Communication Multiplexing of Server with QUIC and SDN in Multihomed Networks

Tomoya Kawana
Department of Computer and Information Sciences
Tokyo University of Agriculture and Technology
Tokyo Japan
s228160z@st.go.tuat.ac.jp

Rei Nakagawa
Department of Electrical Engineering and Computer Science
Tokyo University of Agriculture and Technology
Tokyo Japan
rnakagawa@go.tuat.ac.jp

Nariyoshi Yamai
Department of Electrical Engineering and Computer Science
Tokyo University of Agriculture and Technology
Tokyo Japan
nyamai@cc.tuat.ac.jp

***Abstract*—When mobile devices equipped with multiple wireless network interfaces access the Internet through Wi-Fi, 4G, and 5G, external factors such as radio waves can lead to an increased packet loss rate, resulting in a slowdown in communication speed. To address this issue, two solutions exist: BBR, a congestion control algorithm designed to reduce the impact of packet loss and bicast in multihomed networks. Bicast in multihomed networks leverages multiple networks for communication, allowing simultaneous transmission of the same packet across multiple networks to reduce packet loss rates and prevent communication speed reduction. This paper introduces a novel network architecture that uses QUIC and SDN, both of which can be implemented in the userland. This network architecture enables the realization of a multihomed network independent of the operating system, and the congestion control algorithm can be easily configured. Additionally, because QUIC lacks bicast capabilities for multihomed networks, this deficiency is addressed through SDN implementation. By using the proposed multihomed network, we measure communication speeds for unicast and bicast packets from the server when varying the packet loss rate. The results indicate that the communication speed could be maintained even with a 15% packet loss rate for unicast communication and 32% for bicast communication.**

***Index Terms*—QUIC, SDN, Multihomed Network, Bicast, BBR**

I. Introduction

The number of mobile devices has been increasing significantly according to the Cisco Annual Report [1]. These mobile devices generally communicate using wireless networks, such as 4G, 5G, and Wi-Fi. Wireless networks lead to communication instability compared with wired networks. One of the factors causing this instability is packet loss due to external influences such as radio interference [2]. Packet loss results in lowering communication speed. Hence, mitigating the reduction of communication speed due to packet loss in wireless networks is essential.

TCP, which is currently used in the transport layer protocol, has a problem that the communication speed is significantly reduced in a network where packet loss frequently occurs. Specifically, CUBIC [3], the congestion control algorithm for TCP in Linux and Windows, tends to perform extremely poorly in lossy networks. CUBIC reduces communication speed to 1/10 when the packet loss rate is 0.1% [4]. CUBIC is a loss-based congestion control algorithm, thus the communication speed reduction whenever packet loss occurs. Meanwhile, BBR [4] determines the communication speed based on the RTT and bandwidth, so packet loss have relatively less impact on the communication speed reduction. Therefore, configuring the congestion control algorithm to BBR, which is resistant to packet loss, can prevent communication speed reduction due to packet loss. However, it is hard to introduce new congestion control algorithms because it requires kernel updates. On the other hand, using bicast, where the same packet is transmitted through multiple paths simultaneously in multihomed networks, can reduce packet loss rate and stabilize communication in lossy network. TCP can perform bicast by using Multipath TCP (MPTCP) [5], which is designed to support multihomed networks. To use MPTCP, it must be implemented on both the client and server sides. However, MPTCP cannot be used in all communication devices because it needs to be implemented in the OS.

In contrast, QUIC [6] has been attracting attention as a general-purpose network transport layer protocol. QUIC is a connection-oriented transport layer protocol like TCP, developed based on UDP, and enables flexible control in the userland. Thus QUIC is easy to configure a congestion control algorithm due to flexibility in userland development. Additionally, QUIC does not use IP addresses to identify connections, so QUIC can receive packets from different end-to-end routes as packets on a single connection.

