TCP Download Performance in Dense WiFi Scenarios

Mukulika Maity
Department of Computer Science and Engineering
IIT Bombay
Email: mukulika@cse.iitb.ac.in

Bhaskaran Raman
Department of Computer Science and Engineering
IIT Bombay
Email: br@cse.iitb.ac.in

Mythili Vutukuru
Department of Computer Science and Engineering
IIT Bombay
Email: mythili@cse.iitb.ac.in

Abstract—**How does a dense WiFi network perform, specifically for the common case of TCP download? While the empirical answer to this question is 'poor', analysis and experimentation in prior work has indicated that TCP clocks itself quite well, avoiding contention-driven WiFi overload in dense settings. This paper focuses on measurements from a real-life use of WiFi in a dense scenario: a classroom where several students use the network to download quizzes and instruction material. We find that the TCP download performance is poor, contrary to that suggested by prior work. Through careful analysis, we explain the complex interaction of various phenomena which leads to this poor performance. Specifically, we observe that a small amount of upload traffic generated when downloading data upsets the TCP clocking, and increases contention on the channel. Further, contention losses lead to a vicious cycle of poor interaction with autorate adaptation and TCP's timeout mechanism. To reduce channel contention and improve performance, we propose a modification to the AP scheduling policy to improve the performance of large TCP downloads. Our solution, WiFiRR, picks only a subset of clients to be served by the AP during any instant, and varies this set of "active" clients periodically in a round-robin fashion over all clients to ensure that no client starves. By reducing the number of contending nodes at any point of time, WiFiRR improves the download time of large TCP flows by 3.2$\times$ in a simulation of our classroom scenario.**

I. Introduction

The omni-presence of WiFi needs no justification. While WiFi standards have improved significantly in terms of raw bit-rate, whether this has translated to corresponding improvements in application throughput is unclear. We are specifically interested in dense user scenarios, such as conferences, sports stadiums, and large classrooms, with the latter two being especially nascent with respect to WiFi usage. How does a dense WiFi network perform, specifically for the common case of TCP downloads? This is the focus of our work.

Prior work has shown, both analytically [1], [2] and experimentally [3], [4], that TCP download performance does not degrade with increasing number of users in a WLAN. These results are based on the performance of long running TCP flows in controlled environments, using homogeneous well-tested clients and artificial user traffic. These studies have reported good TCP download performance even with over a hundred clients [3].

In contrast, this paper presents a measurement study of TCP performance "in the wild" over a dense WiFi network, with real users running real applications over a variety of client devices. We conduct several measurements in a WiFi-enabled classroom, where students download online quiz questions and instruction material. Our results show that, in contrast to prior work, TCP performance degrades significantly in a dense usage scenario, even with 20–30 clients per access point. (We focus on a single WiFi BSS, and do not address scaling issues across multiple interfering BSSs.)

We have analyzed why our results differ from the TCP download scenarios in prior research. With long running TCP downloads, the only traffic on the network is TCP data packets in the downlink and ACKs in the uplink. In such cases, the number of contenting nodes on the channel is usually quite low, because the AP alone transmits TCP data, and only the clients that most recently received a data packet are likely to contend for the channel to send an TCP ACK. In contrast, in our real-life measurements, we found significantly higher channel contention due to "chattiness" of real applications that create a small but noticeable amount of extra upload traffic besides TCP ACKs.

For example, in our classroom scenario, a student logs in to the class webpage, authenticates herself, locates a file to download on a webpage (that has several smaller web objects in addition to the main object of interest), using a browser that opens several parallel TCP connections to download the content. In addition, users also have a low volume of background traffic automatically generated by email clients and such. Somewhat surprisingly, this small amount of extra traffic in the upload direction significantly increases the contention on the channel (as the number of active clients is now close to the total number of users), resulting in collisions due to the CSMA MAC protocol's channel arbitration mechanism. As a result, we found that TCP performance degraded severely, and students often took more than 8$\times$ the amount of time to download the files needed for an in-class quiz, as compared to a universe where TCP scaled perfectly with increasing user density.

