handshake procedure, ensuring secure and efficient data exchange. Both methods demonstrate the importance of careful handshakes in data transfer. One of the main features of QUIC is its ability to establish a secure, reliable connection with low latency, even in poor network conditions. The handshake process is similar to the handshake used by TLS (Transport Layer Security), which is the protocol used to secure many web applications. The handshake

process in QUIC involves a series of messages exchanged between the client and server endpoints. The client initiates the handshake by sending a "ClientHello" message to the server, which includes information about the client's preferred cryptographic algorithms, the server's hostname, and other parameters. The server responds with a "ServerHello" message, which includes its own preferred cryptographic algorithms and other parameters. The server also sends a digital certificate that the client can use to verify the server's identity and establish a secure connection. Once the handshake is complete, the client and server can begin sending data to each other over the QUIC connection. Because QUIC uses UDP as its underlying transport protocol, it does not suffer from the same latency issues as TCP, and it can establish connections more quickly and with less overhead.

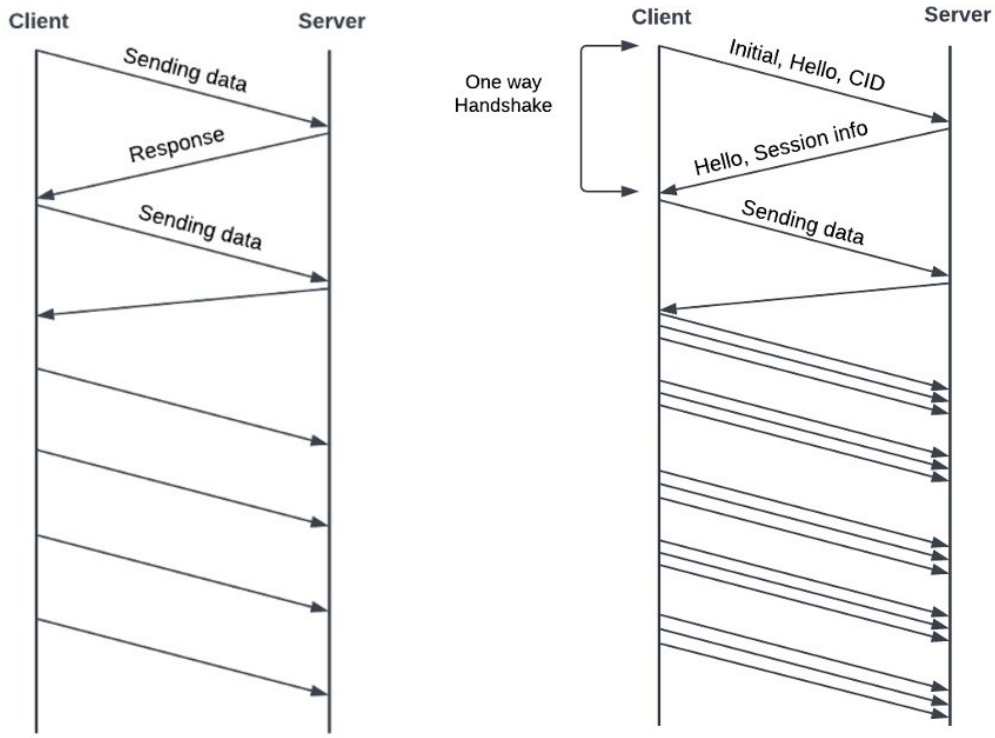

Figure 8. Handshake for MOFI and Q-MOFI.

5.2. Wireshark Results

Wireshark is a popular network protocol analyzer that allows you to capture and examine network traffic in real time. Figure 3 shows the Wireshark result using the UDP protocol. When capturing UDP traffic using Wireshark, you will see a summary of each UDP datagram that is transmitted over the network. UDP datagram in Wireshark typically includes the following information: source IP address and port number, destination IP address and port number, the length of the datagram (including header and data).

Figure 9 shows the Wireshark result of the data transfer between two points by using the UDP protocol. Wireshark also provides a detailed view of each datagram, which includes information about the header and data fields. The header fields in a UDP datagram include the source port number, destination port number, and length. Wireshark can filter and analyze UDP traffic based on various criteria, such as the source or destination IP address, port number, or protocol type. This is useful for troubleshooting network issues, analyzing application behavior, and detecting security threats.

