Regulating Aggregation Level For Low Latency in 802.11ac

Hamid Hassani
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
hassanih@tcd.ie

Francesco Gringoli
Dept. of Information Engineering
CNIT/University of Brescia
Brescia, Italy
francesco.gringoli@unibs.it

Douglas J. Leith
School of Computer Science and Statistics
Trinity College Dublin
Dublin, Ireland
doug.leith@tcd.ie

***Abstract*—In this paper we consider transport layer approaches for achieving high rate, low delay communication over edge paths where the bottleneck is a modern 802.11ac WLAN which can aggregate multiple packets into each WLAN frame. We show that regulating send rate so as to maintain a target aggregation level can be used to avoid queue buildup at the WLAN AP. We present a prototype transport layer implementation of our low delay rate allocation approach and use this to evaluate performance under real radio conditions.**

***Index Terms*—Packet aggregation, Wi-Fi, Low-delay, Rate-allocation**

I. Introduction

While much attention in 5G has been focussed on the physical and link layers, it is increasingly being realised that a wider redesign of network protocols is also needed in order to meet 5G requirements. Transport protocols are of particular relevance for end-to-end performance, including end-to-end latency. For example, ETSI has recently set up a working group to study next generation protocols for 5G [1]. The requirement for major upgrades to current transport protocols is also reflected in initiatives such as Google QUIC [2], Coded TCP [3] and the Open Fast Path Alliance [4].

In this paper we consider next generation edge transport architectures of the type illustrated in Figure 1. Traffic to and from client stations is routed via a proxy located close to the network edge (e.g. within a cloudlet). This creates the freedom to implement new transport layer behaviour over the path between proxy and clients, which in particular includes the last wireless hop. We focus on situations where the last hop in Figure 1 is a modern WLAN and is the bottleneck. The transmission delay of a packet sent over the downlink is composed of two main components: (i) queueing delay at the AP and (ii) MAC contention time. The MAC contention time is determined by the WLAN contention mechanism and typically small, so the main challenge is to regulate the queueing delay.

Key to our work is the ubiquity of aggregation in modern WLANs, a feature that brings goodput near to line-rate by reducing the relative time spent in accessing the channel when transmitting several packets to the same destination. Intuitively, the level of aggregation achieved is coupled to queueing. Namely, when only a few packets are queued then there are not enough to allow large aggregated frames to be assembled for transmission. Conversely, when there is a persistent queue backlog then there is a plentiful supply of packets and large frames can be consistently assembled

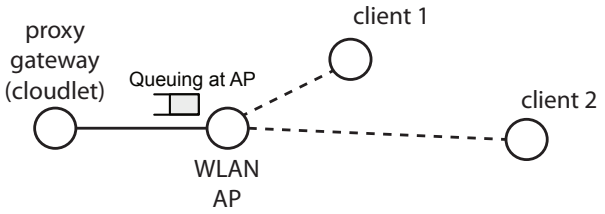


Fig. 1. Schematic of a cloudlet-based edge transport architecture. The bottleneck lies in WLAN hop so that queueing of downlink packets occurs at the AP (indicated on schematic).

but queueing delay may be high and newly arriving packets can be dropped because of the backlog. A simple measure to avoid dangerous backlog episodes is therefore to estimate the aggregation level of frames received at client stations and report this back to the sender, which can then adjust its send rate accordingly. In this paper we show how this can be easily achieved without requiring any modification to kernels and, more importantly, transparently to the AP.

In summary, our main contributions are as follows: (i) we establish that regulating send rate so as to maintain a target aggregation level can be used to realise high rate, low delay communication over modern WLANs; and (ii) we present a prototype transport layer implementation of this low delay rate allocation and evaluate its performance.

A. Related Work

In recent years there has been an upsurge in interest in userspace transports due to their flexibility and support for innovation combined with ease of rollout. Notable examples of new transports developed in this way include Google QUIC [2], UDT [5] and Coded TCP [3], [6], [7]. ETSI has also recently set up a working group to study next generation protocols for 5G [1]. The use of performance enhancing proxies, including in the context of WLANs, is also not new e.g. RFC3135 [8] provides an entry point into this literature. However, none of these exploit the use of aggregation in WLANs to achieve high rate, low delay communication.

Aggregation in WLANs has been standard since the release of the 802.11n amendment in 2009, but the literature has primarily focused on analysis and design for wireless efficiency, managing loss etc. For a recent survey see for example [9]. Most analytic work is confined to so-called saturated conditions where transmitters always have a packet to send, see for example [10]. When stations are not saturated

(so-called finite-load operation) then for WLANs which use aggregation (802.11n and later) most studies resort to the use of simulations to evaluate performance due to the complex interaction between arrivals, queueing and aggregation with CSMA/CA service, and we also adopt this approach here.

