

Coexistence of Age Sensitive Traffic and High Throughput Flows: Does Prioritization Help?

Tanya Shreedhar
Wireless Systems Lab, IIIT-Delhi
tanyas@iiitd.ac.in

Sanjit K. Kaul
Wireless Systems Lab, IIIT-Delhi
skkaul@iiitd.ac.in

Roy D. Yates
WINLAB, Rutgers University
ryates@winlab.rutgers.edu

Abstract—We study the coexistence of high throughput traffic flows with status update flows that require timely delivery of updates. A mix of these flows share an end-to-end path that includes a WiFi access network followed by paths over the Internet to a server in the cloud. Using real-world experiments, we show that commonly used methods of prioritization (DSCP at the IP layer and EDCA at the 802.11 MAC layer) in networks are highly effective in isolating status update flows from the impact of high throughput flows in the absence of WiFi access contention, say when all flows originate from the same WiFi client. Prioritization, however, isn't as effective in the presence of contention that results from the throughput and status update flows sharing WiFi. This results in prioritized status update flows suffering from a time-average age of information at the destination server that is about the same as when all flows have the same priority.

I. INTRODUCTION

IoT devices often communicate their updates, which require timely delivery to a server in the cloud, over an end-to-end path that includes a shared wireless access followed by a multihop path over the Internet to the server. The update traffic often shares the path with traffic that would like to achieve high throughput. Update packets that require timeliness will suffer large delays if queued together with high throughput flows. They may also suffer significant delays in obtaining transmission opportunities over a shared multiaccess when competing for the same with high throughput flows. In practice, the networking stack allows priorities to be associated with data flows using, for example, the mechanism of Differentiated Services Code Point (DSCP). In principle, this can help alleviate the adverse consequences of update packets sharing the network with throughput flows.

In this work, we empirically shed light on the benefits of prioritizing update packets sent over a shared WiFi access to a server in the cloud. We use the time average age of information (AoI) [1] to quantify timeliness. Transmission Control Protocol (TCP) flows are used to emulate high throughput traffic. For end-to-end flows carrying update packets, we regulate the end-to-end rate of updates using the Age Control Protocol (ACP) [2]–[5], which has been shown to provide good timeliness performance over paths of interest in this work.

Work on optimizing metrics of the age of information has considered packet management techniques, including priorities and preemption when multiple sources share a service facility [6]–[12]. Such work often uses queueing models to capture

sharing of the network resources. However, contention has not been modeled when sources share a multiaccess channel. Also, these works assume that all traffic sharing the facility requires timely delivery. Last but not least, it is often assumed that packet management may discard a source packet.

Given the shared WiFi access and Enhanced Distributed Contention Access (EDCA), we have different queues for the ACP and TCP flows in our work as we assign a higher priority to update packets (ACP flows). The queues, however, are FCFS and don't allow preemption. Our specific contributions are:

- 1) We provide an empirical evaluation of the impact of coexisting ACP and TCP flows on the time-average age of information of the ACP flows and the throughputs of the TCP flows. Both flows share a WiFi network and have a server in the cloud as their destination.
- 2) Using different experimental configurations (a) with and without prioritization, (b) with and without shared access, and (c) in the absence of TCP flows, we show that while giving ACP flows higher priority in the absence of contention over the WiFi access (all flows originate from the same WiFi client) effectively isolates the ACP flows from the TCP flows, the contention that is caused when WiFi access is introduced and all flows originate from different WiFi clients results in barely any gains from prioritization.
- 3) We show from our experiments that as the number of ACP flows become large enough, TCP and ACP flows (prioritized) sharing the same WiFi access is worse both in terms of the throughputs of the TCP flows and the timeliness achieved by the ACP flows.

II. RELATED WORK

Several works [6]–[9] analyze the average age of updates in the presence of priority traffic. In [6], the authors analyze the average age of updates when the sources are assigned different priorities for two service facilities. One which allows source agnostic preemption in service by a new arrival of equal or higher priority and the other in which there is a waiting room of size 1 and a new arrival can preempt an update in waiting but not in service. In [7], the authors expand the waiting room to allow each source to have up to one waiting update while the server is busy. This also allows an update in service that is preempted by a higher priority source to be saved in the waiting room to resume service later. In [8], the authors analyze peak age when sources have priorities and queues