Figure 10 shows the Wireshark result using the QUIC protocol. The result provides valuable information for analyzing the network communication between the client and server. In addition to the source and destination IP addresses and ports, the result contains the size of the packet and the checksum, which are important for verifying the integrity and reliability of the data transmission. Moreover, Wireshark offers a thorough view of each datagram that contains details on the header and data fields. The QUIC protocol uses

different types of frames to carry various data payloads, including handshake messages, encrypted data, and control messages. Analyzing these frames can provide insights into the performance and behavior of the QUIC protocol, such as the number of packets exchanged during the connection establishment, the size and type of the data payloads, and the latency of the transmission. By using Wireshark to examine the QUIC frames, you can identify any issues or anomalies that may affect the network communication and optimize the performance of the network.

Source	Destination	Protocol	Length	Info
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	50	64815 → 8080 Len=8
192.168.0.129	192.168.0.125	UDP	67	8080 → 64815 Len=25
192.168.0.125	192.168.0.129	UDP	51	64815 → 8080 Len=9
192.168.0.129	192.168.0.125	UDP	68	8080 → 64815 Len=26
192.168.0.125	192.168.0.129	UDP	51	64815 → 8080 Len=9
192.168.0.129	192.168.0.125	UDP	68	8080 → 64815 Len=26
192.168.0.125	192.168.0.129	UDP	51	64815 → 8080 Len=9
192.168.0.129	192.168.0.125	UDP	68	8080 → 64815 Len=26
192.168.0.125	192.168.0.129	UDP	51	64815 → 8080 Len=9
192.168.0.129	192.168.0.125	UDP	68	8080 → 64815 Len=26
192.168.0.125	192.168.0.129	UDP	51	64815 → 8080 Len=9

Figure 9. Evaluation of MOFI results via Wireshark.

Source	Destination	Protocol	Length	Info
192.168.0.125	192.168.0.129	QUIC	1294	Initial, DCID=5f4842937ea5623b96cd572b559928, PKN: 5, PADDING, CRYPTO
192.168.0.125	192.168.0.129	QUIC	1294	Initial, DCID=5f4842937ea5623b96cd572b559928, PKN: 6, PADDING, CRYPTO
192.168.0.125	192.168.0.129	QUIC	1294	Initial, DCID=5f4842937ea5623b96cd572b559928, PKN: 7, PADDING, CRYPTO
192.168.0.125	192.168.0.129	QUIC	1294	Initial, DCID=5f4842937ea5623b96cd572b559928, PKN: 8, PADDING, CRYPTO
192.168.0.129	192.168.0.125	QUIC	1294	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	1294	Initial, SCID=3619cf3a, PKN: 1, PADDING, CRYPTO
192.168.0.129	192.168.0.125	QUIC	1294	Initial, SCID=3619cf3a, PKN: 2, PADDING, CRYPTO
192.168.0.125	192.168.0.129	QUIC	1294	Handshake, DCID=3619cf3a
192.168.0.125	192.168.0.129	QUIC	78	Handshake, DCID=3619cf3a
192.168.0.125	192.168.0.129	QUIC	71	Protected Payload (KP0), DCID=3619cf3a
192.168.0.125	192.168.0.129	QUIC	67	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	80	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	78	Handshake, SCID=3619cf3a
192.168.0.129	192.168.0.125	QUIC	277	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	67	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	76	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	70	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	80	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	82	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	85	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	81	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	85	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	81	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	85	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	81	Protected Payload (KP0)
192.168.0.125	192.168.0.129	QUIC	86	Protected Payload (KP0)
192.168.0.129	192.168.0.125	QUIC	1394	Protected Payload (KP0)

```
Frame 26: 1294 bytes on wire (10352 bits), 1294 bytes captured (10352 bits) on interface en0, id 0
    Section number: 1
  > Interface id: 0 (en0)
    Encapsulation type: Ethernet (1)
    Arrival Time: Feb 19, 2023 17:07:52.147843000 PKT
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1676808472.147843000 seconds
    [Time delta from previous captured frame: 0.000020000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
```

Figure 10. Representation of Q-MOFI outcomes via Wireshark.

5.3. Latency

Figure 11 shows the latency of the MOFI and Q-MOFI, Different size packets (bytes) are transmitted from one host to another i.e., 100, 200, 300, 400, and 500. The graph indicates that Q-MOFI performs better than MOFI as the packet size increases. As the packet size

increases in MOFI, the transmission time (ms) also increases significantly. However, Q-MOFI performs better than MOFI as the transmission time for packet delivery does not increase significantly. Thus, the graph shows that Q-MOFI outperforms MOFI in terms of latency.