We find that the contention on the wireless channel and the resulting collision losses also have an undesirable effect on several other protocols in the system. For example, we observed that WiFi clients picked lower bit rates during (and for a short period of time after) contention, because most rate adaptation algorithms confuse collisions for channel losses. This lowering of rate increases the time taken for subsequent transmissions, further increasing contention, leading to a vicious cycle. Further, we observed poor interaction between channel contention and TCP's timeout mechanism. We found that the RTT of TCP flows was highly variable due to contention losses, confusing the TCP timeout algorithm, leading to spurious retransmissions. Note that while prior work [5], [6], [7], [8], [9], [10] has also observed some subset of these problems, our analysis has focused on comprehensively identifying *all* factors that contribute to poor TCP download performance in dense scenarios, and understanding their complex interplay.

We also observed that in practice, several device drivers become unresponsive when operating under high contention losses, and need a driver reset to function even after the contention has subsided. All of these real-life effects further exacerbate the performance issues of TCP in a dense WiFi network. Note that we have verified and eliminated other factors (AP buffer mismanagement, wired network or server overload, external interference) as possible causes for the poor performance.

Having identified excessive channel contention as the root cause behind the performance issues, we propose a solution, WiFiRR, to improve the performance of large TCP downloads in dense WiFi scenarios. WiFiRR works as a scheduler at the packet queue of an access point. WiFiRR identifies a subset of clients as "active" during every instant of time (up to 5 clients in our implementation), and the AP serves downlink packets only to these clients. This results in the other clients going quiet during this period, leading to lower contention and improved performance. This set of active clients is varied periodically (every 6s in our case) to cover all clients in a round-robin fashion. Note that while clients may temporarily be deprived of service for short durations, they will eventually see improved performance over large TCP downloads. We evaluate our solution in simulation, and find that WiFiRR improves TCP download time by $3.2\times$ over the base case of serving all clients uniformly all the time. Our solution also improves download time by $2.25\times$ over WiFox [5], another solution that seeks to improve TCP download performance in dense scenarios. We realize that WiFiRR is not suited for dense WiFi deployments that see predominantly short or interactive flows, and adapting WiFiRR to work in such scenarios is part of ongoing work.

Our contributions can be summarized as follows: (a) a real-life measurement study of TCP download performance and its careful analysis, which identifies the factors that contribute (and eliminates the factors that do not) to poor performance in dense scenarios, and (b) a solution approach that improves the download time of large TCP flows by reducing channel contention.

The rest of the paper is organized as follows. Section II discusses related work. Section III describes our measurement study in a real classroom, and Section IV describes some controlled experiments and simulations we conducted to understand the measurement results in the classroom. Section V describes our solution WiFiRR that improves performance by addressing the problems we found. Finally, Section VI concludes the paper.

II. Related Work

Starting with Bianchi's seminal work [11], several researchers have analytically shown that the performance of 802.11 CSMA/CA degrades with increase in offered load, due to increased contention on the wireless channel. This analysis assumes saturated traffic, i.e., all stations are always backlogged and contend for the channel. [12] further generalizes the result, and shows that collision probability increases with increasing number of stations. However, subsequent research [1], [2] has considered a more specific problem of TCP downloads over 802.11. In this case, the analysis shows that the number of contending stations is much lower than the total number of stations due to the TCP data/ack clocking mechanism. When several downlink flows go through an AP, and the AP sends a data packet to a client at a certain instant, the client that received this data packet alone will generate a TCP ACK, and contend with the AP for the channel. All the other clients will not actively contend for the channel at this instant, until data packets arrive for them from the AP. This data/ack clocking mechanism of TCP flows ensures that the contention on the channel and collision probability stay low, with the result that the system throughput does not degrade much with increasing number of clients.

The analysis results of the scaling of TCP downloads have also been backed up by experimental studies [4], [3]. These papers show that the TCP's data/ack clocking mechanism allows TCP downloads to scale to over hundred clients without any significant degradation of aggregate system throughput. However, the experiments in these papers consider only long running TCP flows and emulated user traffic on testbeds of homogeneous nodes. In contrast, our measurement study conducted with several tens of users trying to download files using TCP shows that TCP download does not scale as well in the context of real user traffic.