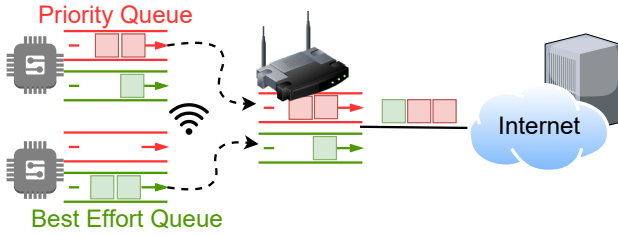


Figure 1: Illustration of priority queueing over a shared WiFi access. Nodes and AP maintain separate queues for different service classes.

are of infinite size for Poisson arrivals and general service times. In [9], the authors propose and analyze three source aware packet management policies considering a memoryless service facility of a single queue and server. The facility sees arrivals from two independent Poisson sources. The policies make different choices regarding the size of the waiting room and whether preemption is allowed in the service. In [11], the update currently in service is preempted instead of discarding a new arrival. In [12] arrivals consist of a mix of ordinary and priority updates. The latter can preempt any update in service. In case the preempted update is ordinary, it is not discarded and is queued for resuming service later.

Systems research that analyzes AoI in real-world settings is relatively limited [2]–[5], [13], [14]. [13] brings forth the need for an AoI optimizer that can adapt to changing network topologies and delays. The Age Control Protocol (ACP) [2]–[4], is a transport-layer solution that works in an application-independent and network-transparent manner and attempts to minimize the age of information of a source at a monitor connected via an end-to-end path over the Internet. In [5], the authors propose a modification to ACP and also compare it with other state-of-the-art TCP congestion control algorithms used in the Internet. In [14], WiFresh, a MAC and application-layer solution to ageing of updates over a wireless network is proposed. There are also works on various other aspects of the metrics of AoI, including optimizing age over multiaccess channels and can be found in [15] and [16].

III. PRIORITIZATION IN NETWORKS

Several mechanisms exist in modern networked systems that commonly address network bottlenecks by allowing priority packets to pass first [17], [18]. The majority of such mechanisms operate by categorizing network traffic into distinct “service classes” – each one assigned a separate queue. Based on the QoS demands of each class, these mechanisms manage the rate of each class queue such that the services can access a bottleneck link depending on their priority. For example, a router at the bottleneck link may handle voice-over-IP (VoIP) application traffic using a high-priority, high-rate queue, while packets of video streaming applications over the same link might be forwarded at a significantly lower rate.

There are several ways in which network operators can classify network flows into different service classes in their

managed routers. For example, operators may use the destination IP address and port to identify an application type (e.g., data egress from Netflix servers) or prioritize based on the transport protocol used (RTP may have a different priority than UDP/TCP traffic) [19]. The most common traffic classification method uses Differentiated Services Code Point (DSCP) markings. Application providers can assign their packets with a unique code in the IP layer. Each code maps to a unique traffic class type that can be treated with a different priority. The current standard dictates network management control traffic to be assigned the highest priority, followed by interactive applications, low-loss low-latency data transfers, and finally, best-effort data transfer applications [17], [18]. As the DSCP value is embedded in the IP header (layer 3) of every packet, it is visible to every router on the Internet and thus allows for end-to-end flow prioritization.

However, since multiaccess schemes like WiFi operate at layer 2 (medium access control) of the networking stack, they remain oblivious to DSCP markings in the IP layer. Instead, the 802.11 standard employs its prioritization using Enhanced Distributed Channel Access (EDCA) or Hybrid Controlled Channel Access (HCCA) [20]. Similar to DSCP, the 802.11 prioritization assigns eight separate queues at the MAC layer in which data packets are segregated based on their priority level (defined as User Priority). Each priority level is assigned to one of the four access categories (analogous to DSCP traffic classes), i.e. background (AC_BK), best-effort (AC_BE), video (AC_VI) and voice (AC_VO) (arranged in increasing priority order). Each access category uses different CSMA/CA minimum and maximum contention window sizes and also inter-frame spacing (IFS). This enables packets belonging to a higher priority access category faster access to the shared channel and less contention from lower priority packets awaiting access.

Recent efforts have mapped DSCP markings to 802.11 EDCA priority and access categories [21]. It is now possible for application providers to assign their traffic higher priority in both wired and wireless networks by setting DSCP in the IP header. Table I summarizes different traffic classes and their priority mappings between DSCP (Diffserv) and 802.11.

