Improving Performance of QUIC in WiFi

Improving Performance of QUIC in WiFi

Jawad Manzoor*, Llorenç Cerdà-Alabern†, Ramin Sadre* and Idilio Drago‡
*Université Catholique de Louvain †Universitat Politècnica de Catalunya ‡Politecnico di Torino
{jawad.manzoor,ramin.sadre}@uclouvain.be llorenc@ac.upc.edu idilio.drago@polito.it

***Abstract*—QUIC is a new transport protocol under standardization since 2016. Initially developed by Google as an experiment, the protocol is already deployed in large-scale, thanks to its support in Chromium and Google's servers. In this paper we experimentally analyze the performance of QUIC in WiFi networks. We perform experiments using both a controlled WiFi testbed and a production WiFi mesh network. In particular, we study how QUIC interplays with MAC layer features such as IEEE 802.11 frame aggregation. We show that the current implementation of QUIC in Chromium achieves sub-optimal throughput in wireless networks. Indeed, burstiness in modern WiFi standards may improve network performance, and we show that a *Bursty QUIC* (BQUIC), i.e., a customized version of QUIC that is targeted to increase its burstiness, can achieve better performance in WiFi. BQUIC outperforms the current version of QUIC in WiFi, with throughput gains ranging between 20% to 30%.**

I. Introduction

With the exponential growth in adoption of mobile phones and other smart connected devices, the usage of wireless networks continues to grow. Today, wireless networks are commonplace in every sector of life including homes, offices, restaurants, hospitals and university campuses. In a recent Cisco white paper [1] it is predicted that wireless and mobile device traffic will exceed that of PCs, and comprise more than 63 percent of total IP traffic by 2021. There is a continuous increase in user demand for richer mobile web content and reduced loading time. At the same time, complex applications and web content put a large computational burden on mobile devices, which are in general more resource-constrained than PCs and laptops.

Nonetheless, technological advancements have allowed to overcome these issues and provide better user experience. On one hand, WiFi technologies, which are based on the IEEE 802.11 standards, have undergone an enormous evolution in the recent years [2]. For instance, the 802.11n standard, which is predominant nowadays, allows coding schemes (MCS) that support data rates up to 600 Mbps. It also includes a high throughput enhancement called *frame aggregation*, which consists of combining two or more data frames into a single transmission, thus reducing the fixed overhead associated with each frame transmission. On the other hand, efforts from content providers in developing customized mobile versions of websites and new low-latency transport protocols such as QUIC have contributed to improve user experience. QUIC started as an experimental protocol designed by Google and has emerged as a serious alternative to TCP. In a recent measurement study [3], it was estimated that around 7% of the Internet traffic is QUIC.

In this paper we experimentally analyze the performance of QUIC in WiFi networks. We start from the observation that Chromium's implementation of QUIC (version 39) has sub-optimal performance in WiFi. We then investigate root-causes for the problem, finding that some of the QUIC features in Chromium impair the protocol performance in WiFi networks.

In particular, bursty traffic has been traditionally considered undesirable, since it can lead to longer queuing delays and multiple consecutive losses which are more difficult to recover. This fact has motivated QUIC to be designed with *packet pacing* [4], with the aim to reduce burstiness and, thus, packet losses. While this consequence is generally true in wired networks, we observe that due to the characteristics of the WiFi medium and interactions between transport and MAC protocols, bursty traffic might actually be beneficial in WiFi. One apparent reason for this behavior is frame aggregation. As mentioned above, frame aggregation is a key feature to achieve high throughput in recent 802.11 standards and burstiness increases aggregation opportunities.

Therefore, we implement and evaluate a *Bursty QUIC* (BQUIC) in Chromium, which is a customized version of QUIC targeted to increase traffic burstiness by reducing the transport layer acknowledgment frequency and disabling packet pacing. We show the advantage of BQUIC for two different use cases of WiFi: (i) in home or enterprise wireless local area networks (WLANs), and (ii) wireless mesh networks (WMN) such as Guifi.net, MadMesh, Merkai and Google WiFi. We analyze these use cases by taking measurements in a lab and a production WMN. Our experimental results show that increasing burstiness of QUIC improves its throughput in WiFi, with gains ranging between 20% to 30%.