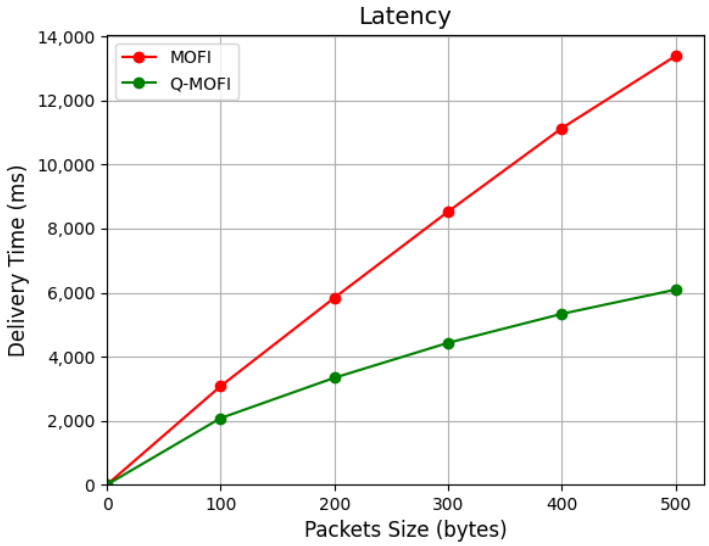

Figure 11. Packet delivery time (ms).

5.4. Binding Delay

Figure 12 shows the binding delay for both MOFI and Q-MOFI. The graph shows that as the number of hosts increases, the MOFI binding delay also increases. However, the binding operation in Q-MOFI is not affected as much as MOFI when the number of hosts increases. This is because Q-MOFI has a congestion control algorithm that can transmit data based on the network conditions and has the ability to migrate data to another path when there is high traffic on one path without any data loss. On the other hand, MOFI does not have any congestion control or flow control mechanism, which is why its performance deteriorates when the number of hosts increases, and the network conditions change. Q-MOFI, on the other hand, has flow control and congestion control mechanisms that can transmit data based on the network's behavior, and its performance is not significantly affected.

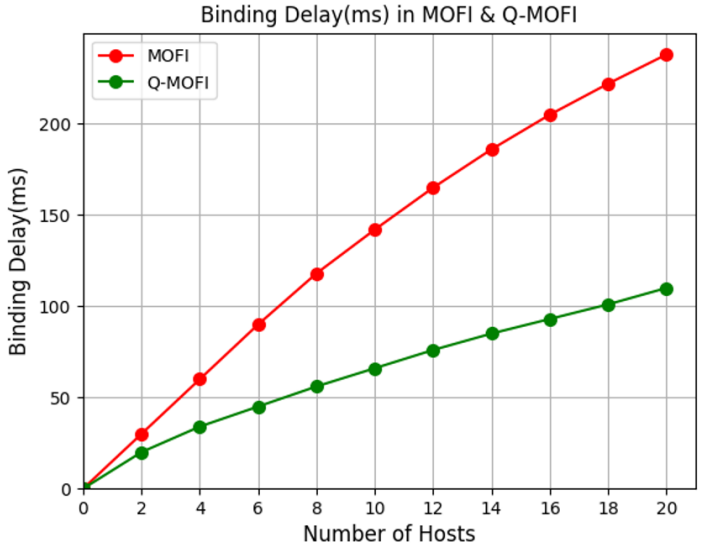

Figure 12. Binding delay (ms) when the host increases.

5.5. Throughput

Throughput is a performance metric that measures the amount of data that can be transmitted over a network in a given period of time. We used the following formula to calculate the throughput.

$$Throughput = \frac{Total\ amount\ of\ data\ transmitted}{time\ taken}$$

This formula calculates the throughput as the amount of data transmitted over a specific period of time. The units of measurement for data can be bytes or bits, and the units of time can be seconds, minutes, or hours. Throughput is typically expressed in bits per second (bps) and represents the overall effectiveness of the communication system. A higher throughput indicates a faster and more efficient system, as it can transfer more data in less time. The throughput of communication protocols, such as UDP or QUIC, is influenced by various factors, including network conditions, packet loss, latency, and congestion control techniques. Throughput is a crucial performance indicator for communication systems, as it determines the quality and speed of data transmission. By optimizing the throughput of a communication protocol and closely monitoring the network condition, it is possible to achieve faster and more efficient data transfer, leading to improved user experience and overall system performance.

Figure 13 shows a comparison of the MOFI and Q-MOFI throughput. Due to its usage of flow control, multiplexing, and congestion control algorithms, Q-MOFI has the potential to achieve greater throughput than MOFI. These features improve the reliability and efficiency of data transmission, enabling more data to be transmitted within a given period of time. The graph shows that Q-MOFI performs much better than MOFI in terms of throughput.