To prevent communication speed reduction in lossy networks, this study introduces a novel network architecture that uses QUIC to enable flexible software-defined control through OpenFlow switches. OpenFlow, the protocol that enables SDN, controls and changes the routing of multihomed networks in QUIC. Additionally, a QUIC proxy is implemented to seamlessly convert communication between TCP and QUIC for both the client and proxy server. This enables communication via QUIC without requiring modifications to existing applications. We implement the proposed network architecture on real devices. Additionally, we evaluate communication speed using QUIC proxy across various packet loss rates and set congestion control algorithm to BBR. Additionally, we assess communication speed employing the proposed architecture for bicast full of server communication.

The contributions of this paper include:

- Introducing a novel network architecture that mitigates communication speed reduction due to packet loss under flexible congestion control algorithms and multiplexing of communications using QUIC.
- Implementing QUIC proxy to consider compatibility with existing applications that do not support QUIC.
- Our proposed architecture can ensure stable communication, even in scenarios where packet loss is up to 15% for unicast and 32% for bicast in wireless networks.

II. FUNDAMENTAL TECHNOLOGIES

This section describes the basics of QUIC, BBR, and SDN.

A. QUIC and TCP

QUIC is designed as a secure general-purpose transport protocol over UDP to improve web performance. It is designed to address many of the limitations of TCP, including reducing latency, enhancing security, and addressing the head-of-line blocking problem. The notable advantage of using QUIC compared to TCP include:

- Support for IP address changes
  When identifying connections, TCP uses a 5-tuple. Therefore, if the IP address changes during the connection, this connection can not be continued. On the other hand, QUIC uses Connection ID (CID) to uniquely identify communications. This feature enables seamless communication persistence, even in cases where IP addresses change, as long as the CID remains consistent.
- Encryption and Authentication
  TLS is used in conjunction with TCP because TCP has no encryption and authentication function. On the other hand, QUIC encrypts communications and provides authentication. Therefore, communication can be initiated securely with just a QUIC handshake. The authentication function can also be used to restrict connections to specific users or devices.
- Operation in the userland
  Since TCP is implemented in the kernel, it is not easy to update functions. On the other hand, QUIC is implemented in the userland, making it independent of the OS kernel. This userland implementation of QUIC offers several advantages, including the ability to utilize new functionality without requiring kernel changes and simplifying portability to other devices. Furthermore, its versatility allows it to be employed across various communication devices that support UDP.

However, QUIC has the disadvantage of a low adoption rate compared to TCP because QUIC is a relatively new transport layer protocol.

B. BBR

The loss-based congestion control algorithm, such as New Reno and CUBIC, considers packet loss as an indicator of congestion. When a packet loss event is detected, these algorithms reduce the window size and try to avoid congestion. Therefore, in a network where packet loss is likely to occur, the window size tends to be small and the communication speed is reduced.

In contrast, BBR sets the sending rate from the round-trip time (RTT) and bandwidth, allowing stable communication up to a packet loss rate of 15% [4]. In addition, BBR outperforms CUBIC in both short and long downloads, especially in networks with packet loss [7] [8]. BBR is a congestion control algorithm designed to optimize the performance of network connections, particularly in scenarios where there is limited available bandwidth or high latency. BBR focuses on estimating the bandwidth of the bottleneck, which typically refers to the link with the narrowest capacity along the data path. This estimation is achieved through real-time probing of the network to determine available bandwidth. Additionally, BBR takes into account the round-trip propagation time, which is the minimum round-trip time probed as an estimate during a specific time window. By actively measuring and factoring in the RTT, BBR dynamically adjusts its sending rate to prevent congestion and minimize queuing delay. The goal of BBR is to adapt its sending rate to maximize link utilization without inducing congestion and excessive queuing at bottleneck points.

C. SDN

SDN is a technology that flexibly manages and configures network settings and functionality through software. In contrast to traditional networks, which are hardware-centric with network devices, such as switches and routers, having their own logic and control mechanisms, SDN separates the control plane from the data plane and centralizes control plane management. The key advantages of SDN are enhanced network flexibility, efficiency improvements, and traffic optimization.