IV. EXPERIMENTAL SETUP AND METHODOLOGY

Figure 2 illustrates our real-world experimental setup. For our experiments, we use the ORBIT testbed [22], which is an open wireless network emulator grid located in Rutgers University, USA. ORBIT houses multiple programmable radio nodes deployed in a controlled grid-like environment with adjacent WiFi nodes along rows and columns at a distance of 1 m from each other. Each ORBIT node runs Ubuntu 18.04 LTS over Linux kernel v5.0. By default, ORBIT nodes use the 1 Gbps ethernet NIC to connect to the Internet. We set up one of the ORBIT nodes as an 802.11n access point configured to operate at 5 GHz on a fixed channel using `hostapd` and the Atheros (`ath9k`) Linux WiFi driver [23]. To focus on the interplay between priorities and contention, we disable the automated WiFi physical layer (PHY) rate control in `ath9k`

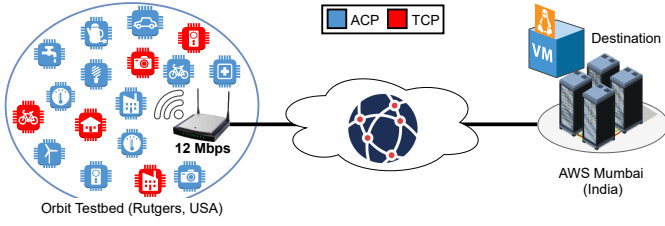


Figure 2: An illustration of the network topology. Clients containing a mix of TCP (red) and ACP (blue) are connected to a WiFi AP located in the Orbit Testbed’s WiFi grid in USA. The server is located in AWS Mumbai, India.

drivers and instead use a *fixed* WiFi PHY rate for the length of an experiment. While most of our experiments use a PHY rate of 12 Mbps, we also use 6 Mbps for some experiments. We provide experiment-specific PHY rates in §V. We select up to 80 nodes as WiFi clients in the ORBIT testbed and associate them to the ORBIT node configured as the WiFi access point. Our WiFi access point routes every packet received over WiFi to the public Internet over Ethernet. We also set up a node in the testbed as a *sniffer* that captures all packets sent over the WiFi channel during our experiments. The sniffer data allows us to estimate MAC layer packet retry percentages suffered by the ACP and TCP flows over the WiFi access. It also helps confirm that EDCA priorities have been applied.

We use an ec2 AWS instance in Mumbai, India, as our destination server for all flows. The baseline round-trip-time (RTT), calculated by sending one packet for every ACK between our WiFi clients and the server is ≈ 200 -210 ms. We evaluate three different flow configurations.

- 1) *ACP-default*. Update packets are sent over an end-to-end path between a WiFi client (the ACP source) and the AWS server (ACP monitor) using the Age Control Protocol [5], [24]. Update packets sent by ACP are given the default priority and treated as best-effort traffic.
- 2) *ACP-priority*. It is same as ACP-default but here ACP packets are given the highest network priority by setting the DSCP value as CS7 (see Table I).
- 3) *TCP-iperf*. We use *iperf3* to generate TCP traffic from WiFi clients to the AWS server. We configure each TCP flow to use the *cubic* congestion control [25]. TCP flows are always treated as best effort in our work.

In addition to priority queueing at our configured WiFi access (available default in the 802.11 standard), we use CAKE [26] at our AP node to support QoS priority at the Ethernet interface. CAKE is a comprehensive network queue management utility that has been deployed as part of the OpenWRT framework and is available in all Linux kernels version 4.19 and later [27]. CAKE supports Differentiated Services (DiffServ) prioritization scheme and maps ACP-priority flows to the highest priority queue (reserved for voice applications) ingressing the Ethernet interface. On the other hand, CAKE treats flows belonging to ACP-default and TCP-iperf as the lowest priority best-effort traffic.

IETF Diffserv Service Class	DSCP	802.11 Access Category	User Priority
Network Control	CS7, CS6	AC_VO (Voice)	7
Signaling	CS5	AC_VI (Video)	5
Multimedia Conferencing/ Streaming	AF41-43, AF31-33	AC_VI (Video)	4
High Throughput Data	AF11-13	AC_BE (Best Effort)	3
Low-Priority Data	CS1	AC_BK (Background)	1

Table I: Diffserv QoS mapping in wired (DSCP) and WiFi access.