Several researchers have reported some subset of the problems we have encountered in our measurement study, and suggested several techniques to address these problems. Prior work [5], [6], [7] has considered the problem of asymmetry between uplink and downlink traffic in WLANs. When a large number of users are downloading traffic over the WLAN, most traffic is downlink. However, the AP that delivers all the downlink traffic has to contend for the channel with the other clients, resulting in an unfair allocation to the downlink traffic. To solve this problem of asymmetry, these papers propose several MAC-layer enhancements to prioritize the AP's channel access. For example, WiFox [5] prioritizes AP's channel access over the clients dynamically depending on the load in the network. The AP accesses medium with high priority when AP's transmission queue size is high and accesses with default priority otherwise. As a result, WiFox claims to give 400-700% increase in throughput and 30-40% improvement in average response time. Our work differs from WiFox and other related work in that we identify and address several factors (besides asymmetry between uplink and downlink) that contribute to poor TCP download performance in dense scenarios.

Other research [8] has observed the effect of channel contention on the RTT of TCP flows through a WLAN. The authors show that highly variable RTTs due to contention lead to incorrect estimation of TCP retransmission timeout, and hence lead to spurious retransmissions. The authors propose prioritization of TCP ACKs as a solution to address this problem. Researchers have also observed the impact of channel contention on bit rate adaptation [9], [10] in dense deployments, and proposed solutions to prevent lowering of bit rate unnecessarily in response to collisions. While each of the above papers measure and analyze a subset of problems that arise in a dense WiFi network, none of them have reported all of the problems or their complex interplay we find in our measurements. To summarize, our work improves over prior work on improving TCP performance in real-life dense scenarios by analyzing the problem more comprehensively, and identifying interplay of all the factors that contribute to poor performance.

III. Measurements in a live classroom

A. Measurement setup

We first describe our data collection method for measurements in a real classroom. In a course with 124 registered students, taught by one of the authors, a subset of lectures involved downloads of supplementary instruction material by students, and some involved graded quizzes. The students used individual laptops and tablets, and some desktops as well, for these activities. Our setup consisted of students connecting to a web server that hosts instruction or quiz content. A small fraction of students used wired access. We had three enterprise-grade WiFi APs, setup in the three non-overlapping 802.11g channels 1, 6, and 11. All the relevant entities were on the same extended LAN.

The activity was as follows. The students browsed to the content server, authenticated themselves, and downloaded a variety of content (video lectures, references, quizzes) over the wireless channel as instructed. We instrumented the web server to log the per-request service time. In addition, we also collected network traces from two vantage points: (i) The WiFi AP was instrumented to collect per-frame MAC layer statistics. Our code had access to hardware registers in the WiFi NIC, that let us determine the fraction of airtime that was spent in transmissions, receptions, and in idle listening at a very fine granularity of 250ns. (ii) A sniffer running tcpdump was connected via an Ethernet hub to the content server to collect TCP and HTTP logs.

Prior to our measurements, we ensured that the WiFi AP, the web server, and the wired backhaul from the AP to the web server were not loaded. That is, the performance seen by the clients in all our measurements was constrained by the wireless network bottleneck. External WiFi interference was minimal.

From all our measurements, we choose one representative dataset to present results from: a quiz conducted in class. In the quiz, 94 students, spread roughly equally over 3 APs, downloaded a quiz question paper of size $\simeq$200KB. A subset of 24 students also downloaded the optional reference material file of size $\simeq$4MB. We pick one of the three APs to present results from; the results at the others were similar. This AP in question served 32 students: all 32 students downloaded the quiz file, and 17 students downloaded the reference material.

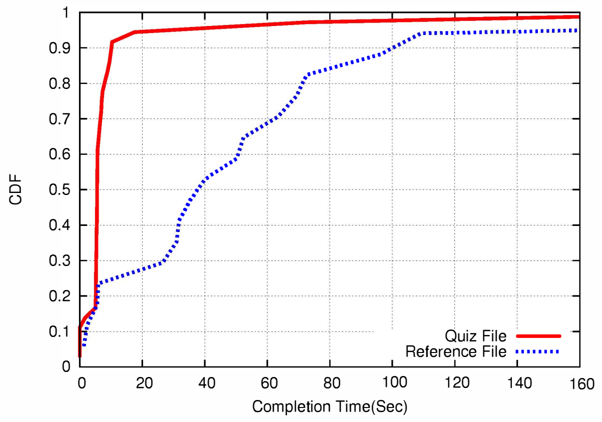

Fig. 1: CDF of the time taken to download the quiz and references files.