TCP BBR [11] is currently being developed by Google and this also targets high rate and low latency, although not specifically in edge WLANs. The BBR algorithm tries to estimate the bottleneck bandwidth and adapt the send rate accordingly to try to avoid queue buildup. The delivery rate in BBR is defined in [12] as the ratio of the in-flight data when a packet left the sender to the elapsed time when its ACK is received. This may be inappropriate, however, when the bottleneck is a WLAN hop since aggregation can mean that increases in rate need not correspond to increases in delay. In addition, a small queue at the AP can be beneficial for aggregation and so throughput.

II. Preliminaries

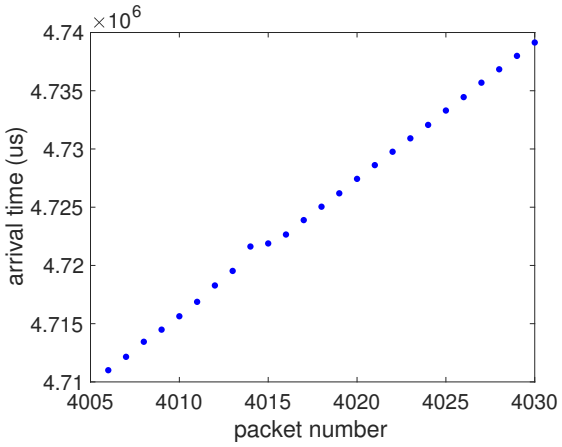
A. Aggregation in 802.11n, 802.11ac etc

A feature shared by all new WLAN standards since around 2009 (when 802.11n was introduced) has been the use of aggregation to amortise PHY and MAC framing overheads across multiple packets. This is essential for achieving high throughputs. Since PHY overheads are largely of fixed duration, increasing data rates reduces the time spent transmitting the frame payload but leaves the PHY overhead unchanged. Hence, the efficiency, as measured by the ratio of the time spent transmitting user data to the time spent transmitting an 802.11 frame, decreases as data rates increase unless the frame payload is also increased i.e. several user packets are aggregated and transmitted in a single frame.

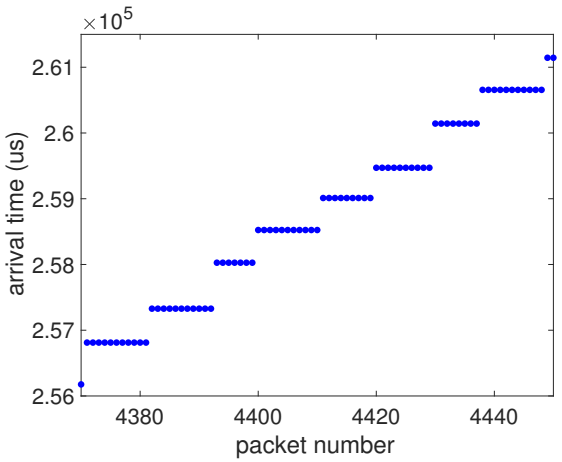
Since the packets aggregated in a frame share the same destination station (at least in current 802.11 standards), aggregation effectively requires that per station queueing be used at the WLAN access point. When few packets are queued then there are insufficient packets available to allow large aggregated frames to be assembled for transmission. While the scheduler might delay transmission in anticipation of more packets arriving it is known that this is generally not advantageous, e.g. see [10].

B. Measuring Aggregation

The level of aggregation can be readily measured at a receiver using packet timestamps. Namely, a timestamp is typically added by the NIC to each packet recording the time when it is received. When a frame carrying multiple packets is received then those packets have the same timestamp and so this can be used to infer which packets were sent in the same frame. For example, Figure 2 shows measured packet timestamps for two different downlink send rates. The setup consists of Linux server (running Debian Jessie) sending UDP packets using Iperf 2.0.5 to two Linux clients (running Debian Stretch and with Broadcom BCM4360 802.11ac NICs) via an Asus RT-AC86U Access Point (which uses a Broadcom 4366E chipset). This setup allows high spatial usage (we observe that almost always three spatial streams are used) and high data rates (up to MCS 9). It can be seen from Figure 2(a) that when the UDP arrival rate at the AP is relatively



(a) 10Mbps send rate



(b) 200Mbps send rate

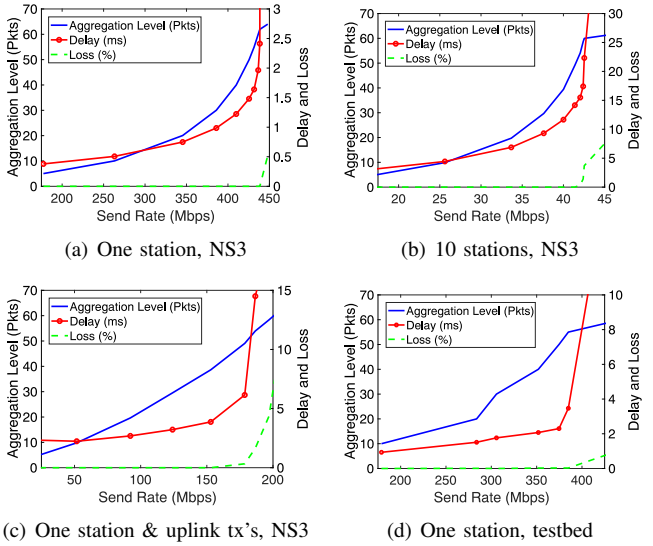
Fig. 2. Illustrating timestamp measurements for UDP packets transmitted over an 802.11ac downlink via aggregated frames to two client stations. The same downlink send rate is used for for both client stations. Data is shown for one of the client stations: while the sending rate does not allow aggregation at the top, at the bottom the queue is constantly full and aggregation occurs.