We use three different experiment configurations and simultaneously run ACP and TCP flows to evaluate the gains from prioritizing ACP flows. Specifically, in *Baseline Priority* (BP) we initiate one or more ACP-priority and TCP-iperf flows from a single WiFi client. This setting eliminates any interference between the flows due to contention over the WiFi access. It focuses on the co-existence of ACP prioritized flows and TCP flows in what effectively is a setting with a single server and one FCFS queue for every priority class. In *Multiaccess Priority* (MP), each ACP-priority and TCP-iperf flow runs on a separate WiFi client. The flows therefore compete for the shared WiFi multiaccess, resulting in contention. Lastly, in *Best Effort* (BE), as in MP, each flow begins in a different WiFi client. We have ACP-default flows where no priority is assigned along with TCP-iperf flows. All flows are thus treated as best effort.

Evaluation Metrics. We define our evaluation metrics. The performance of an ACP flow (ACP-default or ACP-priority) is evaluated in terms of the estimate of *time-average age* [1] at the source. Note that since the source of the flow (a WiFi client) is not time-synchronized with the AWS server, age can’t be calculated at the server. We bank on ACP ACK packets sent by the server back to the source in response to every update packet sent by the source to estimate the age. The round-trip-time (RTT) corresponding to an ACK-ed source packet is assumed to be the packet’s system time. Age is assumed to reset to this time when the source receives an ACK. *Out-of-sequence* older ACKs are discarded, which is in line with the *freshness* requirement. Using RTTs as an estimate of system time can lead to over-estimation of age. However, since we consider a linear age function, the bias in estimation does not affect the optimal operating point. Works [3], [5] contains the design principles and details about ACP. In §V we present the *mean time-average age*, which is the mean calculated over all ACP flows.

We also discuss ACP throughput, which is the end-to-end rate (Mbps) of ACK-ed source packets and is calculated by the source based on the ACK packets it receives and the size of sent update packets. An update packet is of size 1024 bytes in all our experiments. We will also present *WiFi MAC packet retry percentages*. These simply capture the percentage of packets

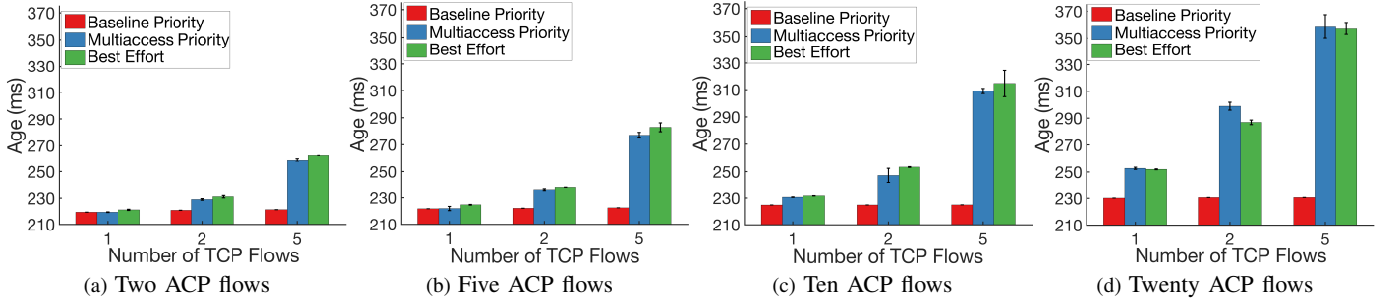


Figure 3: Mean time-average age achieved by 2, 5, 10 and 20 ACP flows for *Baseline Priority*, *Multiaccess Priority* and *Best Effort* configurations for 1, 2, and 5 coexisting TCP flows.

on air that were *retries* for a source. The retry packets are marked with a retry flag which is captured by the sniffer. Last but not least, TCP throughput is also a metric of interest. For all metrics, we present the mean calculated over 3 repeats of an experiment, where each experiment is 1000 seconds long.

V. EVALUATION

We discuss our observations from experiments performed using the methodology described in §IV. They help gain insight into whether prioritizing benefits ACP flows when they share a WiFi access with TCP flows.

Are there any gains from prioritizing ACP flows over the shared WiFi multiaccess?