Our results are a step forward for understanding performance trade-offs in QUIC. They can help in the design of the standard protocol, which would extract better performance from lower layer protocols. Since layer-2 protocols in wired and wireless networks are nowadays significantly different, we believe that the transport protocol can be improved by individually tuning it for both networks.

II. Background and related work

A. 802.11 and frame aggregation

802.11 is a set of IEEE standards that regulate wireless transmission. A recent measurement study [5] with millions of Cisco Meraki access points (APs) shows that around 99% of

the APs use the 802.11n and 802.11ac standards. These standards introduce enhancements to increase data rates. Among them, *frame aggregation* is a simple method to enhance throughput.

The wireless medium has a high overhead, which includes the MAC and PHY headers, acknowledgments (ACK), backoff time and inter-frame spacing. For ACKs and small segments, the overhead in terms of bytes can be higher than the actual payload. The frame aggregation scheme amortizes this overhead and achieves high data throughput by combining multiple data frames into a single transmission unit.

Aggregation in WiFi MAC architecture is supported at two layers. In the first layer multiple MAC service data units (MSDUs) are aggregated into an A-MSDU. In the second layer multiple A-MSDUs are combined to form an aggregated A-AMPDU. A detailed description of these concepts can be found in [6]. Their impact on throughput have also been extensively evaluated [7], [6], [8]. We here evaluate how to profit from the mechanisms to improve QUIC performance.

B. QUIC protocol

QUIC is a user-space transport protocol running on top of UDP. QUIC provides several cross-layer enhancements, covering the weaknesses of TCP for transporting web content. For example, in the case of HTTP/2 running over TLS and TCP, the loss of a single TCP packet blocks all HTTP/2 streams, since a single connection is shared by all streams. QUIC instead is designed to handle the streams, thus eliminating head-of-line blocking delays in case of a packet loss.

QUIC profits from recent advances in TLS, implementing new TLS 1.3 concepts such as zero handshake latency. That is, QUIC is able to reduce latency by reusing credentials of known servers on repeated connections. QUIC also provides an improved congestion controller, better RTT estimation, and a better loss recovery mechanism than TCP.

QUIC is still under development, thus its features and operations are not completely standardized yet. To understand its internal workings, we have studied the QUIC source code in the open-source Chromium project [9]. Our experiments in this paper have been performed with QUIC version 39.

Two aspects of QUIC implementation particularly influence how the protocol interacts with 802.11 frame aggregation: (i) acknowledgment modes, and (ii) packet pacing.

1) QUIC acknowledgment modes: Chromium's implementation of QUIC includes two acknowledgment modes:

- **TCP_ACKING**: This mode is similar to TCP delayed acknowledgment, in which an ACK is generated for every 2 received packets in accordance with RFC 1122. This was the default mode of QUIC in Chromium at the time of writing.
- **ACK_DECIMATION**: In this mode acknowledgments are delayed up to a maximum of 10 packets (unless `unlimited_decimation` is enabled) and a cumulative ACK is generated. The maximum duration for which the ACK can be delayed is 25 ms and the actual `delay_time` is calculated on the fly as the minimum between `max_delay_time` and one quarter of minimum RTT observed during the session. The `ACK_DECIMATION` is only considered after at least 100 packets have been received to avoid interfering with slow start. This mode was disabled in Chromium at the time of writing. In either mode, if an out-of-order packet or a previously missing packet is received, the ACK is sent without any delay to inform the sender immediately about it.

2) Packet pacing: Chromium's implementation of QUIC includes a packet pacing mechanism that aims to reduce sending bursts of packets by introducing delay between consecutive packets. Reducing traffic burstiness is known to prevent congestion and, as a consequence, reduce the undesirable effects of packet loss. However, it may also hinder performance in high-speed networks with low loss rates.