OpenFlow is a protocol used in SDN to separate the control of network devices from the actual data transmission process. In the OpenFlow, network devices are responsible solely for data forwarding, while the management of routing, flow control, and related functions is centralized within the OpenFlow controller. OpenFlow switches operate by processing incoming packets based on flow tables that dictate packet handling. These flow tables can define packet behaviors according to attributes like IP and MAC addresses. Consequently, OpenFlow switches offer flexibility in configuring routing control.

III. NETWORK ARCHITECTURE FOR COMMUNICATION STABILIZATION WITH QUIC AND SDN

TCP, which uses CUBIC as its congestion control algorithm, has the problem of significantly reducing communication

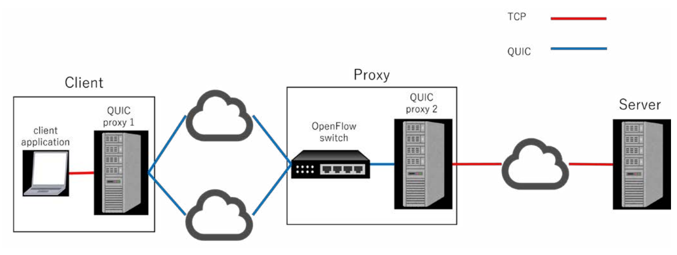

Fig. 1. Proposed network architecture

speed in networks where packet loss occurs. To address this problem, we implement a network that uses BBR resistant to packet loss as a congestion control algorithm and performs bicast in a multihomed network.

We propose the network architecture used for data transmission from a server to a client. The proposed network architecture is shown in Figure 1. In this study, we use the capabilities of QUIC and OpenFlow to establish a multihomed network environment. QUIC, being implemented in the userland, is independent of the OS. It ensures uninterrupted communication even when IP addresses change, enabling simultaneous use of multiple networks. OpenFlow facilitates control over communication routes through the configuration of flow tables. This combination of QUIC and OpenFlow can realize multihomed networks. The client and proxy server incorporate a QUIC proxy, converting seamlessly TCP to QUIC and QUIC to TCP. Communication between the client and QUIC proxy 1, QUIC proxy 2 and the server communicate via TCP, while QUIC proxy 1 and QUIC proxy 2 communicate via QUIC. BBR is configured as a congestion control algorithm for QUIC.

A. *QUIC proxy*

Currently, only 26.8% of HTTP servers support QUIC, and many servers do not support this protocol [9]. To use the proposed network without necessitating modifications to existing applications, we implement a QUIC proxy in a client and proxy server.

Figure 2 presents a ladder diagram that depicts the flow of the HTTP transaction as it passes through a QUIC proxy. HTTP transactions are executed as follows. Initially, a handshake takes place between the client and QUIC proxy 1, QUIC proxy 1 and QUIC proxy 2, and QUIC proxy 2 and the server. The QUIC proxy then converts the TCP data received from the client and server into QUIC STREAM frames and sends them to the QUIC proxy to which it is connected. Similarly, the QUIC proxy converts the received QUIC STREAM frames into TCP and sends them to the client and server. ACKs are used in each connection to confirm the successful reception of data. This mechanism allows the QUIC proxy to promptly retransmit packets in case of packet loss in the wireless network using QUIC. The process of packet retransmission in QUIC is illustrated in Figure 3.