Figure 3 shows the mean time-average age achieved by ACP flows sharing the WiFi network with TCP flows. Figures 3a, 3b, 3c and 3d show the mean age for when the number of ACP flows are set to 2, 5, 10, and 20 respectively. For each selection of a number of ACP flows, the mean age is shown for when the number of TCP flows are 1, 2, and 5. Further, for each selection of number of ACP and TCP flows, the mean is shown for the network configurations of *Baseline Priority* (BP), *Multiaccess Priority* (MP) and *Best Effort* (BE).

Contention over the shared WiFi multiaccess increases in the configurations MP and BE when the number of ACP clients or TCP clients increases. Let's begin by considering the mean age achieved under *Baseline Priority*. For a given number of ACP flows, for example, 5 flows in Figure 3b, the mean age stays unchanged for different numbers of TCP flows. For 5 ACP flows, this age is ≈ 222 ms for 1, 2, and 5 TCP flows. The age is ≈ 230.5 ms when there are 20 ACP nodes as in Figure 3d. The age stays the same for a given number of ACP nodes because in BP all ACP and TCP flows originate in the same WiFi client. Because of their higher priority than TCP flows, ACP flows are unaffected by changes in the number of TCP flows originating in the WiFi client. On the other hand, an increase in the number of ACP flows does result in an increase in the mean age. This is because updates from a larger number of ACP flows share the same priority queue in the WiFi client. Mean age increases from ≈ 220 ms for 1 - 2 ACP flows (Figure 3a) to ≈ 230 ms for 20 ACP flows (Figure 3d). As seen in Figure 3 for MP and BE configurations in which flows

are distributed over different WiFi clients sharing the WiFi multiaccess, mean age increases significantly as the number of TCP flows increases for any selection of the number of ACP flows. Assigning a higher priority to ACP flows, as in MP, is ineffective in isolating them from the effects of TCP flows sharing the multiaccess. Also, the mean age when using *Multiaccess Priority* is in general not much smaller than when treating ACP with the same priority as TCP when using *Best Effort*.

Further, we observe from Figure 3 that for any selected number of TCP and ACP flows, contention over the multiaccess results in a higher mean age in comparison to BP. In fact, even for just 2 ACP and 2 TCP flows, we see that the mean age for the setting of MP is about 9 ms more than that for *Baseline Priority*. This gap increases rapidly with an increase in the number of ACP and TCP flows. For example, it jumps to 38 ms for 2 ACP and 5 TCP flows, is 55 ms for when we have 5 ACP and 5 TCP flows, and is 130 ms for 20 ACP and 5 TCP flows. To understand the reason behind significantly worse mean age when using MP, we begin by considering the throughput obtained by the TCP flows. Later we also look at the WiFi MAC layer retry percentages suffered by update packets of ACP flows and also their round-trip times (RTTs).

Figure 4 shows the sum throughput (sum of throughputs of all ACP and TCP flows) for the network configurations BP, MP, and BE, and different numbers of TCP and ACP flows. It can be observed that the sum throughput is about the same for all the configurations and all numbers of TCP and ACP flows. It stays in the very narrow range of 8.8 to 9 Mbps. Essentially, the TCP and ACP flows together achieve the available data payload rate of about 9 Mbps, given the link rate of 12 Mbps. The figure also shows the share of ACP flows and that of TCP flows in the sum throughput. As can be seen, the fraction of sum throughput that corresponds to ACP flows increases with the number of ACP flows. As expected, the sum throughput of ACP flows for *Baseline Priority* is only a function of the number of ACP flows and is not impacted by the number of TCP flows. This throughput is 0.8 Mbps for when we have 2 ACP flows and goes up to about 5 Mbps for 20 ACP flows.

Further note, for a given number of ACP flows, the sum throughput of TCP flows stays about the same for the con-