C. Transport protocols enhancements for WiFi

To the best of our knowledge, no previous work has evaluated the performance of QUIC in WiFi by exploring its interactions with the wireless medium and 802.11 enhancements such as frame aggregation. However, given the popularity of TCP, it is not a surprise that previous works have targeted similar problems in TCP deployments.

Considering the role of TCP acknowledgments for the protocol reliability, reducing the acknowledgement frequency and performing delayed cumulative acknowledgements may provide benefits in wireless networks. Many works propose to reduce TCP acknowledgement frequency in wireless networks. Altman el. al [10] investigated the impact of increasing the TCP delayed acknowledgement mechanism to more than two segments as recommended by RFC 1122. Singh et. al [11] propose TCP with adaptive delayed acknowledgement, which aims to reduce the number of ACKs to one per congestion window. Oliveira el. al [12] propose Dynamic Adaptive Acknowledgement where the delay window is adjusted according to the channel condition. In [13], the same authors provide an improved delaying window strategy for robustness against losses.

These works agree that a key factor affecting TCP performance in wireless networks is the contention and collision between ACK and data packets. Reducing the number of ACKs saves wireless resources and reduces interferences with other packets. Moreover, lowering the ACK frequency increases burstiness of traffic as the sender releases a micro burst of packets after receiving the cumulative ACK. This behavior may reduce the inter-packet time increasing the opportunities for frame aggregation at the 802.11 MAC layer.

The closest work to ours is [14], which provides a deep view on QUIC performance. The authors show, for example, that since QUIC runs on user-space it incurs performance penalties particularly for mobile devices that are usually constrained by processing power. We extend the knowledge about QUIC performance here, showing how the protocol interacts with features of lower-layer protocols in WiFi.

III. Methodology

We now describe our methodology, covering our customization to Chromium's QUIC implementation (Sec. III-A), our

test environment (Sec. III-B and Sec. III-C) and the performance metrics used in the experiments (Sec. III-D). The results presented in the evaluation have been obtained parsing traces captured with tcpdump.

A. Bursty QUIC

Our goal is to study QUIC's behavior and improve its performance in WiFi networks. Based on the observations in the previous section, we believe that the (non)burstiness of QUIC traffic cannot fully exploit frame aggregation in WiFi MAC layer. Therefore, we tune QUIC to produce bursty traffic: We have compiled a version of Chromium with `ACK_DECIMATION` as default acknowledgment mode and without packet pacing. Disabling packet pacing is important since it can neutralize the effect of burstiness created by `ACK_DECIMATION`. These two features are not controllable from the browser configuration — we had to study the source code and make required modifications. We call this tuned version *Bursty QUIC* (BQUIC).

We focus only on WiFi and do not evaluate the scenarios with wired or hybrid connectivity between client and server. There are concerns about the impact of high burstiness on packet drops and queuing delays in wired networks, particularly in long Internet paths. However, we will show that the performance gains in the WiFi environment are high and can potentially overshadow other effects. Moreover, in real-world scenarios service providers are increasingly deploying caches and CDN nodes closer to end-users [15]. Thus, the scenario tested in the following is already popular and tends to become widespread as more servers are deployed closer to users.

In the following experiments we place the server geographically close to the clients i.e., at WiFi access point or mesh network gateway, with an average RTT in the range of 6 ms to 10 ms between the clients and server. Performing experiments on Internet-wide scale are left for future work. We carry out experiments in two testbeds (i) a controlled lab and (ii) a real production mesh network.

B. Lab testbed

Our lab testbed consists of a client connected to a WiFi router, all using 802.11n. The WiFi router in our testbed is connected to a server through a Gigabit Ethernet connection. We use two devices as clients (i) an Android smartphone with ARM Cortex A-57 quad-core CPU and 2 GB RAM running Android 6.0 and (ii) a Raspberry PI 3 (RPi) having quad-core ARM Cortex-A53 CPU and 1 GB RAM running Debian 9. The server in our testbed has a quad-core Intel Core i5-3470 CPU and 8 GB RAM running Ubuntu 16.04. We cross-compile chromium browser with QUIC and BQUIC for ARM and Android and deploy it on the respective clients.