low each received packet has a distinct timestamp whereas at higher arrival rates, see Figure 2(b), packets start to be received in bursts having the same timestamp. This behaviour reflects the use by the AP of aggregation at higher arrival rates, as confirmed by inspection of the radio headers in the corresponding tcpdump data.

III. Low Delay High-Rate Operation

To provide high rate, low latency communication for next generation 5G edge services we would like to select a downlink send rate which is as high as possible yet ensures that a persistent queue backlog does not develop at the AP. While measurements of round-trip time might be used to estimate the onset of queueing and adjust the send rate, it is known that this can be inaccurate when there is queueing in the reverse path [13] and in 802.11 networks [14]. Accurately measuring one-way delay is also known in general to be challenging[1]. In contrast, the number of packets aggregated in

[1]The impact of clock offset and skew between sender and receiver applies to all network paths. In addition, when a wireless hop is the bottleneck then the transmission delay can also change significantly over time depending on the number of active stations e.g. if a single station is active and then a second station starts transmitting the time between transmission opportunities for the original station may double.



(a) One station, NS3 (b) 10 stations, NS3

(c) One station & uplink tx's, NS3 (d) One station, testbed

Fig. 3. Measurements of average aggregation level, one-way packet delay and packet loss vs the send rate for a range of network conditions. (a) downlink flow to one client station, (b) 10 downlink flows to each of 10 client stations, data shown is for one of these flows, (c) setup as in (a) but with contention from an uplink flow, (d) setup as in (a) but measurements are from a hardware testbed located in an office environment.

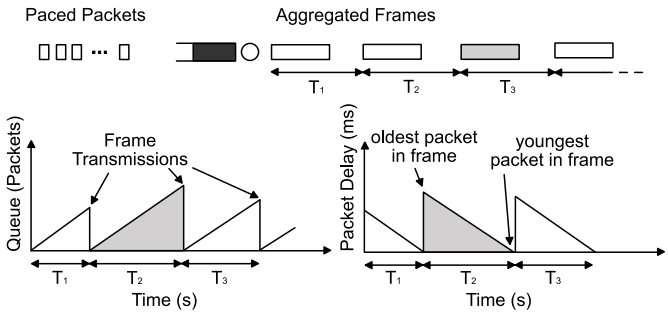


Fig. 4. Illustrating connections between queueing, packet delay and frame aggregation at the AP. Packets arriving at the AP are queued for transmission, the queue growing roughly linearly over time as packets arrive in a paced fashion. When a transmission opportunity occurs an aggregated frame is constructed from the queued packets. Provided the number of queued packets is less than the maximum frame aggregation level N_{max} then the queue backlog is cleared by the transmission. The delay of the oldest packet in a frame is upper bounded by the time between transmission opportunities.

a frame is relatively easy to measure accurately and reliably at the receiver, as already noted.

Figure 3 shows measurements of the mean aggregation level, packet delay and loss vs the send rate to a client station for a range of network configurations. A number of features are evident. Firstly, as the send rate is increased the aggregation level increases monotonically until it reaches the maximum value N_{max} supported by the MAC (for the data shown $N_{max} = 64$ packets). Secondly, the packet delay increases monotonically with send rate, initially increasing slowly but then increasing sharply as the send rate approaches the downlink capacity. Observe that the sharp increase in delay coincides with aggregation level approaching its upper limit of 64 packets and with the onset of packet loss. Note that all packet loss in this data is due to queue overflow since we verified that link layer retransmissions repair channel losses.