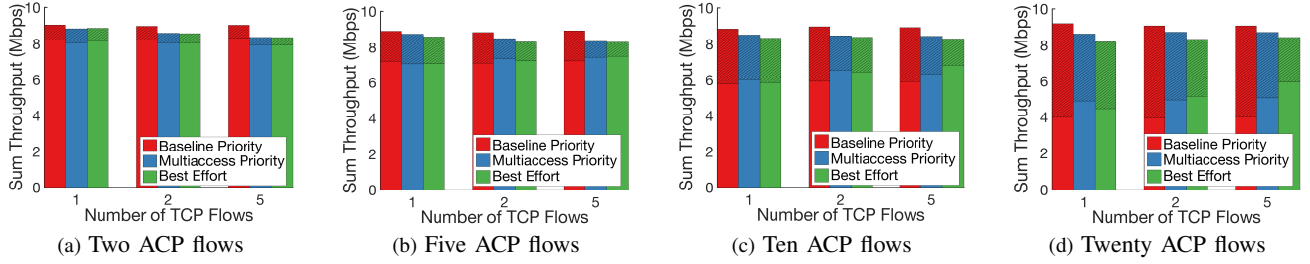


Figure 4: Sum throughput of ACP and TCP flows with their respective shares for 2, 5, 10 and 20 ACP flows. For each stacked bar, the diagonally striped top part corresponds to the sum of ACP flows and the bottom part shows the sum TCP throughput. For each number of ACP flows, the throughputs are shown for 1, 2, and 5 coexisting TCP flows and for *Baseline Priority*, *Multiaccess Priority* and *Best Effort*.

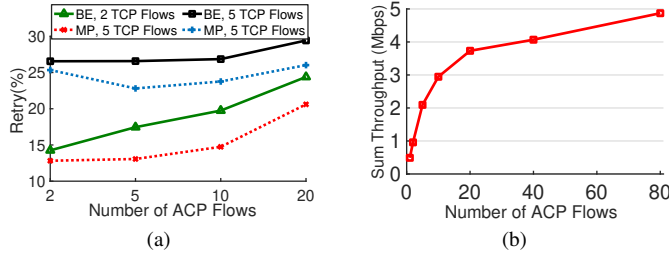


Figure 5: (a) Retry percentages of update packets sent in ACP flows as a function of number of ACP sources. Percentages are shown for *Best Effort* and *Multiaccess Priority*, and for 2 and 5 TCP flows. (b) ACP sum throughput in the absence of TCP as a function of the number of ACP flows sharing a 6 Mbps WiFi link.

figurations BP, MP, and BE. For BE, it is within ≈ 2 Mbps of sum TCP throughputs for *Baseline Priority*. Specifically, for larger numbers of ACP flows, the TCP sum throughput is greater by at most ≈ 1 Mbps when using *Multiaccess Priority* compared to using BP. When using BE, it is at most ≈ 2 Mbps higher. TCP throughput benefits in BE because ACP flows have the same access priority as TCP flows. The above observation tells us that the significant increases in mean age seen in Figure 3 with an increase in the number of TCP flows for a given number of ACP flows, for MP and BE, may not be entirely attributed to TCP's throughput share.

While an increase in TCP flows doesn't impact the TCP sum throughput, it results in increased MAC layer retries of packets of ACP flows. It also results in ACP flows experiencing large RTTs. Figure 5a shows the packet retry percentages as a function of the number of ACP flows for two and five TCP flows and the configurations of *Multiaccess Priority* and *Best Effort*. Retry percentages increase by about 5% - 10% when the numbers of TCP flows increase from 2 to 5. We also see higher retry percentages when there are larger numbers of ACP flows. Also, observe that for a given number of TCP flows, *Best Effort* sees higher retry percentages than MP. This is because having priority has ACP flows see a little less contention over the WiFi multiaccess.

We also look at the mean RTTs of updates packets to see the impact of increased MAC layer retries. Figures 6a

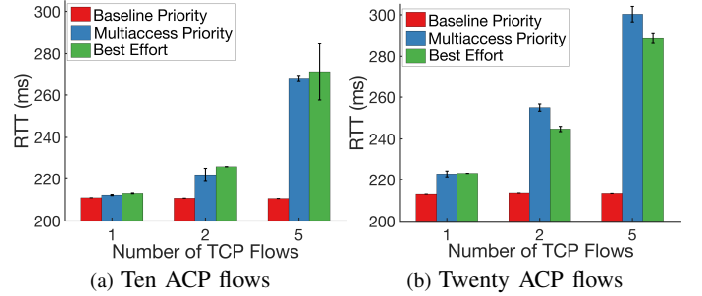


Figure 6: Mean RTT of ACP flows. For ten and twenty ACP flows, RTT is shown for *Baseline Priority*, *Multiaccess Priority* and *Best Effort*, and for 1, 2, and 5 coexisting TCP flows.

and 6b show, respectively for 10 and 20 ACP flows, significant increases in RTT as the number of TCP flows increase from 1 to 5, for MP and BE. These, together with the retry rates, explain the large mean ages observed when using multiaccess in comparison to when using *Baseline Priority*.