C. Wireless community network

Our second testbed is a production wireless community network deployed in a neighborhood of the city of Barcelona (Spain) called Sants [16]. The network was started in 2009 and in 2012 was joined by nodes installed at *Universitat Politècnica de Catalunya* (UPC) within the EU CONFINE project [17]. The network is operative since 2009. The nodes use the linux/openwrt [18] based distribution provided by the Quick Mesh Project (QMP) [19], which runs the BMX6 mesh routing protocol [20]. From now on we will refer to this network as *QMPSU*. QMPSU is part of a larger community network started in 2004, which has more than 30.000 operative nodes called Guifi.net [21]. At the time of writing QMPSU has around 80 active nodes. Fig. 1 shows the geographic location of active nodes and links, using distinct colors to represent wireless links configured with different channels. In QMPSU there are 2 gateways that connect QMPSU to the rest of Guifi.net and the Internet.

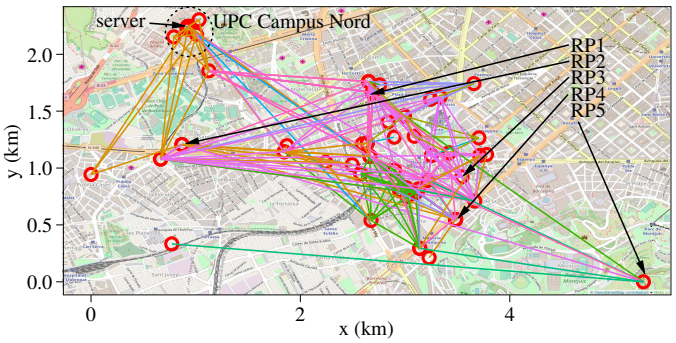

Figure 1: QMPSU geographical topology. Colors indicate links configured in the same WiFi channel.

QMPSU is 802.11an-based and the most common hardware is the Ubiquiti NanoStation M5, equipped with a sectorial antenna and running QMP firmware. There are also a number of point-to-point links using Ubiquiti parabolic antennas running the original manufacturer firmware. QMPSU also has a live monitoring web page updated hourly. A detailed description of QMPSU can be found in [22], and a live monitoring page updated hourly can be accessed on-line [23].

QMPSU has been deployed by its own users. Its unplanned spread out using heterogeneous WiFi devices in an urban area has produced a high diversity on the quality of the links. Thus, it offers a very realistic testbed to evaluate the performance of QUIC under a variety of conditions.

We deploy five RPi clients attached using the Ethernet port to the premises of different volunteers across the QMPSU network. Moreover, we set up a server in one of the gateways of QMPSU to the Internet. Nodes are marked as RP or server respectively in Fig. 1.

The server has an Intel dual-core CPU and 8 GB RAM, running Ubuntu 16.04. The hardware specifications of the RPi are similar to the smartphone used in the Lab testbed, and we will show later that lab results with smartphone and RPi are quite similar. For this reason we only use RPis for experiments in QMPSU for convenience of deployment and maintenance. Tab. I shows the number of wireless hops from the clients to the gateway (W-hops). Note that many of these hops use different frequencies and thus are not interfering with each other.

Table I: Characteristics of the client locations.

RP	Name	W-hops
$RP1$	BCNevaristoarnus5Rd3-BPi	4
$RP2$	GS-BCNpisuerga17Rd1	3
$RP3$	GS26gener10-8710	3
$RP4$	GSgV-rb-dce0	1
$RP5$	BCNJardiBotanicSants186-ba35	5

Table II: Statistics of cloned web pages with the number of objects of various file types.

Website	HTML	CSS	JS	Image	Other	Total	Size (kB)
Google	2	1	3	5	1	12	56
Live	2	2	2	2	0	8	262
Twitter	6	1	4	2	3	16	421
Wikipedia	1	1	2	20	1	25	441
Reddit	4	2	5	26	2	39	470
Yahoo	16	13	5	48	4	86	839
Ebay	4	1	6	3	14	28	985
Instagram	3	1	7	25	1	37	1 409
YouTube	8	3	5	113	20	149	2 911
Facebook	1	1	8	123	1	134	3 560
Amazon	5	2	14	41	2	64	3 723

D. Measuring performance

We have selected 10 websites from Alexa's top 100 list, downloaded their landing pages and other publicly available pages and hosted them on our servers. The selected websites are a mix of social networks, online shopping, news and search engines. The main characteristics of the cloned pages are summarized in Tab. II.