B. Results

First, we present the most important performance metric – the completion time, since this delay determines how users perceive the quality of the network. The completion time is measured as the time from the issue of HTTP GET request for the particular file to the last packet of the download received by the user. Fig 1 shows the CDF of the completion times for all the clients, for both the quiz and reference files. To put these numbers in perspective, let us calculate the expected download time. First, note that our classroom was such that even the farthest client could comfortably operate at the highest 54Mbps datarate of 802.11g, when operating in isolation (we verified this during AP placement). This physical layer data rate translates to about 24Mbps of TCP-layer throughput, after accounting for link-layer overheads and the overheads of TCP ACKs. If we go by prior work that claims that TCP download throughput scales perfectly with the number of clients, each client should have gotten a TCP throughput of 24Mbps/32. Assuming all clients downloaded both the quiz and reference file, which we overestimate as 5MB worth of content per client, the expected download time still works out to only about 54s. In contrast, the highest completion times in Fig 1 was 229s for the quiz file, and 478s for the reference file[1]!

Next, we investigate why the completion time was so bad. Upon looking at the TCP time-sequence graphs, we found that some clients suffered severe TCP segment losses, and often timed out several times during the course of the measurement. Fig 2 shows the average TCP retransmission rate (averaged across all clients every 2s) as a function of time; most of these retransmissions were due to a TCP timeout. Fig 3 shows the TCP time sequence graph of a client that experienced multiple TCP timeouts.

To understand why the application-layer performance was so bad, we analyze the logs collected at the AP to understand the MAC-layer performance in the network. Using our custom instrumentation of the AP driver, we determined what fraction of the airtime was reported as "busy" at the AP. This air occupancy percentage is shown in Fig 4 for the duration of the quiz download. Also shown is the aggregate download

[1]This of course created logistical problems; the instructor had to give time extensions to those students who experienced delay in downloading!

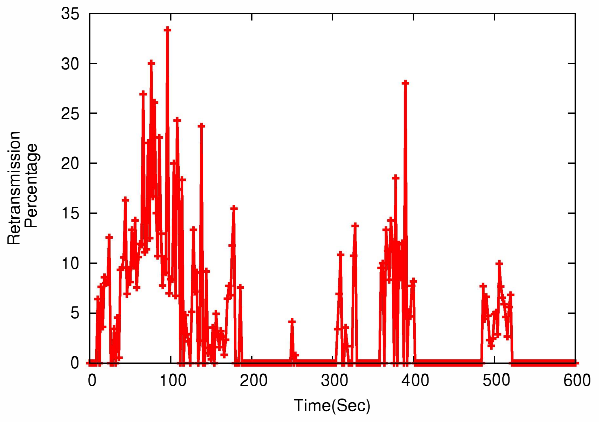

Fig. 2: The average TCP retransmission rate across all clients vs. time.

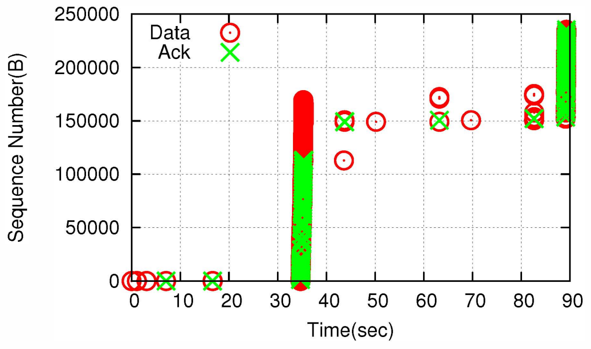

Fig. 3: TCP time sequence diagram of a client that experienced multiple timeouts.

throughput of the AP during this time. Both measures are shown as two-second averages. We see from the figure that there are periods where the channel is busy, and there are also periods where the channel is idle for large fractions of time. This behavior fits well with our earlier observation of TCP timeouts. We also note that, irrespective of the channel busy percentage, the aggregate throughput is poor most of the time.