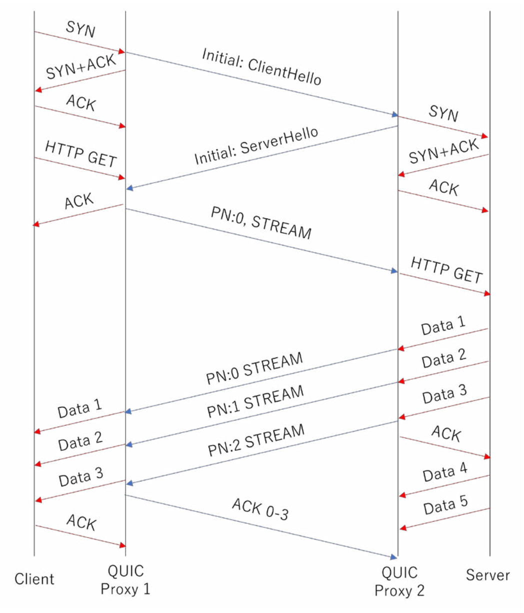

Fig. 2. HTTP transaction over QUIC proxy (PN: packet number)

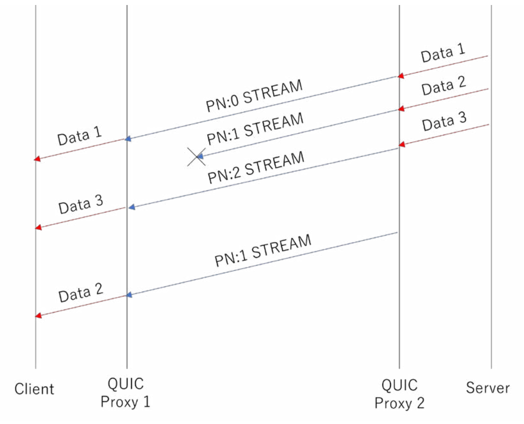

Fig. 3. Packet retransmission in QUIC (PN: packet number)

B. *OpenFlow for bicast*

OpenFlow enabled bicast, which simultaneously transmits a single packet across multiple networks. Bicast offers a significant advantage over unicast, particularly in terms of lowering packet loss rates. This is because a packet loss in bicast occurs when packet losses simultaneously happen in multiple networks. In our study, we have chosen to implement bicast only for data originating from the server side. The client must receive all packets to download data from the server. Therefore, a packet loss causes the communication speed to decrease because of the overhead of retransmission. Typically, the data from the client side is sent only for ACK. These data are less sensitive to packet loss and have minimal effects on communication speed, even in the presence of some degree of packet loss. In addition, if bicast of the data from the client using QUIC is enabled, OpenFlow must be implemented in the client. However, it is hard to implement OpenFlow for all

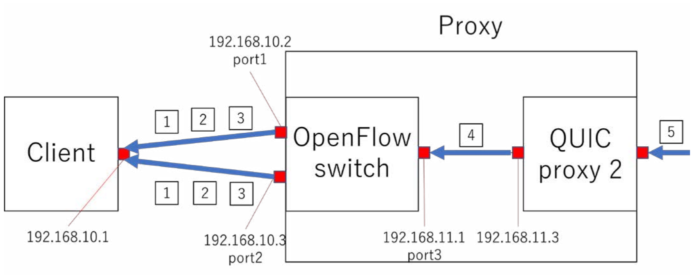

Fig. 4. Server communication using OpenFlow.

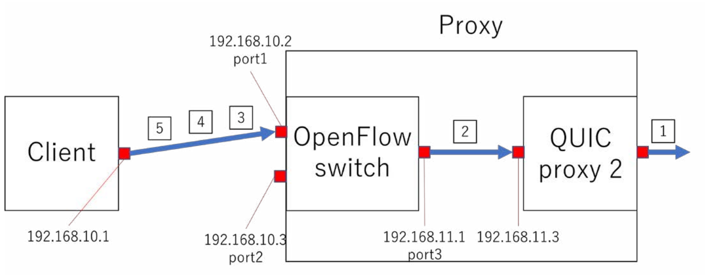

Fig. 5. Client communication using OpenFlow.

clients. Thus, we perform bicast only the data from the server side by using OpenFlow.