Summary — Gains from prioritizing ACP flows vanish quickly with an increase in contention over the shared WiFi multiaccess. The increased contention leads to higher retries and higher RTTs, resulting in higher time-average age.

How does the performance of ACP flows sharing a 6 Mbps WiFi link without interference from TCP flows compare to all flows sharing a 12 Mbps WiFi link?

Figure 7 shows the mean age achieved by ACP flows when they share the WiFi multiaccess of rate 6 Mbps in the absence of TCP flows (labeled *No TCP*) and when the ACP and TCP flows share a 12 Mbps in MP configuration. For the *No TCP* setting, we see that the mean age is 209.43, 219.18, 243.79, 275.15, 327.41 ms, respectively for 5, 10, 20, 40, and 80 ACP flows. The corresponding sum ACP throughputs (see Figure 5b) are 2, 2.9, 3.7, 4 and 5 Mbps. So with 80 ACP flows sharing the WiFi multiaccess with a link rate of 6 Mbps, and utilizing almost all of it (a sum throughput of 5 Mbps), the mean age is 327.41 ms. Compare these mean ages for the *No TCP* setting with 10 and 20 ACP flows under *Multiaccess Priority* when a 12 Mbps WiFi link is shared with 1 - 5 TCP flows (see: Figure 7). For 10 ACP nodes it is 230.7, 247.25,

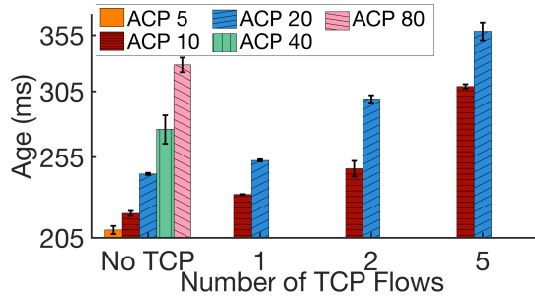


Figure 7: Mean time-average age for ACP flows with increasing TCP flows. In *No TCP*, all ACP flows share a 6 Mbps WiFi link. For all other settings, ACP and TCP flows share a 12 Mbps link in *Multiaaccess Priority* configuration.

and 309 ms. For 20 ACP it is 252.76, 298.9, and 358.77 ms, respectively. Clearly, ACP achieves lower age even with 80 flows and lower link rate (of 6 Mbps) when compared with 20 ACP flows coexisting with TCP flows even at a higher link rate of 12 Mbps.

For TCP, throughput is the utility of interest. For TCP flows sharing a 6 Mbps WiFi link (without any ACP flows), the sum TCP throughput is ≈ 5.5 Mbps, which is the expected payload rate after accounting for overheads like packet headers. For ACP and TCP flows sharing a 12 Mbps link, the sum throughput is 5 Mbps (1, 2 or 5 TCP flows) for when we have 20 ACP flows and is in the range of 6 - 6.5 Mbps for 10 ACP flows (see Figure 4). In fact, it is only when we have very few ACP flows, that the TCP sum throughputs are much larger than 5.5 Mbps. For when TCP shares with only 1 ACP flow, the sum TCP throughput is as high as 8 Mbps. With 5 ACP flows, the sum throughput ranges from 7 - 7.5 Mbps.

Additionally, the retry percentages for *No TCP* are 2% for 5 ACP flows and increase to 17% for 80 ACP flows (plot not included). Contrast these with the much higher retry rates in Figure 5a for when ACP flows share a 12 Mbps link with TCP flows. We also look at the impact of retry rate on RTTs and find that even the RTTs are smaller, with 20 ACP flows seeing an RTT less than 220 ms.

Summary — When 20 ACP flows share a 12 Mbps access with TCP flows, having TCP and ACP flows use non-interfering 6 Mbps WiFi links is beneficial to both. ACP mean ages are much smaller and TCP gets a higher sum throughput of 5.5 Mbps.

VI. CONCLUSION

We studied the impact of prioritization on the performance of age-sensitive traffic in the presence of competing network traffic. We considered an array of experimental configurations in real-world network settings. Our results indicate that ACP flows gain from prioritization only when contention over the wireless access from competing traffic is low. The gains are non-existent as the contention increases. We also find that a large number of ACP and TCP flows using non-interfering

6 Mbps WiFi links results in both better throughput and age performance, respectively for TCP and ACP flows, than when the flows share a 12 Mbps access.