We load these pages from the clients using the default Chromium QUIC implementation and BQUIC. To automate the page loading we use *Chrome-HAR-capturer*[1] to connect to remote clients in the lab or WMN and repeatedly load the pages multiple times while capturing traffic at both client and server sides.

We parse the HAR file and the captured traffic to calculate various metrics such as the page load time (PLT), throughput, and packet inter-arrival time over 30 runs. We also analyze data segments and ACK packets. We compute throughput by dividing the amount of bits sent in the UDP payloads of the QUIC connections over the time of the transfer, ignoring connection establishment time. We have also instrumented the web server to log the CWND size on every acknowledgment.

To emulate bulk file transfers, we have created a synthetic web page with a large image of 10 MB that we have downloaded in 100 runs over 10 days from the lab nodes and the mesh nodes. We calculate the mean throughput and 95% confidence interval for all runs. We also compute the relative improvement achieved by BQUIC, referred to as *gain*.

IV. Evaluation

A. Bulk transfer throughput

Tab. III shows the mean values of the measured throughput and end-to-end % loss obtained by downloading our 10 MB synthetic web page during the 100 runs. The losses have been computed by comparing the identification field of the IP header of transmitted and received datagrams. We have computed the 95% confidence intervals for throughput, and they are small in all cases (less than 10%). In the lab testbed, we can see that the RPi achieves higher throughput than the smartphone. This is mainly because the antenna gain of the RPi is higher than in the smartphone, and thus, the network card can use MCS with higher bitrates during the transfer. However, the throughput gain of BQUIC over QUIC is similar for both devices (26% and 23% in the smartphone and the RPi, respectively).

Table III: Performance comparison of QUIC and BQUIC

		Throughput (Mbps)			Loss rate (%)	
	Device	QUIC	BQUIC	Gain	QUIC	BQUIC
Lab	Android	34.8	43.8	26%	-	-
	RPi	42.38	52.3	23%	-	-
Mesh network	RP1	8.2	10.6	29%	0.63	0.8
	RP2	1.97	2.54	28%	0.79	1.23
	RP3	12.8	15.5	20%	0.36	0.4
	RP4	20.9	27.2	30%	0.01	0.01
	RP5	18.6	24.5	31%	0.05	0.06

Regarding the losses measured at the transport layer, Tab. III shows that they are negligible. This is normal on a WiFi link of an acceptable quality, since 802.11 retransmits lost unicast frames multiple times before abandoning its transmission. Indeed, the worst connected device (RP2) has a loss of only 0.79% in QUIC and 1.23% in BQUIC. We can see that the loss rate slightly increases in BQUIC, but it is negligible. Moreover, the high throughput gain achieved in BQUIC surpasses the negative effects.

Notice that there are significant differences between the client in terms of delays, number of hops and link capacities, which is expected as it is a production network. Despite these differences and the large variations of measured throughput between mesh nodes (1.97 Mbps for RP2 and 20.9 Mbps for RP4 with QUIC) we observe significant performance improvements (between 20% and 31%) in all cases. Since there are too many factors such as the antenna hardware, firmware, wireless link conditions etc. for each node, investigating the low level details to find the root cause of the observed differences is out of scope of this paper. Our main objective is to experimentally show that burstiness increases performance of QUIC in WiFi.