Figures 4 and 5 show server and client communications using OpenFlow, respectively, and Tables I and II show flow and group tables, respectively. The OpenFlow switch duplicates QUIC packets from the server and sends them to the client through two different routes. However, a conventional flow table typically allows only one port to transmit packets. To address this issue, we use a group table in the flow table actions to perform bicast. The group table serves as a mechanism that enables the collective definition of multiple ports for processing, thereby facilitating the use of diverse routes. To ensure that the packets are sent through all selected ports, we configure the "all" type in the group table settings. In practice, the group table is used by specifying the ID of the group table as the output destination in the Instruction of the flow table. Consequently, packets originating from the servers are sent through all selected ports. Notably, if these packets match the defined match fields in Table I, their source IP address, source MAC address, and destination MAC address are dynamically altered to align with the designated routing requirements. Thus, bicast is performed by configuring the OpenFlow flow table and group table.

TABLE I
FLOW TABLE OF OPENFLOW IN FIGURE 4 AND 5

Match fields	Instructions
dst_ip=192.168.10.1	processing group table 1
src_ip=192.168.10.1	send to QUIC proxy 2

TABLE II
GROUP TABLE OF OPENFLOW IN FIGURE 4 AND 5

Group ID	Type	Action Bucket
1	all	send to client via port 1
		send to client via port 2

TABLE III
BICAST FLOW TABLE IN FIGURE 6

Match fields	Instructions
dst_ip=192.168.11.3	Output:group1
src_ip=192.168.11.3	dst_MAC=QUIC proxy 2 Output: 3

IV. EVALUATION

A. Test environment

In our experimental network, as shown in Figure 6, we implement the multihomed network using QUIC. Hardware specifications of the client and proxy are shown in Table VI, and Raspberry pi 4 Model B is used as the server. The flow and group table of the OpenFlow switch for bicast are shown in Table III and IV, respectively, and the flow table for unicast is shown in Table V. The network configuration during these experiments involved the following components and configurations:

- The client application and QUIC proxy 1 are implemented on a single terminal as the client. The communication between the client application and QUIC Proxy 1 uses a loopback address.
- The OpenFlow switch, OpenFlow Controller, and QUIC proxy 2 are implemented on a single terminal as the proxy. In addition, the OpenFlow switch and QUIC proxy 2 run in different namespaces. The namespace is a mechanism for creating virtual nodes and using virtual NICs to communicate with a host.
- Open vSwitch is used as the OpenFlow switch, and Ryu is used as the OpenFlow controller. The OpenFlow controller sends an ARP response to the ARP request received from the client.
- In the implementation of QUIC, picoquic written in the C language is used. This is because picoquic can use BBR as a congestion control algorithm.

In addition, the network of TCP, as shown in Figure 7, is compared with the performance of QUIC. Linux implements MPTCP with multihomed support for TCP. However, MPTCP implemented in Linux does not have a scheduler to bicast only the server side, so TCP also performs bicast by using the OpenFlow switch on the server side. Also, unlike QUIC, TCP requires the same IP address for incoming packets, so the OpenFlow switch is used on the client side to change the source IP address of packets from the server. For TCP, CUBIC is configured as the congestion control algorithm.

B. Measurement of the communication speed

In this experimental study, we compare communication speeds between unicast and bicast packets on the server side in