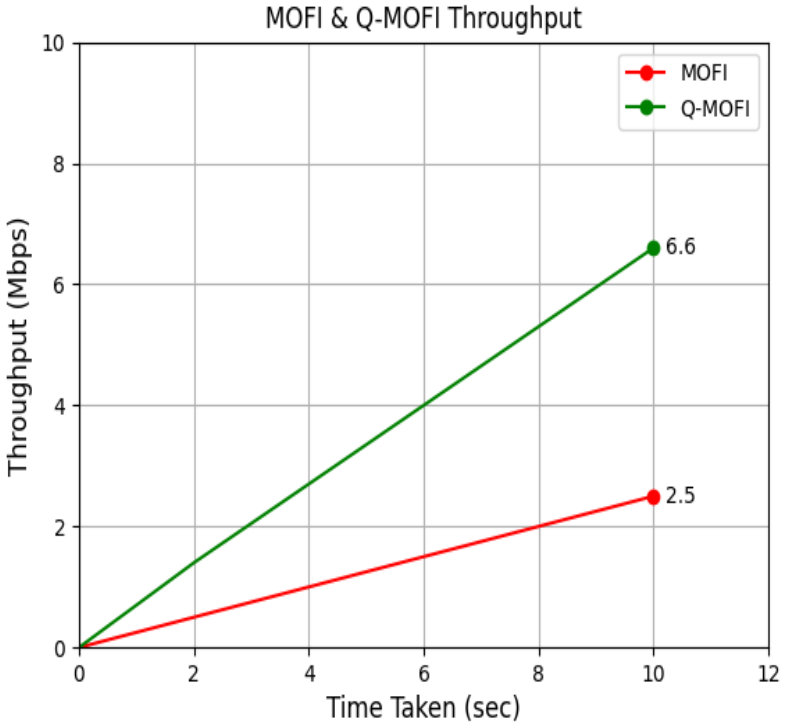

Figure 13. Throughput comparison.

5.6. Packet Loss

Figure 14 shows the delivery time or latency for MOFI and Q-MOFI with varying packet loss percentages in a network for a message transmission from one host to another. It shows that as the packet loss percentage increases, the latency for both protocols also increases. If there is a packet loss of 5% or 10%, the latency increases for both MOFI and

Q-MOFI because the lost packets will need to be re-transmitted, which will increase the overall time it takes for the data to be transmitted. However, the increase in latency is more significant for MOFI compared to Q-MOFI when there is packet loss.

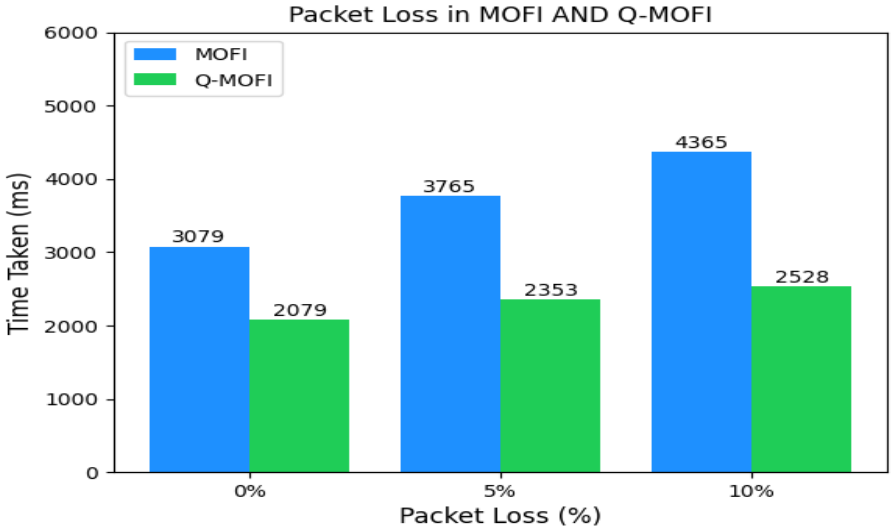

Figure 14. Packet loss.

5.7. Time Taken According to Packet Size

Figure 15 illustrates the time taken by MOFI and Q-MOFI for data transfer measured in megabits per second (Mbs). The tests were conducted ranging from 1 Mbs to 9 Mbs. The results indicate that Q-MOFI requires less time compared to MOFI. As the data volume increases, MOFI takes more time, whereas Q-MOFI continues to perform well. Multiplexing in Q-MOFI allows for multiple data streams to be sent over a single connection, leading to better utilization of the underlying network resources and potentially faster data transfer. Q-MOFI's multiplexing divides data into multiple independent streams, each assigned its own stream identifier. These streams are interleaved and transmitted within a single packet, capable of carrying multiple data streams simultaneously. In contrast to MOFI, which transfers only one data stream at a time over a single connection, Q-MOFI offers a significant performance improvement.