We can understand the behaviour in Figure 3 in more detail by reference to the schematic in Figure 4. Packets are transmitted by the sender in a paced fashion. On arriving at the AP they are queued until a transmission opportunity occurs. The queue occupancy increases roughly linearly since the arriving packets are paced (have roughly constant inter-arrival times). Upon a transmission opportunity the queued packets are assembled into an aggregated frame and transmitted. Provided the queue is less than N_{max} the queue backlog is cleared by this transmission. For example, consider the shaded frame in Figure 4. This frame is transmitted at the end of time interval T_2 and the packets indicated by the shaded area on the queueing plot are aggregated into this frame. The oldest packet in this frame could have arrived just after interval T_1 and so may have waited up to T_2 seconds before transmission. Later arriving packets will, of course, experience less delay than this. The intervals T_1, T_2 etc between frame transmissions are random variables due to the randomised channel access mechanism used by 802.11 transmitters. Importantly, these intervals depend on the aggregation level, i.e. the duration T_2 depends on the time taken to send the frame aggregated from packets arriving in interval T_1 etc, and in turn the aggregation level depends on the interval duration since more packets arrive in a longer interval. The delay and aggregation level are therefore coupled to one another and this is what we see in Figure 3. Note that the intervals between transmissions may also vary due to contention with other transmitters (uplink transmissions by clients, transmissions by other WLANs sharing the same channel etc), link layer retransmissions, transmissions by the AP to other clients (recall modern APs use per station queueing so the coupling is only via these intervals) and so on but the basic setup remains unchanged and this is also reflected in Figure 3.

The data in Figure 3 suggests that if we could operate the system at an aggregation level of around 32 packets we might obtain a high transmit rate while maintaining low delay. It is this observation that underlies the approach we propose here. Note that the AP transmit efficiency increases with the aggregation level since the overheads on a frame transmission are effectively fixed and so sending more packets in a frame increases efficiency. Hence, operating at less than the maximum possible aggregation level N_{max} incurs a throughput cost and there is therefore a trade-off between delay and rate. However, Figure 3 indicates that this trade-off is quite favourable, i.e. low delay comes at the cost of only a relatively small reduction in rate compared to the maximum possible.

A. *Controlling Delay*

We proceed by introducing a simple feedback loop that adjusts the sender transmit rate (corresponding to the AP arrival rate, assuming no losses between sender and AP) to maintain a specified target aggregation level. Namely, time is partitioned into slots of duration Δ seconds and we let $\mathcal{T}_{i,k}$ denote the set of frames transmitted to station i in slot k. Station i measures the number of packets $N_{i,f}$ aggregated in frame f and reports the average $\mu_{N_i}(k) := \frac{1}{|\mathcal{T}_{i,k}|}\sum_{f\in\mathcal{T}_{i,k}} N_{i,f}$ back to the sender. The sender then uses proportional-integral (PI) feedback[2] to increase its transmit

[2]While design of more sophisticated control strategies is of interest, this is an undertaking in its own right and we leave this to future work.

rate x_i if the observed aggregation level $\mu_{N_i}(k)$ is less than the target value N_ϵ and decrease it if $\mu_{N_i}(k) > N_\epsilon$. This can be implemented using the pseudo-code shown in Algorithm 1. Note that this feedback loop involves three design parameters, update interval Δ, feedback gain K and target aggregation level N_ϵ. Typical values are $\Delta = 500$ms, $K = K_0/n$ with $K_0 = 1$ (where n is the number of client stations in the WLAN) and $N_\epsilon = 32$ packets.

Algorithm 1 Feedback loop adjusting transmit rate x_i to regulate aggregation level μ_{N_i}.

$k = 1$
while 1 **do**
  $\mu_{N_i} \leftarrow \frac{1}{|\mathcal{T}_{i,k}|} \sum_{f \in \mathcal{T}_{i,k}} N_{i,f}$
  $x_i \leftarrow x_i - K(\mu_{N_i} - N_\epsilon)$
  $k \leftarrow k + 1$
end while

B. Multiple Stations: Equal Airtime Fairness

When there are multiple client stations we can modify Algorithm 1 as follows to allocate roughly equal airtime to each station. Recall that the airtime used to transmit the payload of station i is $T_i = \mu_{N_i} L/\mu_{R_i}$, where L is the number of bits in a packet, μ_{N_i} is the number of packets in a frame and μ_{R_i} is the MCS rate used to transmit the frame in bits/s. So selecting $\mu_{N_i} = \mu_{R_i}/\mu_{R_{i^*}}$ makes the airtime equal with $T_i = T_{i^*}$ for all stations. Letting $\boldsymbol{x}$ denote the vector of downlink send rates to ensure equal airtimes we therefore increase the rate x_{i^*} of the station i^* with highest MCS rate $\mu_{R_i}(k)$ when its observed aggregation level $\mu_{N_i}(k)$ is less than the target value N_ϵ and decreases x_{i^*} when $\mu_{N_i}(k) > N_\epsilon$, i.e. at slot k

$$x_{i^*}(k+1) = x_{i^*}(k) - K(\mu_{N_i}(k) - N_\epsilon) \quad (1)$$

The rates of the other stations are then assigned proportionally,

$$x_i(k+1) = x_{i^*}(k+1)\frac{\mu_{R_i}}{\mu_{R_{i^*}}},\ i = 1, \ldots, n \quad (2)$$

Note that the update (1)-(2) only uses readily available observations. Namely, the frame aggregation level $N_{i,f}$ and the MCS rate $R_{i,f}$, both of which can be observed in userspace by packet sniffing on client i.