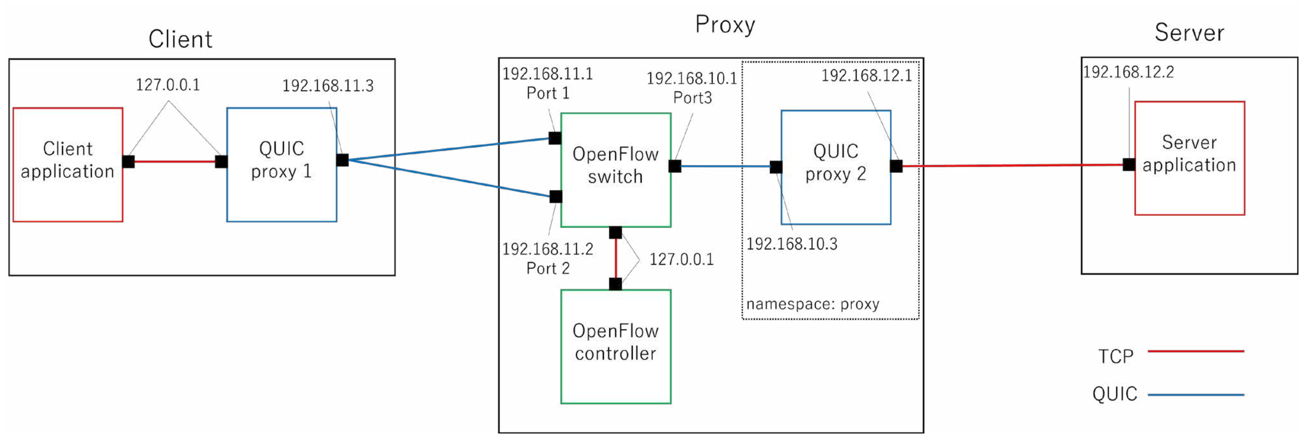

Fig. 6. Experimental setup with QUIC and OpenFlow.

TABLE IV
BICAST GROUP TABLE IN FIGURE 6

Group ID	Type	Action Bucket

1	all	src_ip=192.168.11.1
		dst_MAC=QUIC proxy 1
		Output: 1

		src_ip=192.168.11.2
		dst_MAC=QUIC proxy 1
		Output: 2

TABLE V
UNICAST FLOW TABLE IN FIGURE 6

Match fields	Instructions
dst_ip=192.168.11.3	dst_MAC=QUIC proxy 1 Output: 1
src_ip=192.168.11.3	dst_MAC=QUIC proxy 2 Output: 3

a multihomed network, using QUIC and TCP. We set specific network conditions using the Linux Traffic Control (TC) so that we simulate a wireless network between the client and proxy, We fix the RTT between the client and proxy to 100 ms to reproduce the actual latency and limit the bandwidth between the client and proxy to 100Mbps. In addition, we set the RTT between the proxy and server to 5 ms. To assess performance, we measure the time required to download a 100 MB file using HTTP communication initiated from the client to the server. We vary the packet loss rate between the client and proxy as a crucial parameter for the analysis.

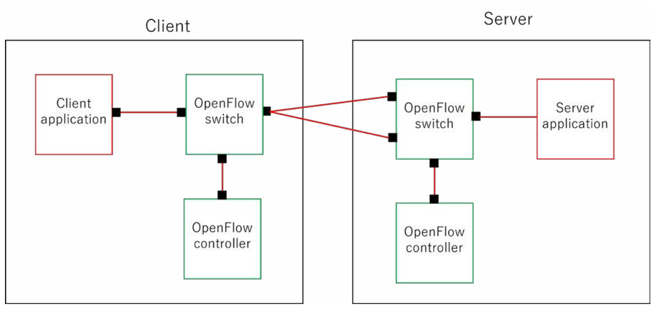

Fig. 7. Experimental setup with TCP and OpenFlow.

TABLE VI
HARDWARE SPECIFICATIONS OF FIGURE 6

	Client	Proxy
CPU	Intel Xeon CPU E31245 @ 3.30 GHz	Intel Core i7-11700 @ 2.50 GHz
Memory	4GB×2 DDR3 1333 MHz	8GB DDR4 3200 MHz
OS	Ubuntu 20.04.6 LTS	Ubuntu 20.04.6 LTS