We further analyze the AP's logs to determine what caused the AP to deliver such low throughput, even when the channel was busy. We verified that the signal strength at all clients was good enough to support high bit rates. The other possibility is that of collisions, due to multiple clients picking the same backoff counter and transmitting in the same slot during CSMA MAC's channel arbitration mechanism. Collisions are notoriously hard to detect using packet logs, because collisions often result in a synchronization error at the physical layer, thereby leaving no trace in any kind of packet tracing mechanism. So, we use hardware registers exported to the device driver to determine the amount of "wasted" airtime at the AP, defined as the amount of time spent in one of the following activities: (i) transmitting packets that failed to elicit a link-layer ACK, (ii) receiving packets which could not be successfully decoded (either due to synchronization error at the physical layer, or a CRC error after synchronizing with the transmission). Fig 4 also shows this wasted airtime percentage at the AP as a function of time. This figure shows that around 10–20% of the AP's airtime is often wasted, possibly due to collisions on the channel.

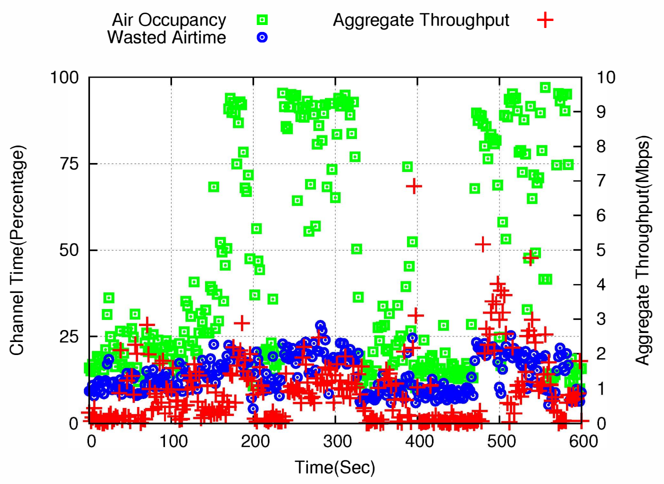

Fig. 4: Aggregate throughput, air occupancy, and wasted airtime at the AP.

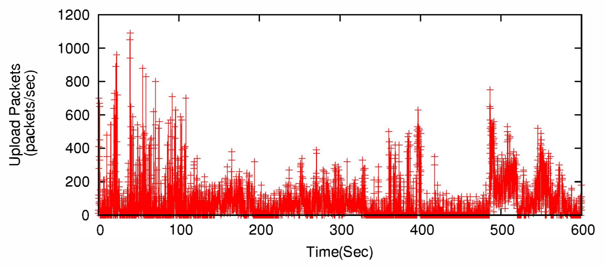

Fig. 5: Upload traffic (pkts/s) generated by clients during the quiz.

We next investigate why the contention and collision rate on the channel was so high. Prior work, as discussed in Section II, shows that if the only traffic on the channel is TCP data and ACKs, the contention on the channel should be very low. However, in our measurement, we found that there was a small amount of extra upload traffic besides the TCP ACKs. Figs 5 and 6 show the rate of upload traffic in packets/sec and in kbps respectively. These metrics are shown as averages over 100ms intervals; this indicates the burstiness of the upload traffic. This traffic consists of GET requests for various embedded objects on the course webpage, traffic generated in navigating the authentication page, TCP handshake packets for the multiple connections the browser opens, and some small amount of extra background traffic likely generated by email clients, Dropbox, and other such applications. Note that the amount of upload traffic is very low, averaging at about 8kbps in aggregate across all clients at the AP. However, it appears that this traffic was enough to increase the contention on the channel, and cause collision losses.

The contention on the channel due to a large number of active clients is further exacerbated by the interaction with the bit rate adaptation. It is well known in prior work that most rate adaptation algorithms mistake collision losses for poor signal on the channel, and lower the bit rate in the hope of increasing the probability of packet delivery. However, transmissions at lower bit rate take up more airtime, further

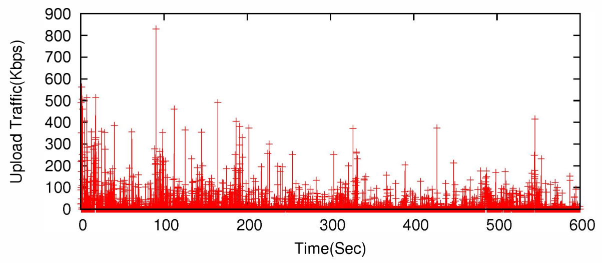

Fig. 6: Upload traffic (kbps) generated by clients during the quiz.