B. Web page load time

In order to see the impact of BQUIC upon different types of web browsing we perform experiments using the cloned websites. RP5 is used as client. Fig. 2 shows the mean PLT for various cloned web pages which are summarized in Tab. II. Using BQUIC we observe a decrease in PLT for all websites ranging from 5% for small web pages such as Google and Live up to 25% for large web pages such as Amazon and Facebook. We can see that the larger the page is, the larger is the reduction of the PLT. This is an expected result, since

[1]https://github.com/cyrus-and/chrome-har-capturer

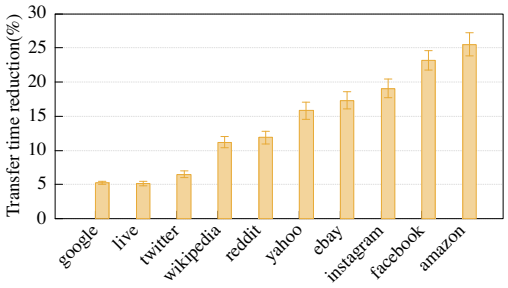

Figure 2: Relative reduction in PLT achieved for various websites by using BQUIC vs QUIC

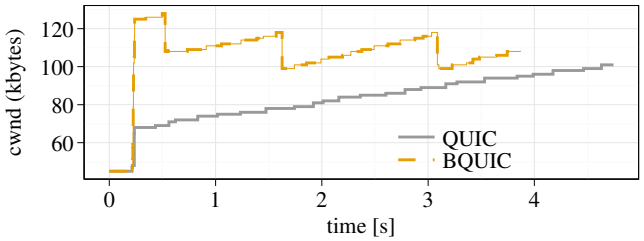

Figure 3: CWND size comparison

the connection establishment, which includes the exchange of certificates, has a larger relative overhead for small web pages.

C. Detailed analysis

To better understand the difference in behavior of QUIC and BQUIC, we now perform a detailed analysis for *one* of the experimental runs from Section IV-A with RP5 node. We observe similar trends for the other nodes, albeit to different extent.

1) CWND size: We instrument the web server to log the size of the congestion window upon each acknowledgement. Fig. 3 shows the CWND size of standard QUIC and BQUIC. We can see that QUIC exits from slow start phase much earlier than BQUIC and thus achieves lower throughput. In QUIC the slow start phase exits when increasing delay is detected. The detection algorithm is called on every new ACK frame and a new RTT measurement is performed. If the minimum delay of the first few packets of the current burst exceeds the minimum delay during the session by a certain threshold, the slow start phase exits. The early exit from slow start in QUIC is conceivably due to packet pacing which reduces aggregation opportunities and allows only a few packets to be transmitted together. The next packets get transmitted in separate unit after gaining access to the wireless medium which injects extra delay. The increased delay is detected by the algorithm and it exits slow start. In BQUIC there is no packet pacing and the inter packet time is much smaller which allows a large number of consecutive packets to be aggregated and transmitted as part of a single unit. Therefore the CWND increases to a much larger value before exiting slow start.

2) Sequence-Acknowledgement analysis: Fig. 4 shows the ACK reception (blue vertical bar) and packet transmission (yellow circle) at the sender side during an interval of 20 ms.

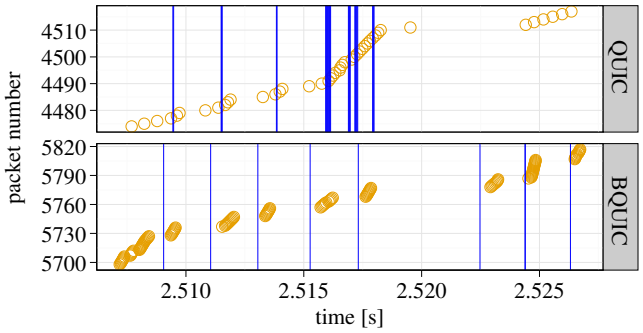

Figure 4: Transmission of data packets (circles) and reception of ACKs (vertical bars) at the sender side