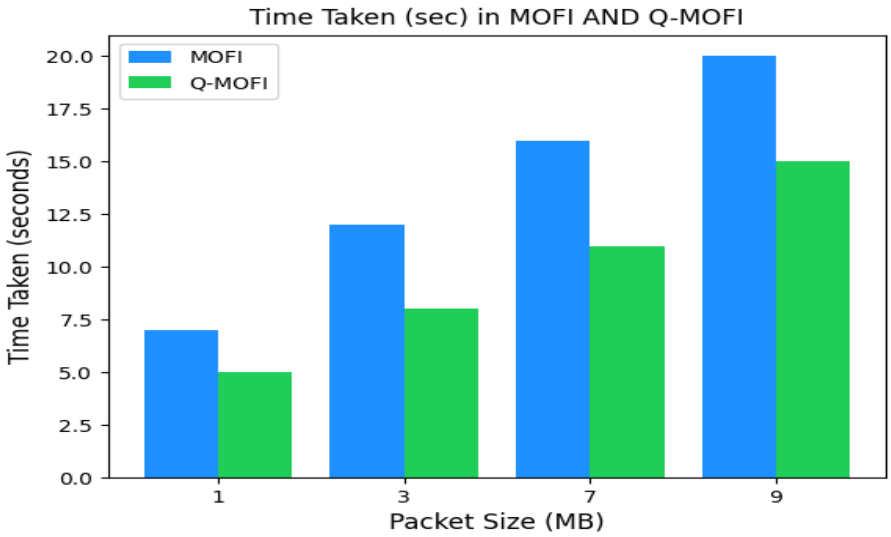

Figure 15. Time taken packet size (MB).

6. Conclusions and Future Work

The internet is expected to become a "mobile hub" in the future, underscoring the importance of ensuring seamless mobility for mobile devices and users. The Q-MOFI model, based on the QUIC protocol, has been compared to the existing MOFI model and it outperforms it in terms of latency, throughput, delay, and packet loss. It also shows significant gains in throughput and a reduction in latency. However, the Q-MOFI model has limitations, including security and privacy implications, and further research is needed to address these concerns. It is superior to UDP in numerous applications, fostering uninterrupted connectivity and enhanced mobility for mobile devices and IoT services. It is ideal for real-time data transmission applications, like video streaming, online gaming, and remote conferencing. Q-MOFI aims to reduce latency, improve transmission reliability, and increase throughput, but it faces challenges like scalability, interoperability with existing systems, and potential security issues.

In conclusion, the Q-MOFI model, built upon the QUIC protocol, presents a compelling rationale for the advancement of mobile-centric internet services. Its superiority over current MOFI designs in terms of latency, throughput, delay, and packet loss makes it suitable for a wide range of applications. Further research and development could lead to innovative solutions that improve the security, privacy, and convenience of mobile internet services.

Author Contributions: Conceptualization, M.T.; overall writing and editing, M.T.; validation, M.T.; conceptualization, M.G.; methodology, M.G.; software, M.G.; validation, M.G. resources, M.G.; writing—original draft preparation, M.T.; writing—review and editing, S.J.K. and H.A.; supervision, M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data will be provided on request.

Acknowledgments: The authors would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under Project Number No. R-2024-936.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fang, C.; Yao, H.; Wang, Z.; Wu, W.; Jin, X.; Yu, F.R. A survey of mobile information-centric networking: Research issues and challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2353–2371. [CrossRef]
2. Jung, L.T.; Wagan, A.A. Distributed Network Mobility Management Scheme for Network Mobility. In Proceedings of the 2018 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 13–14 August 2018; pp. 1–6. [CrossRef]
3. Ullah, A.; Chen, X.; Yang, J. Design and Implementation of MobilityFirst Future Internet Testbed. In Proceedings of the 2020 3rd International Conference on Hot Information-Centric Networking (HotICN), Hefei, China, 12–14 December 2020; pp. 170–174.
4. Abadleh, A.; Tareef, A.; Btoush, A.; Mahadeen, A.; Al-Mjali, M.M.; Alja'Afreh, S.S.; Alkasasbeh, A.A. Comparative Analysis of TCP Congestion Control Methods. In Proceedings of the 2022 13th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 21–23 June 2022; pp. 474–478.
5. Al-Kadhim, H.M.; Al-Raweshidy, H.S. Energy efficient and reliable transport of data in cloud-based IoT. *IEEE Access* **2019**, *7*, 64641–64650. [CrossRef]
6. Khan, H.U.; Hussain, A.; Nazir, S.; Ali, F.; Khan, M.Z.; Ullah, I. A service-efficient proxy mobile IPv6 extension for IoT domain. *Information* **2023**, *14*, 459.
7. Khan, S.; Luo, F.; Zhang, Z.; Ullah, F.; Amin, F.; Qadri, S.; Heyat, M.; Ruby, R.; Wang, L.; Ullah, S.; et al. A survey on X. 509 public-key infrastructure, certificate revocation, and their modern implementation on blockchain and ledger technologies. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2529–2568.
8. Polese, M.; Jana, R.; Zorzi, M. TCP and MP-TCP in 5G mmWave networks. *IEEE Internet Comput.* **2017**, *21*, 12–19. [CrossRef]
9. Wang, Q.; Dai, W.; Zhang, C.; Zhu, J.; Ma, X. A Compact Constraint Incremental Method for Random Weight Networks and Its Application. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–9. [CrossRef]
10. Polese, M.; Chiariotti, F.; Bonetto, E.; Rigotto, F.; Zanella, A.; Zorzi, M. A survey on recent advances in transport layer protocols. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3584–3608.
11. Dai, M.; Sun, G.; Yu, H.; Niyato, D. Maximize the Long-Term Average Revenue of Network Slice Provider via Admission Control Among Heterogeneous Slices. *IEEE/ACM Trans. Netw.* **2023**, 1–16. [CrossRef]