REFERENCES

- [1] S. Kaul, R. Yates, and M. Gruteser, "Real-Time Status: How Often Should One Update?" in *Proc. IEEE INFOCOM Mini Conference*, 2012.
- [2] T. Shreedhar, S. K. Kaul, and R. D. Yates, "Poster: ACP: Age Control Protocol for Minimizing Age of Information over the Internet," in *MOBICOM*. ACM, 2018.
- [3] T. Shreedhar, S. K. Kaul, and R. D. Yates, "An Age Control Transport Protocol for Delivering Fresh Updates in the Internet-of-Things," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019.
- [4] T. Shreedhar, S. Kaul, and R. Yates, "ACP: An End-to-End Transport Protocol for Delivering Fresh Updates in the Internet-of-Things," *CoRR*, 2019. [Online]. Available: <http://arxiv.org/abs/1811.03353>
- [5] T. Shreedhar, S. K. Kaul, and R. D. Yates, "An Empirical Study of Ageing in the Cloud," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021.
- [6] S. K. Kaul and R. D. Yates, "Age of Information: Updates with Priority," in *IEEE International Symposium on Information Theory (ISIT)*, 2018.
- [7] A. Maatouk, M. Assaad, and A. Ephremides, "Age of Information With Prioritized Streams: When to Buffer Preempted Packets?" in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019.
- [8] J. Xu and N. Gautam, "Peak Age of Information in Priority Queuing Systems," *IEEE Transactions on Information Theory*, 2021.
- [9] M. Moltafet, M. Leinonen, and M. Codreanu, "Average AoI in Multi-Source Systems With Source-Aware Packet Management," *IEEE Transactions on Communications*, 2021.
- [10] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015.
- [11] E. Najm and E. Telatar, "Status updates in a multi-stream M/G/1/1 preemptive queue," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018.
- [12] E. Najm, R. Nasser, and E. Telatar, "Content Based Status Updates," *IEEE Transactions on Information Theory*, 2020.
- [13] C. Sönmez, S. Baghaee, A. Ergişi, and E. Uysal-Biyikoglu, "Age-of-Information in Practice: Status Age Measured Over TCP/IP Connections Through WiFi, Ethernet and LTE," in *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2018.
- [14] I. Kadota, M. S. Rahman, and E. Modiano, "Wifresh: Age-of-information from theory to implementation," in *International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021.
- [15] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: A new concept, metric, and tool," *Foundations and Trends in Networking*, 2017.
- [16] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of Information: An Introduction and Survey," *IEEE Journal on Selected Areas in Communications*, 2021.
- [17] F. Baker, J. Babiarez, and K. H. Chan, "Configuration Guidelines for DiffServ Service Classes," RFC 4594, Aug. 2006. [Online]. Available: <https://rfc-editor.org/rfc/rfc4594.txt>
- [18] "Differentiated Services Field Codepoints (DSCP)," <https://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml>.
- [19] "QoS: Classification Configuration Guide," https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_classn/configuration/xs-16/qos-classn-xe-16-book/qos-classn-ntwk-trfc.html, 2018.
- [20] A. Malik, J. Qadir, B. Ahmad, K.-L. Alvin Yau, and U. Ullah, "QoS in IEEE 802.11-based wireless networks: A contemporary review," *Journal of Network and Computer Applications*, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1084804515000892>
- [21] T. Szigeti, J. Henry, and F. Baker, "Mapping Diffserv to IEEE 802.11," RFC 8325, Feb. 2018. [Online]. Available: <https://rfc-editor.org/rfc/rfc8325.txt>
- [22] "Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)," <https://www.orbit-lab.org/>.
- [23] Linux Manual, "hostapd - IEEE 802.11 AP," <https://manpages.debian.org/testing/hostapd/hostapd.8.en.html>.
- [24] "ACP+: Improved Age Control over the Internet," <https://github.com/tanyashreedhar/AgeControlProtocolPlus>, 2021.
- [25] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, 2008.
- [26] T. Høiland-Jørgensen, D. Täht, and J. Morton, "Piece of CAKE: a comprehensive queue management solution for home gateways," in *IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2018.
- [27] "tc-cake(8) — Linux manual page," <https://man7.org/linux/man-pages/man8/tc-cake.8.html>.