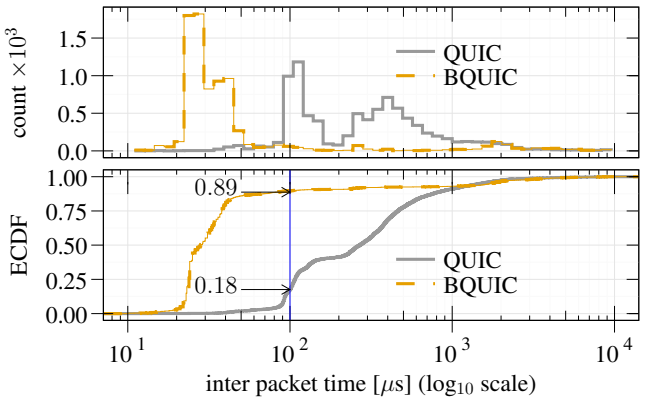

Figure 5: Inter packet time histogram and ECDF

The effect of packet pacing in QUIC can be observed in the upper sub-figure where segments are mostly evenly spaced from one another. In the middle of the figure many ACKs are received together and as a consequence many segments are transmitted by the sender, which can be observed by a steeper slope. The bottom sub-figure represents BQUIC. The figure shows much less ACKs in the interval, due to `ACK_DECIMATION`. Furthermore, since a large window is acknowledged by each ACK and pacing is disabled, a burst of packets is released shortly after an ACK is received.

3) Inter packet time: We measure the inter packet time (IPT) at the server side to get a better insight into the different acknowledgement strategies. Fig. 5 shows the IPT histogram (upper sub-figure) and empirical cumulative distribution function, ECDF (lower sub-figure). We can see that in BQUIC 89% packets are sent with an IPT lower than 100μs, while in QUIC this value is only 18%. The histogram shows two peaks in QUIC around 100 μs and 500 μs due to different pacing rates used by QUIC during this execution. The pacing rate is decided by QUIC on the fly depending on the link conditions such as bandwidth, RTT etc., and varies between different nodes in the mesh and even different runs using the same node. On the other hand, BQUIC IPT is very small and is concentrated around 30 μs.

4) Throughput: Fig. 6 shows the throughput of QUIC and BQUIC computed by averaging over intervals of 50 ms during the bulk transfer time (recall that we compute the throughput

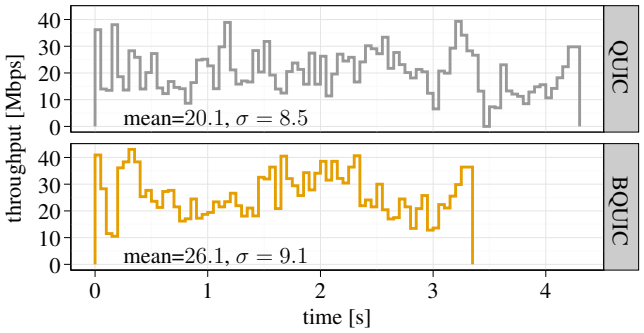

Figure 6: Throughput

trimming out the connection establishment time). As shown in the figure, the overall throughput increases from 20.1 Mbps in QUIC, to 26.1 Mbps in BQUIC (30% gain). Note that, despite BQUIC being more bursty than QUIC at packet level, as shown before, Fig. 6 depicts similar variations of throughput at larger time scale. Indeed, the standard deviation of the throughput measured at 50 ms intervals increases only from 8.5 in QUIC to 9.1 in BQUIC (7%).

Takeaway: Increasing burstiness of QUIC in WiFi provides significant gain in throughput without introducing many negative effects such as packet losses or high jitter.

V. Conclusions

We analyzed the performance of QUIC in WiFi, investigating the interactions of the protocol with 802.11 frame aggregation. We first highlighted that Chromium's QUIC (v.39) delivers sub-optimal throughput in typical WiFi scenarios. The root-cause is the way QUIC paces packets in the network and its acknowledgment mechanisms. Whereas these mechanisms are desirable to reduce packet loss in Internet paths, they prevent QUIC from benefiting from frame aggregation.

We implemented and evaluated *BQUIC*, i.e., a customized version of QUIC that increases traffic burstiness and, thus, opportunities for frame aggregation in WiFi. We carried out experiments using both a controlled testbed and a production WMN. Results showed that BQUIC increases the throughput between 20% to 30% with respect to current Chromium's implementation of the protocol.

Since our results are true for WiFi only and burstiness may be undesirable in some other scenarios, we believe that the protocol could be tuned for particular cases, e.g., enabling BQUIC whenever applications are running on WiFi. Performing experiments on an hybrid scenario combining WiFi and Internet-wide scale are left for future work.