IV. Performance Evaluation

We implemented the controller both in an experimental testbed and in the NS3 simulator. We use the experimental testbed to evaluate performance under real channel conditions and hardware/software constraints while the NS3 allows us to evaluate performance with larger numbers of client stations, and also allows controlled changes to channel conditions.

A. Experimental Testbed

We select an Asus RT-AC86U as Access Point: it is equipped with a Broadcom 4366E chipset and supports 802.11ac MIMO with up to four spatial streams. We activate only the 5GHz radio and configure channel bandwidth to 80MHz, keeping other settings at their defaults. This setup allows high spatial usage (we observe that almost always three spatial streams are used) and high data rates (up to MCS11). Note that we also carried out experiments with different chipsets at the AP (e.g., QCA chipsets) and did not observe any major differences. By default aggregation supports AMSDU's with up to 128 packets in a frame (namely 64 AMSDUs each containing two packets). In our tests we disabled AMSDU's to force AMPDU aggregation since this facilitates monitoring, in which case up to 64 packets can be aggregated in a frame. All nodes are located within 10 meters from the AP and we carefully inspected that all stations are always receiving 11ac aggregates. While it may be interesting to also add fall-back procedures for managing corner cases where the sender switches to legacy modulations, we prefer to leave its analysis and design as future work.

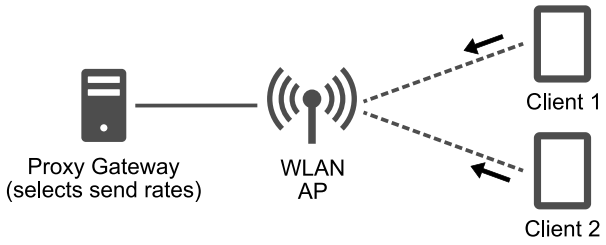
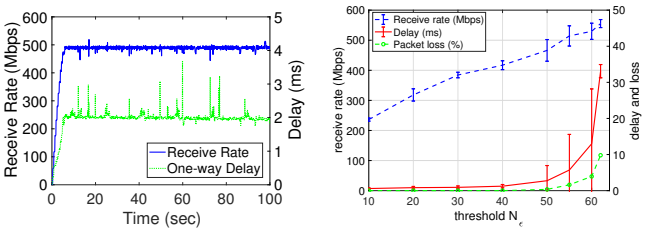


Fig. 5. Schematic of scheduler architecture. Clients send reports of observed aggregation level and MCS rate to proxy which then uses this information to adjust the downlink send rate to each station.

A Linux server connected to this AP via a gigabit switch uses Iperf 2.0.5 to generate UDP downlink traffic to the WLAN clients, which are Linux boxes running Debian Stretch and with Broadcom BCM4360 802.11ac NIC. As all machines are tightly time synchronised over a LAN, we use the sender-side timestamp inserted by Iperf to estimate the one-way packet delay (the time between when a packet is passed into the socket in the sender and when it is received). Note, however, that in production networks accurate measurement of one-way delay is typically not straightforward as it is difficult to maintain accurate synchronisation between server and client clocks (NTP typically only synchronises clocks to within a few tens of milliseconds).

B. Prototype Rate Allocation Implementation

We implemented feedback of measured aggregation level from the clients to the sender using the software architecture illustrated in Figure 5. Clients measure/estimate the aggregation level of received frames and periodically report this data back to the sender at intervals of Δ seconds as the payload of a UDP packet. The sender uses a modified version of Iperf 2.0.5 where we implement the feedback collector and our rate control algorithm. Recall that we are considering next generation edge transports and so the sender would typically be located in the cloud close to the network edge. While it may be located on the wireless access point this is not essential, and indeed we demonstrate this feature in all of our experiments by making use of a proprietary closed access point. In the following, the connection between the sender and the AP is a dedicated link that we carefully size to never act as a bottleneck. For the same reason, the delay introduced by this link is constant over time, and negligible with respect to the timing of the control algorithm.



(a) (b)

Fig. 6. (a) Illustrating low-latency high-rate operation in an 802.11ac WLAN and (b) measured goodput, delay and loss vs target aggregation level N_ϵ (each data point summarises 250s of measurements). Measurements are from a hardware testbed located in an office environment.

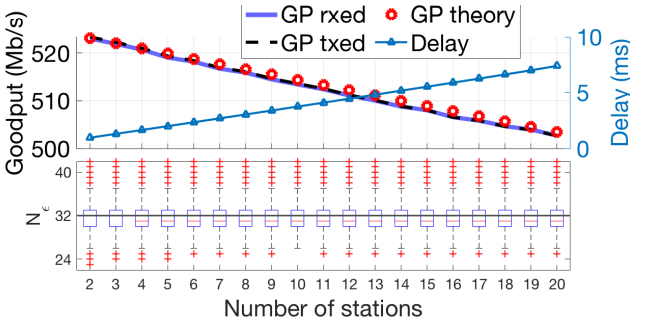


Fig. 7. NS3 sum-goodput and delay vs number of receivers (top) and corresponding distribution of aggregation level about the target value of $N_\epsilon = 32$ (bottom). Top plot: the *GP theory* line is the upper limit computed by assuming an AMPDU with $N_\epsilon = 32$ packets, 10 feedback packets per second per receiver, 10 beacons per second, and no collisions.