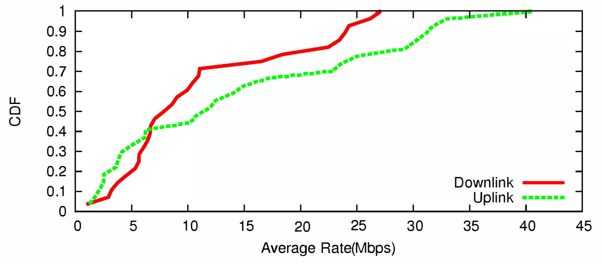

Fig. 7: CDF of the time-averaged bit rates of clients in the uplink and downlink directions.

increasing the contention on the channel. Fig 7 shows a CDF of the time-averaged bit rates of the clients during the quiz. We see that most clients were operating at a very low average rate, suggesting that the rate adaptation algorithms were using lower rates upon observing losses.

In addition to the metrics reported above, the overall experience of the students in using the WiFi network was very bad. When the students were simultaneously downloading large files and stressing the wireless networks, students often complained that their WiFi was not responding. We found many instances where a driver reset was needed to get the WiFi interface to work, even after the contention had subsided. We conjecture these to be possible device driver bugs that were triggered under the high loss rate situations we encountered in class. It is likely that such situations are not well tested in client driver code.

Are collisions due to the CSMA MAC protocol mechanisms alone enough to explain the high losses we saw in our measurements? Or were there any other factors at work? The limited control to vary parameters and monitor performance in a live measurement makes it difficult to answer some questions, which we seek to address with a combination of simulations and controlled experiments in the lab in the next section.

IV. Simulations and Controlled Experiments

In this section, we describe several results from simulations and controlled experiments conducted in the lab to better understand the classroom WiFi measurements.

A. Simulation

As described earlier, accurately measuring collision losses in an experiment is a hard problem. Even using multiple wireless sniffers on the air cannot guarantee that we can identify the error rate due to collisions, because the sniffers themselves may fail to decode most collisions. Therefore, we resort to simulations, where we can instrument the simulator to identify collisions.

TABLE I: Simulation setup

Parameter	Value
WiFi Protocol	802.11g
Rate adaptation algorithm	Minstrel
AP queue size	512
Number of clients	30
Download size	5MB
Upload traffic	10Kbps

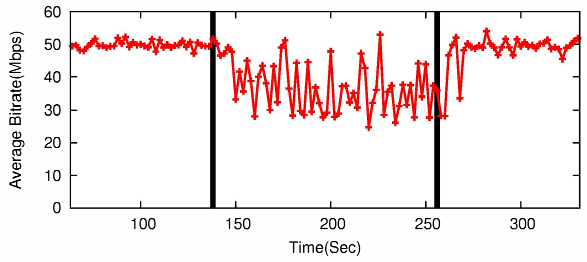

Fig. 8: The bit rates chosen by a WiFi client in the presence of contention.

We use the ns-3 network simulator to perform simulations. Our simulation model consists of 30 802.11g WiFi clients connected to an AP. To simulate the traffic seen in the classroom, each client downloads a 5MB file from a server via the AP, while uploading CBR traffic at the rate of 10 kbps over a TCP connection. The clients are placed close enough to the AP to eliminate the possibility of any channel losses, so that a packet loss occurs only if two clients pick the same backoff value and collide during the CSMA MAC operation. We have used the Minstrel rate adaptation algorithm, and a queue size of 512, as used in the real AP. Table I summarizes the parameters in our simulations.

Our simulation resulted in a download time between 292 and 435 seconds for the clients, which roughly matches the download performance seen during the classroom quiz. During the download, the collision rate on the channel (defined as the fraction of airtime on the channel that was wasted in collision) was between 15% and 25%, proving that the poor TCP performance was primarily due to collision losses on the channel.