12. Yamanaka, T.; Iwai, T.; Kubo, R. Quality of performance aware data transmission for energy-efficient networked control. *IEEE Access* **2021**, *9*, 5769–5778. [CrossRef]
13. Hu, J.; Wu, Y.; Li, T.; Ghosh, B.K. Consensus Control of General Linear Multiagent Systems with Antagonistic Interactions and Communication Noises. *IEEE Trans. Autom. Control* **2019**, *64*, 2122–2127. [CrossRef]
14. Alvarez-Horcajo, J.; Lopez-Pajares, D.; Arco, J.M.; Carral, J.A.; Martinez-Yelmo, I. TCP-path: Improving load balance by network exploration. In Proceedings of the 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), Prague, Czech Republic, 25–27 September 2017; pp. 1–6.
15. Xie, H.; Li, T. Revisiting loss recovery for high-speed transmission. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 11–13 April 2022; pp. 1987–1992.
16. Hotchi, R.; Chibana, H.; Iwai, T.; Kubo, R. Active queue management supporting TCP flows using disturbance observer and smith predictor. *IEEE Access* **2020**, *8*, 173401–173413.
17. Li, L.; Yao, L. Fault Tolerant Control of Fuzzy Stochastic Distribution Systems with Packet Dropout and Time Delay. *IEEE Trans. Autom. Sci. Eng.* **2023**, 1–10. [CrossRef]
18. Liu, C.; Wu, T.; Li, Z.; Ma, T.; Huang, J. Robust Online Tensor Completion for IoT Streaming Data Recovery. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 10178–10192. [CrossRef] [PubMed]
19. Toprasert, T.; Lilakiataskun, W. TCP congestion control with MDP algorithm for IoT over heterogeneous network. In Proceedings of the 2017 17th International Symposium on Communications and Information Technologies (ISCIT), Cairns, QLD, Australia, 25–27 September 2017; pp. 1–5.
20. Al-Saadi, R.; Armitage, G.; But, J.; Branch, P. A survey of delay-based and hybrid TCP congestion control algorithms. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3609–3638. [CrossRef]
21. Garcia-Luna-Aceves, J.J.; Albalawi, A.A. A Connection-Free Reliable Transport Protocol. In Proceedings of the 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 6–8 November 2020; pp. 1–6.
22. Wang, X.; Parish, D.J. Optimized multi-stage TCP traffic classifier based on packet size distributions. In Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service, Athens, Greece, 13–19 June 2010; pp. 98–103.
23. Kim, S.; Shin, S.; Moon, J. UDP-based Extremely Low Latency Streaming. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022; pp. 94–99.
24. Li, H.; Zhao, Y.; Wu, R. Optimal Design of UDP Protocol in Embedded Real-Time OS. In Proceedings of the 2021 13th International Conference on Advanced Infocomm Technology (ICAIT), Yanji, China, 15–18 October 2021; pp. 180–184.
25. Hei, X.; Chen, J.; Lu, H.; Xie, G.; Meng, H. A UDP-based way to improve data transmission reliability. In Proceedings of the 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, China, 28–30 May 2017; pp. 2612–2617.