C. Modifications to NS3

We also implemented the controller as a new module in NS3. Based on the received feedbacks it periodically configures the sending rate of `udp-client` applications colocated at a single node connected to an Access Point. Each wireless station receives a single UDP traffic flow at a `udp-server` application that we modified to collect frame aggregation statistics and periodically transmit these to the controller. We also developed a round-robin scheduler at the AP and added new functions to let stations determine the MCS of each received frame together with the number of MPDU packets it contains. In the following experiments we use a 802.11ac physical layer operating over an $80MHz$ channel, using VHT rates for data frames and legacy rates for control frames. Unless otherwise stated $N_{max} = 64$ packets, target $N_\epsilon = 32$, PHY MCS=9 and the number of spatial streams NSS = 2. As validation we reproduced a number of the simulation measurements experimentally and found them to be in good agreement. We ran all simulations on NS-3.29 over a quad-core Intel Core i7 workstation: as our code adds minor modifications to WiFi classes, the simulation time for all the considered topologies is not affected by our control framework. The new NS3 code and the software that we used to perform experimental evaluations are available open-source [3].

D. Goodput vs Delay and Loss

Figure 6(a) shows a typical time history of rate and delay obtained by regulating the aggregation level. These measurements are from a hardware testbed located in an office environment. It can be seen that the one-way delay is low, at around 2ms, while the send rate is high, at around 500Mbps (this data is for an 802.11ac downlink using three spatial streams and MCS 9). Increasing the send rate further leads to sustained queueing at the AP and an increase in delay, but the results in Figure 6(a) illustrate the practical feasibility of operation in the regime where the rate is maximised but the onset of sustained queueing is avoided.

Figure 6(b) shows measurements of the client station goodput (i.e. receive rate after any packet loss), one-way delay and packet loss (measured at the transport layer, so after any link layer retransmissions) as the target aggregation level N_ϵ is varied. Also shown are error bars indicating one standard deviation. Note that losses due to decoding errors were always recovered by retransmissions at the link-layer, thus packet loss is associated only with queue overflow within the AP.

When the aggregation level exceeds 40 packets (well below the upper limit $N_{max} = 64$), delay and loss rise rapidly with N_ϵ. When the aggregation is regulated to a lower level, i.e., below 40 packets, the delay is consistently low and there are no queue overflow losses. For N_ϵ around 30 or 40, which ensures low delay, the rate is around 80% of the maximum.

E. Multiple Clients

1) Clients Same Distance From AP: We begin by considering a network where client stations are all located two meters from the AP. Fig. 7 (top) shows measurements of the aggregated application layer goodput and average delay vs the number of receivers for a target $N_\epsilon = 32$. The aggregated goodput measured at the receivers is close to the theoretical limit supported by the channel (MCS) configuration, being only a few Kb/s below this for 20 receivers. This goodput is evenly shared by the receivers (the measured Jain's Fairness Index is always 1). The average delay increases almost linearly at the rate of $350\mu s$ per additional station. The lower plot in Fig. 7 shows the measured distribution of frame aggregation level, with the edges of the boxes indicating the 25th and 75th percentiles. Here the feedback algorithm tightly concentrates the aggregation level around the target value of $N_\epsilon = 32$. We did not observe any losses.

2) Randomly Located Clients: We next consider a scenario where client stations are randomly located in a square of side 40m with the AP at its center. We choose MinstrelHT algorithm as rate controller: this adjusts the MCS used by each client station based on its channel conditions (better for stations closer to the AP, worse for those further away). To ease visualisation we use NSS=1 which helps to reduce the MCS fluctuations generated by Minstrel. We ran experiments with eight receivers until we collected 200 points where the rate controller converged to a stable choice for all receivers, i.e., with more than 85% of frames received with the same MCS. We group receivers by MCS and report statistics on N_ϵ for each group as boxplots in the top-left plot in Figure 9. Thick circles indicate the choice of rate allocation that assigns equal airtime to all receivers: we notice that the measured rate allocation is maintained close to those values.

[3]Code can be obtained by contacting the corresponding author.

