Packet delivery time as a tie-breaker for assessing Wi-Fi access points

François Michel
UCLouvain
francois.michel@uclouvain.be

Olivier Bonaventure
UCLouvain
olivier.bonaventure@uclouvain.be

I. Introduction

As the Internet becomes more accessible and the connected devices more diverse we observe a large variety of services proposed to the Internet users. Starting with file exchange where the connection throughput had the most important impact on the user's perceived quality of experience, Internet users can now perform voice or video calls, access to a remote desktop or even play video games running in the cloud. These services distinguish themselves from file transfer by the fact that their quality of experience does not evolve linearly with the connection throughput. Video and voice transfers need a sufficient amount of bandwidth but won't use more bandwidth than what is required by the audio or video content. On the other hand, these real-time services are sensitive to delay. For instance, video-conferencing applications will suffer from a large delay as it reduces the interactivity of the video call. These applications are also sensitive to the delay jitter. Video-conferencing applications need every video frame to arrive a few milliseconds after the last played frame. A frame arriving too late due to packet loss or delay won't be played on time and therefore deteriorate the quality of experience. Applications sometimes introduce playback buffers to cope with packet loss and jitter. However, this solution introduces an artificial latency between the two communicating peers as the video frames spend some time in the playback buffer before they are actually played. These playback buffers thus need to be as small as possible to avoid this introduced latency. Forward Erasure Correction (FEC) can be used to cope with packet losses and therefore lower the latency induced by playback buffers. However, FEC doesn't help with jitter as the redundancy packets will arrive after the delayed video frame. In this case, playback buffers appear as the only remaining solution. This solution is however not suitable for applications with tight delay constraints such as cloud gaming services where the delay between a button press and its impact on screen must be as short as possible. Delay and jitter can come from different sources. For instance, the bufferbloat problem is a well known source of delay due to large buffers on routers in the Internet [1, 2] and to network access technologies having to deal with numerous users and bursty traffic such as 3G or 4G antennas [3]. Several techniques have been developed to reduce bufferbloat and the jitter it induces. Transport protocols use pacing to avoid sending data in bursts and filling the buffers. Some transport layer congestion control algorithms also focus on emptying the buffers and avoiding bufferbloat [4, 5].

Besides bufferbloat, some network access technologies are more prone to high jitter than other. This is the case for 802.11 Wi-Fi channels [6] that suffer from delay and jitter due to their collision avoidance mechanisms.

802.11 allows devices to be connected to the Internet using radios. Devices send data packets to an access point and the access point relays them to the Internet. These wireless communications are prone to interference because the medium is heavily shared, leading to frames losses. 802.11 therefore defines a retransmission mechanism in order to retransmit lost frames. The transmitter waits for a random amount of time before retransmitting a data frame when it does not receive an acknowledgement. This mechanism is known as *random backoff* and is used as a collision avoidance mechanism: if the frame loss was due to two hosts sending data at the same time, they avoid subsequent collisions by both waiting a different amount of time before retransmitting the frame. By default, 802.11 hosts will attempt to retransmit a lost frame several times before giving up, considering that the frame is definitely lost and transmit subsequent frames. The random backoff time ranges from a few microseconds to more than 20 milliseconds, leading to potentially large frame delivery times. Depending on the signal strength, 802.11 hosts may or may not transmit more than one frame at a time using Aggregated MAC Protocol Data Units (A-MPDU). With lower signal strengths and bitrates, A-MPDUs are barely used, meaning that all subsequent frames will be delayed until the current frame is transmitted successfully. When they are enabled, Block Acknowledgements signal which frames were lost. However, while subsequent frames were successfully received, some Wi-Fi drivers wait for some time to recover the lost frames before delivering the next packets to the upper layers [7]. All this make 802.11 channels prone to jitter, being unsuitable for real-time applications in some cases.

II. Experimenting with 802.11

We study how interference can impact the packets delivery time with 802.11. Our experimental setup is composed of one laptop, one Wi-Fi access point (1 meter away from the laptop) and one QUIC server on the Internet. The laptop and access point are located in a residential house, with several other Wi-Fi access points operating at both 2.4GHz and 5GHz. The access point is a Turris Omnia using OpenWrt 19.07 [8] and the `ath9k` Wi-Fi driver configured for using 802.11n at 2.4GHz. The QUIC server sends 25 video frames per second to the laptop at a fixed interval of one frame every 40 milliseconds during 5 minutes. We capture the raw 802.11 frames exchanged between the access point and the laptop by setting one of the interfaces of the access point into Monitor mode and study the amount of time needed to successfully transmit each frame from the access point to the laptop during the transfer.