26. Guo, R.; Liu, H.; Liu, D. When Deep Learning-Based Soft Sensors Encounter Reliability Challenges: A Practical Knowledge-Guided Adversarial Attack and Its Defense. *IEEE Trans. Ind. Inform.* **2023**, 1–13. [CrossRef]
27. AL-Dhief, F.T.; Sabri, N.; Latiff, N.A.; Malik, N.N.N.A.; Abbas, M.; Albader, A.; Mohammed, M.A.; AL-Haddad, R.N.; Salman, Y.D.; Khanapi, M.; et al. Performance comparison between TCP and UDP protocols in different simulation scenarios. *Int. J. Eng. Technol.* **2018**, *7*, 172–176.
28. Dey, N.; Neha, N.; Hariprasad, M.S.; Hya, S.; Moharir, M.; Akram, M. A Detail Survey on QUIC and its Impact on Network Data Transmission. In Proceedings of the 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 28–30 April 2022; pp. 378–385. [CrossRef]
29. Michel, F.; Cohen, A.; Malak, D.; De Coninck, Q.; Médard, M.; Bonaventure, O. FlEC: Enhancing QUIC With Application-Tailored Reliability Mechanisms. *IEEE/ACM Trans. Netw.* **2023**, *31*, 606–619. [CrossRef]
30. Ramasamy, V.; Pop, M.D. The Future Network 2030: A Simplified Intelligent Transportation System. In *Intelligent Technologies for Sensors: Applications, Design, and Optimization for a Smart World*; Apple Academic Press: Palm Bay, FL, USA, 2023; p. 315.
31. Soni, M.; Rajput, B.S. Security and performance evaluations of QUIC protocol. In *Data Science and Intelligent Applications: Proceedings of ICDSIA 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 457–462.
32. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The QUIC transport protocol: Design and Internet-scale deployment. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Beijing, China, 19–23 August 2017; pp. 183–196.
33. Jeddou, S.; Diez, L.; Abdellah, N.; Baina, A.; Calvo, R.A. On the Performance of Transport Protocols Over mmWave Links: Empirical Comparison of TCP and QUIC. *IEEE Open J. Commun. Soc.* **2023**, *4*, 2596–2608. [CrossRef]
34. Cao, K.; Wang, B.; Ding, H.; Lv, L.; Tian, J.; Hu, H.; Gong, F. Achieving Reliable and Secure Communications in Wireless-Powered NOMA Systems. *IEEE Trans. Veh. Technol.* **2021**, *70*, 1978–1983. [CrossRef]
35. Guo, Y.; Zhang, C.; Wang, C.; Jia, X. Towards Public Verifiable and Forward-Privacy Encrypted Search by Using Blockchain. *IEEE Trans. Dependable Secur. Comput.* **2023**, *20*, 2111–2126. [CrossRef]
36. Cao, K.; Ding, H.; Li, W.; Lv, L.; Gao, M.; Gong, F.; Wang, B. On the Ergodic Secrecy Capacity of Intelligent Reflecting Surface Aided Wireless Powered Communication Systems. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 2275–2279. [CrossRef]
37. Cui, Y.; Li, T.; Liu, C.; Wang, X.; Kühlewind, M. Innovating transport with QUIC: Design approaches and research challenges. *IEEE Internet Comput.* **2017**, *21*, 72–76. [CrossRef]