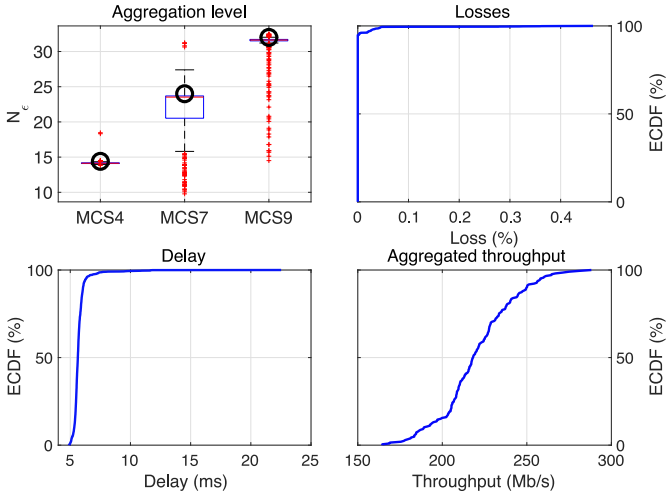


Fig. 8. Performance with 8 receivers placed randomly in a square of side 40m: MCS is chosen by MinstrelHT algorithm, NSS = 1. NS3.

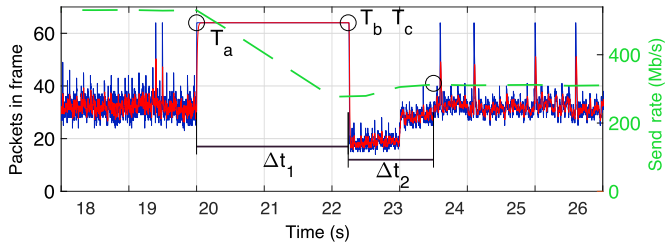


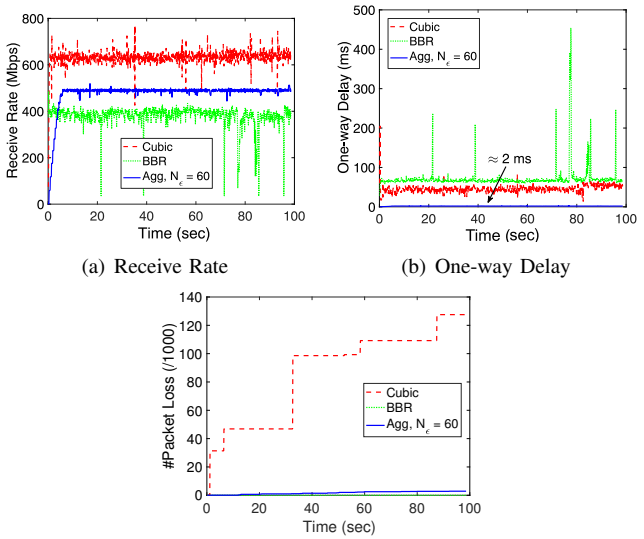
Fig. 9. Illustrating adaption of send rate by feedback algorithm in response to a change in channel conditions (from use of 2 spatial streams down to 1 spatial stream). Single client station, MCS 9, NS3.

In clockwise order Figure 9 then shows the ECDF of losses, aggregated goodput and delay. Observe that losses occur in this configuration because of far away nodes not being able to correctly decode all packets. Delay is consistently less than 6ms. The aggregate goodput can drop as low as $150Mb/s$ when MinstrelHT selects MCS4 for all receivers, but converges to $300Mb/s$ with MCS 9 (the theoretical maximum goodput with MCS 9, NSS 1 and $N_\epsilon = 32$ is 307Mb/s).

3) Responding to Channel Changes: The feedback algorithm used by the AP regulates the send rate to maintain a target aggregation level. It therefore adapts automatically to changing channel conditions. This is illustrated in Fig. 9. Initially the AP uses 2 spatial streams, MCS 9 and then at $t = T_a = 20s$ it switches to 1 spatial stream. For $\Delta t_1 = 2.24s$ all AMPDUs hit the maximum aggregation level of 64 packets and we start observing losses. During this time it can be seen that the algorithm, which updates the send rate twice per second ($\Delta = 500ms$), is slowing down the sending rate. It takes four rounds to reach a rate compatible with the channel, but it takes a little bit more to stabilise the aggregation level at the AP in $t = T_b$. After another three rounds (for approximately $\Delta t_2 = 1.26$) it can be seen that the sending rate settles at its new value in $t = T_c$.

F. Performance Comparison With TCP Cubic & BBR

We briefly compare the performance of our proposed aggregation-based rate control algorithm with TCP Cubic [15], the default congestion control algorithm used by



(a) Receive Rate (b) One-way Delay

(c) #Packet Loss

Fig. 10. Comparing the performance of aggregation-based rate control algorithm with TCP Cubic and BBR. The one-way delay in (b) is averaged over 100ms intervals. Experimental testbed.

Linux and Android. In addition, we compare performance against TCP BBR [11] since this is a state-of-the-art congestion control algorithm currently being developed by Google and which also targets high rate and low latency. In the following experiments we use the default parameters that are configured in the Debian Jessie sender.