B. Controlled experiments

Next, we perform several small-scale controlled experiments in the lab to better understand the impact of collisions and contention on the channel. We first try to understand the impact of collision losses on the rate adaptation algorithm. We set up an experiment where a WiFi client (a Linux laptop with a Qualcomm Atheros QCA9565 / AR9565 Wireless Network Adapter (rev 01) device driver) is downloading a large file via the AP. After 2 minutes, we introduce 14 other WiFi clients on to the channel. These clients are Microtik single-board computers, that generate TCP upload traffic at the rate of 100kbps. After a further 2-minute duration of high channel contention between the 15 WiFi clients, we

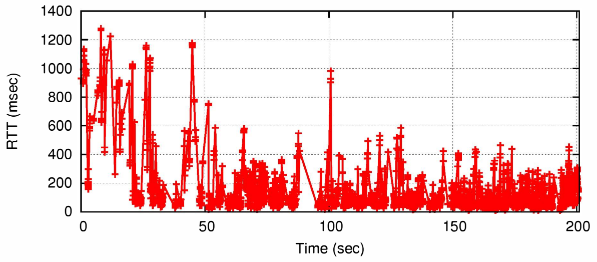

Fig. 9: Highly variable RTTs of a TCP flow caused by high channel contention.

turn off the Microtik boards, and let the laptop traffic run for some more time. Fig 8 shows the time-averaged bit rate (averaged over 2 sec) of the laptop in the upload direction for the duration of this experiment. We observe from the figure that the rate adaptation algorithm lowers its bit rate due to collision losses, as expected. Further, we note that the rate adaptation algorithm takes around 12 seconds (after the Microtik boards are turned off) to recover from the effects of channel contention, which is approximately the timescale at which popular bit rate adaptation algorithms adapt rates. Similar results were observed with 3 other laptops running 3 different device drivers as well. This recovery time of the bit rate adaptation algorithm has a significance in explaining our measurement data of the previous section. The bursty upload traffic seen in our experiment has several periods of quiet between bursts of high upload activity. However, due to the relatively long recovery time of the rate adaptation algorithm, the effects of contention (i.e., the lowering of transmit bit rates) persist even in the periods between upload bursts, magnifying the effect of the small amount of upload traffic.

Next, we identify the impact of collision losses on TCP performance. We setup an experiment with 21 WiFi clients (7 laptops and 14 Microtik single-board computers) connected to an AP. The clients download a 9MB file from a server connected to the AP. The clients run a browser emulation script that generates around 50 GET requests to download several embedded objects in a sample webpage. In addition, the clients also generate a bursty upload TCP traffic at an average rate of 200 kbps, to simulate other background application traffic. This mix of upload and download traffic created enough contention on the channel to slow down the TCP downloads, and we observed several of the effects noticed in the classroom measurements. A wireless sniffer over the channel reported that 22% of frames decoded were marked as link-layer retries. While this is not a true indication of the collision rate on the channel (as the sniffer may have missed capturing several frames due to synchronization errors caused by collisions etc.), it gives us enough indication that the loss rate due to collisions is substantial.

High channel contention also leads to variable delays in transmitting a packet (a transmission may succeed without collisions sometimes, but may require several unsuccessful attempts and multiple backoffs some other time). For example, Fig 9 shows the highly variable TCP RTT of a single client in the controlled experiment with 21 clients described above. Sudden RTT variations confuse the TCP's RTO estimation algorithm, and may lead to TCP timing out unnecessarily, while the data packet is still in transit. In fact, we collected client-side logs in the experiment above and noticed 61 instances of spurious TCP timeouts and retransmissions (summed across all the clients), where the original transmission and the subsequent retransmission were both received by the client after a long delay. While we could not collect client-side logs in the classroom, we did notice highly variable RTTs and expect several of the TCP retransmissions seen were in fact unnecessary.

Finally, we could reproduce the phenomenon of client device drivers becoming non-responsive under high contention with several different laptops and device drivers in our controlled experiments as well.

Note that we have used our controlled experiments to verify that there were no other causes of packet loss introduced by the wired channel or the AP in our measurements. We repeated our experiments with APs from two popular vendors, and obtained similar results. We also verified that there was no buffer mismanagement at the AP. For example, with vendor supported AP logs we verified that the AP buffer always had enough packets to transmit over the wireless link, and that buffer underflow was not the reason for poor throughput.