38. Biswal, P.; Gnawali, O. Does QUIC make the web faster? In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
39. De Coninck, Q.; Bonaventure, O. Multiflow QUIC: A generic multipath transport protocol. *IEEE Commun. Mag.* **2021**, *59*, 108–113. [CrossRef]
40. Jung, J.; An, D. Access latency reduction in the QUIC protocol based on communication history. *Electronics* **2019**, *8*, 1204.
41. Iyengar, J.; Thomson, M. QUIC: A UDP-Based Multiplexed and Secure Transport. May 2021. Available online: https://www.rfc-editor.org/info/rfc9000 (accessed on 11 January 2024).
42. Celestino, A.; Romano, S.P. An attempt at introducing Multipath in QUIC. In Proceedings of the 2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN), Zagreb, Croatia, 2–5 July 2019; pp. 352–357.
43. Tan L.; Su, W.; Liu, Y.; Gao, X.; Li, N.; Zhang, W. Proactive connection migration in QUIC. In Proceedings of the MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Darmstadt, Germany, 7–9 December 2020; pp. 476–481.
44. Yan, Y.; Yang, Z. When QUIC's Connection Migration Meets Middleboxes: A case study on mobile Wi-Fi hotspot. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6.
45. Kanagarathinam, M.; Hasan, F.; Rengan, S.; Singh, S.; Choudhary, K.; Qureshi, F.; Lee, H. Enhanced QUIC Protocol for transferring Time-Sensitive Data. In Proceedings of the 2022 IEEE International Conference on Communications Workshops (ICC Workshops), Seoul, Republic of Korea, 16–20 May 2022; pp. 1–6.
46. Nepomuceno, K.; De Oliveira, I.N.; Aschoff, R.R.; Bezerra, D.; Ito, M.S.; Melo, W.; Szabó, G. QUIC and TCP: A performance evaluation. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 00045–00051.
47. Xu, H.; Han, S.; Li, X.; Han, Z. Anomaly Traffic Detection Based on Communication-Efficient Federated Learning in Space-Air-Ground Integration Network. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 9346–9360. [CrossRef]
48. Zhang, H.; Mi, Y.; Fu, Y.; Liu, X.; Zhang, Y.; Wang, J.; Tan, J. Security Defense Decision Method Based on Potential Differential Game for Complex Networks. *Comput. Secur.* **2023**, *129*, 103187. [CrossRef]
49. Yosofie, M.; Jaeger, B. Recent progress on the QUIC protocol. *Network* **2019**, *77*, 77–81.
50. Bujari, A.; Franco, M.; Palazzi, C.E.; Quadrini, M.; Roseti, C.; Zampognaro, F. Use of QUIC Protocol for Efficient Data Transmission Over Satellite in Emergency Scenario. In Proceedings of the 2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), Cosenza, Italy, 13 September 2023; pp. 1–6. [CrossRef]
51. Kim, J.-I.; Jung, H.; Koh, S.-J. Mobile oriented future Internet (MOFI): Architectural design and implementations. *ETRI J.* **2013**, *35*, 666–676. [CrossRef]
52. Kakhki, A.M.; Jero, S.; Choffnes, D.; Nita-Rotaru, C.; Mislove, A. Taking a long look at QUIC: An approach for rigorous evaluation of rapidly evolving transport protocols. *Commun. ACM* **2019**, *62*, 86–94. [CrossRef]
53. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. 2018, pp. 1–160. Available online: https://www.rfc-editor.org/info/rfc8446 (accessed on 11 January 2024).
54. Thomson, M.; Turner, S. Using TLS to Secure QUIC. Mozilla and sn3rd, January 14, 2021, Internet-Draft: Draft-ietf-quic-tls-34. Available online: https://datatracker.ietf.org/doc/html/draft-ietf-quic-tls-34 (accessed on 11 January 2024).
55. Viernickel, T.; Froemmgen, A.; Rizk, A.; Koldehofe, B.; Steinmetz, R. Multipath QUIC: A deployable multipath transport protocol. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–7.
56. Firmansyah, M.H.; Jung, J.-H.; Koh, S.-J. Proxy-Based Adaptive Transmission of MP-QUIC in Internet-of-Things Environment. *Electronics* **2021**, *10*, 2175.
57. Luglio, M.; Quadrini, M.; Roseti, C.; Zampognaro, F.; Romano, S.P. A QUIC-based proxy architecture for an efficient hybrid backhaul transport. In Proceedings of the 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 24–27 February 2020; pp. 144–146.
58. Li, B.; Wang, C.; Xu, Y.; Ma, Z. An MMT based heterogeneous multimedia system using QUIC. In Proceedings of the 2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT), Dalian, China, 22–23 October 2016; pp. 129–133.
59. Tong, V.; Tran, H.A.; Souihi, S.; Mellouk, A. A novel QUIC traffic classifier based on convolutional neural networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
60. Basyoni, L.; Erbad, A.; Alsabah, M.; Fetais, N.; Mohamed, A.; Guizani, M. QuicTor: Enhancing Tor for real-time communication using QUIC transport protocol. *IEEE Access* **2021**, *9*, 28769–28784. [CrossRef]
61. Shreedhar, T.; Panda, R.; Podanev, S.; Bajpai, V. Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads. *IEEE Trans. Netw. Serv. Manag.* **2021**, *19*, 1366–1381. [CrossRef]
62. Feng, B.; Zhang, H.; Zhou, H.; Yu, S. Locator/identifier split networking: A promising future Internet architecture. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2927–2948.
63. Mazhar, M.H.; Shafiq, Z. Real-time video quality of experience monitoring for HTTPS and QUIC. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1331–1339.
64. Cook, S.; Mathieu, B.; Truong, P.; Hamchaoui, I. QUIC: Better for what and for whom? In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.

65. Khan, S.; Luo, F.; Zhang, Z.; Rahim, M.A.; Ahmad, M.; Wu, K. Survey on issues and recent advances in vehicular public-key infrastructure (VPKI). *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1574–1601. [CrossRef]
66. De Coninck, Q.; Bonaventure, O. Multipath QUIC: Design and evaluation. In Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, Incheon, Republic of Korea, 12–15 December 2017; pp. 160–166.
67. Kharat, P.K.; Rege, A.; Goel, A.; Kulkarni, M. QUIC protocol performance in wireless networks. In Proceedings of the 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 3–5 April 2018; pp. 472–476.