Figure 10 shows typical receive rate and one-way delay time histories measured for the three algorithms. It can be seen from Figure 10(a) that Cubic selects the highest rate (around 600Mbps) but from Figure 10(b) that this comes at the cost of high one-way delay (around 50ms). This is as expected since Cubic uses loss-based congestion control and so increases the send rate until queue overflow (and so a large queue backlog and high queueing delay at the AP) occurs. As confirmation, Figure 10(c) plots the number of packet losses vs time and it can be seen that these increase over time when using Cubic, each step increase corresponding to a queue overflow event followed by backoff of the TCP congestion window.

BBR selects the lowest rate (around 400Mbps) of the three algorithms, but surprisingly also has the highest end-to-end one-way delay (around 75ms). High delay when using BBR has also previously been noted by e.g. [16] where the authors propose that high delay is due to end-host latency within the BBR kernel implementation at both sender and receiver. However, since our focus is not on BBR we do not pursue this further here but note that the BBR Development team at Google is currently developing a new version of BBR v2.

Our low delay aggregation-based approach selects a rate (around 480 Mbps), between that of Cubic and BBR, consistent with the analysis in earlier sections. Importantly, the end-to-end one-way delay is around 2ms i.e. more than 20 times lower than that with Cubic and BBR. It can also be seen from Figure 10(c) that it induces very few losses (a handful out of the around 4M packets sent over the 100s interval shown).

V. Summary & Conclusions

In this paper we establish that regulating send rate so as to maintain a target aggregation level can be used to realise high rate, low delay communication over modern WLANs. We present a prototype transport layer implementation of this low delay rate allocation and evaluate its performance. Note that the results presented here are a preliminary evaluation which demonstrates that measuring aggregation level can be a useful tool, and we leave theoretical analysis and evaluation in more complex scenarios (including co-existence with legacy WLAN clients) to future work. We also plan to analyse how features from next generation Wi-Fi aimed at improving the throughput, like OFDMA-based Multiuser-MIMO and Real Simultaneous Dual Band (RSDB), may affect our control algorithm and how we can exploit the new features for increasingly reducing the overall latency.

VI. Acknowledgements

Authors are grateful to the Relations and International Mobility office of the University of Brescia that provided the necessary funding for hosting Prof. Douglas Leith in Brescia for one month during which part of the measurements reported in this paper were collected.

References

[1] *Next Generation Protocols – Market Drivers and Key Scenarios*. European Telecommunications Standards Institute (ETSI), 2016.

[2] J. Iyengar and I.Swett, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," *IETF Internet Draft*, 2015.

[3] M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. J. Leith, and M. Medard, "Congestion control for coded transport layers," in *Proc Int'l Conf on Communications (ICC)*, 2014, pp. 1228–1234.

[4] *Open Fast Path http://www.openfastpath.org/*, 2016.

[5] Y. Gu and R. L. Grossman, "Udt: Udp-based data transfer for high-speed wide area networks," *Comput. Netw.*, vol. 51, no. 7, pp. 1777–1799, 2007.

[6] M. Karzand, D. J. Leith, J. Cloud, and M. Medard, "Design of FEC for Low Delay in 5G," *IEEE J. Selected Areas in Comms (JSAC)*, vol. 35, no. 8, pp. 1783–1793, 2016.

[7] A. Garcia-Saavedra, M. Karzand, and D. J. Leith, "Low Delay Random Linear Coding and Scheduling Over Multiple Interfaces," *IEEE Trans Mobile Computing*, vol. 16, no. 11, pp. 3100–3114, 2017.

[8] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," Internet Requests for Comments, RFC Editor, RFC 3135, 2001.

[9] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Impact of ieee 802.11n/ac phy/mac high throughput enhancements on transport and application protocols: A survey," *IEEE Comms Surveys Tutorials*, vol. 19, no. 4, pp. 2050–2091, 2017.

[10] T. Li, Q. Ni, D. Malone, D. Leith, Y. Xiao, and T. Turletti., "Aggregation with Fragment Retransmission for Very High-Speed WLANs," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 591–604, 2009.

[11] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[12] Y. Cheng, N. Cardwell, S. H. Yeganeh, and V. Jacobson, "Delivery Rate Estimation," *IETF Internet Draft*, 2017.

[13] A. Pathak, H. Pucha, Y. Zhang, Y. C. Hu, and Z. M. Mao, "A measurement study of internet delay asymmetry," in *Proc 9th Int'l Conf on Passive and Active Network Measurement*, ser. PAM'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 182–191.

[14] D. Malone, D. J. Leith, and I. Dangerfield, "Inferring queue state by measuring delay in a wifi network," in *Traffic Monitoring and Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 8–16.

[15] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[16] Y. Im, P. Rahimzadeh, B. Shouse, S. Park, C. Joe-Wong, K. Lee, and S. Ha, "I sent it: Where does slow data go to wait?" in *Proc Fourteenth EuroSys Conf*, ser. EuroSys '19. New York, NY, USA: ACM, 2019, pp. 22:1–22:15.