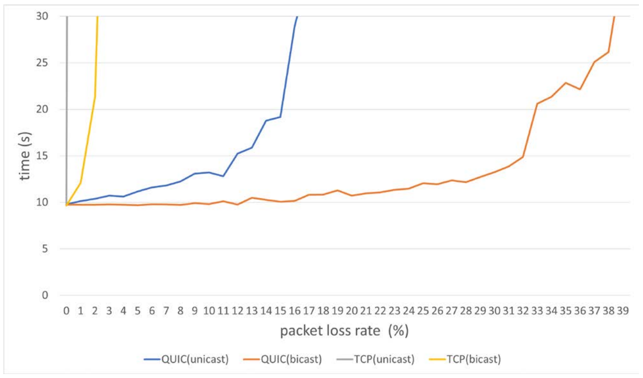

Fig. 8. Unicast and bicast download time in TCP and QUIC.

Figure 8 shows the time taken to transfer 100 MB of data from the server to the client when the packet loss rate between the client and proxy was varied from 0% to 39%.

In TCP, the communication speed decreases to less than 1/3 when the packet loss rate is 1% for unicast and 3% for bicast. In QUIC, unicast communication consistently demonstrates highly reliable performance with stable communication speeds up to a packet loss rate of 11%. However, beyond a packet loss rate of 11%, the communication speed begins to reduce. This decrease becomes particularly pronounced when the packet loss rate exceeds 15%, resulting in a sharp drop in communication speed to approximately half of the rate observed at a 0% packet loss rate. In the context of bicast, our experiments reveal that the communication speed remains stable even under relatively higher packet loss rates compared to unicast. Specifically, we observed stable performance up to a packet loss rate of 32%.

Bicast is effective in ensuring reliable communication. However, in this network, duplicating all server side data places an increased load on the entire network. This load becomes particularly evident when dealing with substantial volumes of data. Large data transfers can significantly amplify the network load, potentially impacting its overall performance. Therefore, for scenarios with a packet loss rate of up to approximately 15%, unicast is the preferred choice. In cases where the packet loss rate exceeds 15%, particularly when faced with higher packet loss rates, bicast is the preferred choice.

V. Conclusion

In this study, we successfully implemented a novel multi-homed network by harnessing the capabilities of QUIC and SDN. By configuring QUIC with the BBR congestion control algorithm, we explored the use of both unicast and bicast in a wireless network prone to packet loss. The results indicate stable communication with up to a 15% packet loss rate for unicast and 32% for bicast. In the future, we would like to develop a scheduler that dynamically switches between unicast and bicast.

References

[1] CISCO, "Cisco Annual Internet Report (2018–2023) White Paper", https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] A. Mahanti, N. Carlsson, C. Williamson, and M. Arlitt, "Ambient Interference Effects in Wi-Fi Networks", 9th International IFIP TC 6 Conference on Networking (NETWORKING), Lecture Notes in Computer Science, vol. 6091 (2010), pp. 160-173

[3] Lisong Xu, Sangtae Ha, Injong Rhee, Vidhi Goel, and Lars Eggert, "CUBIC for Fast and Long-Distance Networks", RFC 9438, August 2023, https://www.rfc-editor.org/info/rfc9438

[4] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson, "BBR: Congestion-Based Congestion Control", ACM Queue, vol.14, no.5, pp.20-53, 2016

[5] A. Ford, C. Rauciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, March 2020, https://www.rfc-editor.org/info/rfc8684

[6] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, May 2021, https://www.rfc-editor.org/info/rfc9000

[7] Aitor Martin, and Naeem Khademi, "On the Suitability of BBR Congestion Control for QUIC over GEO SATCOM Networks", ANRW '22: Proceedings of the Workshop on Applied Networking Research, No.3, pp.1-8, 2022

[8] Yue Wang, Kanglian Zhao, Wenfeng Li, Juan Fraire, Zhili Sun, and Yuan Fang, "Performance Evaluation of QUIC with BBR in Satellite Internet", 2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), pp.195-199, 2018

[9] W3Techs, "Usage statistics of http/3 for websites", https://w3techs.com/technologies/details/ce-http3