We study two different configurations. In the first one, the access point is configured to use the first 2.4GHz Wi-Fi channel (2.412GHz), while in the second one the access point uses the channel 11 (2.462GHz). Everything else stays unchanged. The signal strength is similar and the bitrate chosen by the device is 144 Mb/s in both cases. Those two configurations would thus be considered equivalent by the end user. Figure 1 shows the CDF of the frame delivery time for these two configurations. We can see that while these two configurations would be considered similar by common systems (the signal strength and bitrate are similar), the channel 11 was crowded with other devices and thus more prone to interference in our experiments. There is a noticeable difference in frame delivery time that can significantly impact the quality of experience perceived by delay-sensitive applications. For instance, a cloud gaming service delivering videos at a rate of 60 frames per second would need at least one additional frame in its playback buffer to cope with jitters of more than 16 milliseconds, which happens in more than 1% of the cases for the configuration using channel 11.

III. Exposing the frame delivery time to end-users

We saw in the previous section that while the bitrate and the signal strength are the common metrics exposed to end-users in most systems, they may not be sufficient to determine if a specific access points is suitable for real-time applications. With an increasing amount of tightly delay-constrained services such as cloud gaming, there is an interest in exposing the frame

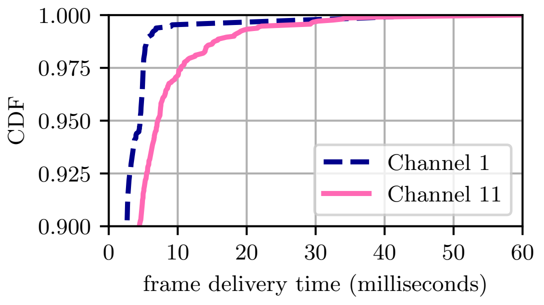

Figure 1: *Frame delivery time CDF for 802.11n on channels 1 and 11.*

delivery time as a metric to determine the quality of Wi-Fi access points. The average, median or a percentile of the frame delivery time could be displayed to the user alongside the bitrate and signal strength of the wireless connection. As this raw metric and its impact on real-time services may not be intuitive, operating systems such as Android or iOS could leverage this metric to flag Wi-Fi access points that are suitable for specific use-cases such as cloud gaming in an intuitive manner for the user.

On the other side, access points could report this metric to devices using beacon frames [9] and to maintainers using standard monitoring solutions such as SNMP [10].

IV. Exposing the frame delivery time to applications

Besides users, applications may also benefit from knowing the frame delivery time distribution of the available network interfaces. As this metric could be available at the startup of the application, the latter may leverage the metric to choose the correct interface (e.g. cellular VS Wi-Fi) to initiate the transfer instead of changing the used interface during the transfer, often resulting in a change of IP address, potentially breaking the current connection. Some transport protocols such as Multipath TCP or QUIC that feature connection migration may also decide during the transfer to change the network interface used to communicate with the peer (e.g. fall over the cellular interface for smartphones) if the current one has a large frame delivery time. Video or audio applications may also use this information to start the service with a tailored playback buffer in order to ensure a smooth playback from the beginning as the buffer avoids the perturbations caused by the Wi-Fi jitter.

Applications and transport protocols may also decide to renounce to some throughput in order to reduce the jitter. 802.11e allows to use Traffic Identifiers (TID) to reduce the Wi-Fi random backoff for frames tagged as video or audio traffic. The Linux kernel supports a mapping between Differentiated Service Code Point (DSCP) field of the IP header and 802.11e traffic identifiers [11]. This allows applications to apply these tags on a per-socket basis. Reducing the backoff will however increase the contention on the channel and likely reduce the throughput.

References

[1] Jim Gettys. Bufferbloat: Dark buffers in the internet. *IEEE Internet Computing*, 15(3):96–96, 2011.

[2] CACM Staff. Bufferbloat: What's wrong with the internet? *Communications of the ACM*, 55(2):40–47, 2012.

[3] Haiqing Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Tackling bufferbloat in 3g/4g networks. In *Proceedings of the 2012 Internet Measurement Conference*, pages 329–342, 2012.

[4] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, and Van Jacobson. BBR Congestion Control. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-00, Internet Engineering Task Force, July 2017. Work in Progress.

[5] Lawrence S Brakmo, Sean W O'Malley, and Larry L Peterson. Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 24–35, 1994.

[6] IEEE Computer Society LAN/MAN Standards Committee et al. Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11*, 2007.

[7] Linux mac80211 driver. `https://github.com/torvalds/linux/blob/f8e6dfc64f6135d1b6c5215c14cd30b9b60a0008/net/mac80211/rx.c#L1177`, 2021.

[8] Openwrt. `https://openwrt.org/`, 2021.

[9] Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, and Alec Wolman. Beacon-stuffing: Wi-fi without associations. In *Eighth IEEE Workshop on Mobile Computing Systems and Applications*, pages 53–57. IEEE, 2007.

[10] Mark Fedor, Martin Lee Schoffstall, James R. Davin, and Dr. Jeff D. Case. Simple Network Management Protocol (SNMP). RFC 1157, May 1990.

[11] Tim Szigeti, Jerome Henry, and Fred Baker. Mapping Diffserv to IEEE 802.11. RFC 8325, February 2018.