Acknowledgments

This work was supported by the EMJD-DC program and Spanish grant TIN2016-77836-C2-2-R.

References

[1] (2017) Cisco Visual Networking Index. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] G. R. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. P. Costa, and B. Walke, "The ieee 802.11 universe," *IEEE Communications Magazine*, vol. 48, no. 1, 2010.

[3] A. Langley *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the SIGCOMM*, 2017, pp. 183–196.

[4] F. Gratzer, "Quic-quick udp internet connections," *Future Internet and Innovative Internet Technologies and Mobile Communications*, 2016.

[5] A. Bhartia, B. Chen, F. Wang, D. Pallas, R. Musaloiu-E, T. T.-T. Lai, and H. Ma, "Measurement-based, practical techniques to improve 802.11 ac performance," in *Internet Measurement Conference*. ACM, 2017, pp. 205–219.

[6] D. Skordoulis, Q. Ni, H. h. Chen, A. P. Stephens, C. Liu, and A. Jamalipour, "Ieee 802.11n mac frame aggregation mechanisms for next-generation high-throughput wlans," *IEEE Wireless Communications*, vol. 15, no. 1, pp. 40–47, February 2008.

[7] B. S. Kim, H. Y. Hwang, and D. K. Sung, "Effect of frame aggregation on the throughput performance of ieee 802.11n," in *2008 IEEE Wireless Communications and Networking Conference*, March 2008, pp. 1740–1744.

[8] Y. Lin and V. W. S. Wong, "Wsn01-1: Frame aggregation and optimal frame size adaptation for ieee 802.11n wlans," in *IEEE Globecom 2006*, Nov 2006, pp. 1–6.

[9] (2018) QUIC, a multiplexed stream transport over UDP. [Online]. Available: https://www.chromium.org/quic/

[10] E. Altman and T. Jiménez, "Novel delayed ack techniques for improving tcp performance in multihop wireless networks," in *IFIP International Conference on Personal Wireless Communications*. Springer, 2003, pp. 237–250.

[11] A. K. Singh and K. Kankipati, "Tcp-ada: Tcp with adaptive delayed acknowledgement for mobile ad hoc networks," in *Wireless Communications and Networking Conference*, vol. 3. IEEE, 2004, pp. 1685–1690.

[12] R. De Oliveira and T. Braun, "A dynamic adaptive acknowledgment strategy for tcp over multihop wireless networks," in *INFOCOM 2005*, vol. 3. IEEE, 2005, pp. 1863–1874.

[13] ——, "A smart tcp acknowledgment approach for multihop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 2, pp. 192–205, 2007.

[14] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at quic," in *Proceedings of the 2017 Internet Measurement Conference*, 2017.

[15] M. Trevisan, D. Giordano, I. Drago, M. Mellia, and M. Munafò, "Five years at the edge: Watching internet from the isp network," to appear in Proceedings of CoNEXT'18, Heraklion, Greece, 2018.

[16] "Sants-UPC Community Newtork," http://sants.guifi.net.

[17] "Community Networks Testbed for the Future Internet, CONFINE," http://confine-project.eu/, FP7 European Project 288535.

[18] "OpenWrt Linux distro. for embedded devices," https://openwrt.org.

[19] "Quick Mesh Project," http://qmp.cat.

[20] L. Cerdà-Alabern, A. Neumann, and L. Maccari, "Experimental evaluation of bmx6 routing metrics in a 802.11 an wireless-community mesh network," in *Future Internet of Things and Cloud (FiCloud)*. IEEE, 2015, pp. 770–775.

[21] "Open, Free and Neutral Network Internet for everybody," http://guifi.net/en.

[22] L. Cerdà-Alabern, A. Neumann, and P. Escrich, "Experimental evaluation of a wireless community mesh network," in *MSWiM'13*. Barcelona, Spain: ACM, Nov. 3–8, 2013.

[23] "qMp Sants-UPC monitoring page," http://dsg.ac.upc.edu/qmpsu.