V. Our Solution Approach

We now describe our solution, WiFiRR, that seeks to improve the worst case completion time of large TCP downloads in a dense WiFi setting like classrooms.

A. *Design of WiFiRR*

The measurements and analyses of the previous sections lead us to conclude that high channel contention is root cause for poor TCP download performance. To address this problem, we seek to limit the number of clients contending for the wireless channel at any instant by modifying the AP's MAC-layer scheduling policy. One can plug in any MAC scheduling policy which ensures fairness, we are using FIFO with round-robin. Our solution, WiFiRR, is designed a modification to access point driver code that manages the AP's buffers. Normally, APs transmit packets belonging to all clients from its buffer in a FIFO manner. With WiFiRR, an AP designates a subset of K out of the total N clients as "active" during a given time slot T. Whenever the AP gets a chance to transmit, the AP looks through its queue and preferentially picks packets to these K active clients, skipping over packets from non-active clients in the queue. The AP varies the set of K active clients in a round-robin fashion in every slot, so that every client eventually gets a chance to make progress. The AP transmits broadcast and management frames normally, as per their turn in the queue. Of course, the AP does not keep the link idle: if there are no broadcast frames or frames to active clients, it will transmit frames to non-active clients.

Now, in a time slot T, since we suppress downlink TCP data and ACK packets for the marked inactive clients, their TCPs will go quiet for the duration of the slot, and the uplink traffic from the clients is greatly suppressed as well. This leads to a lower number of contending nodes, fewer collision losses, and eventually, better TCP performance for the active clients. Our solution does not require any change at the clients.

While our solution leads to a temporary stalling of the non-active clients, the overall performance improves over large

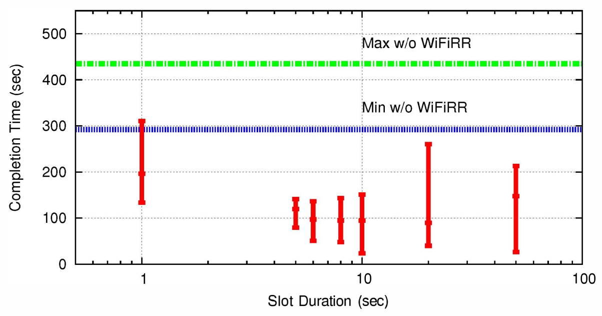

Fig. 10: Min, median, max download completion times vs T (K fixed at 5).

TCP downloads, because the clients will experience much better network conditions once they become active. We realize that our solution may not be beneficial in cases where the TCP flows are very short or interactive, because the negative effects of stalling may overweigh the benefits in such cases. Enhancing the core ideas of WiFiRR to better handle short and interactive TCP flows is part of ongoing work.

B. Results

We implemented WiFiRR in ns-3. We use the simulation scenario as described in Section IV-A to evaluate our scheme: a simulation of 30 clients performing a large download (of 5MB each), while simultaneously generating a low rate upload traffic. We evaluate the performance gains with WiFiRR in this scenario.

We experimented with different values of slot duration T and number of active clients K. Fig 10 shows the minimum, median, and maximum download completion times over 30 clients with WiFiRR, where the slot duration is varied from 1 sec to 50 sec, and the number of active clients K is fixed at 5. The green and blue lines also show the minimum and maximum completion times without WiFiRR. We find that the worst case completion time reduces to 137 sec from 435 sec (3.2$\times$ reduction) when the slot duration is 6 sec. Intuitively, the optimal choice of slot duration is dependent on the RTT of the flows. The slot duration should be in the order of few RTT in order for TCP flows to adapt to the change in channel contention. The average RTT of the TCP flows in our simulation was around 1.2s due to high contention, resulting in a slot duration of 6s working best[2].

Fig 11 shows a similar comparison of completion times with and without WiFiRR, where we set the slot size T to 6s, but vary the number of active clients K. Here, $K = 5$ leads to the best performance of 3.2$\times$ reduction in worst case completion time. We have also verified that WiFiRR leads to a lower rate of collisions. While the average rate of collisions on the channel was 15-25% without WiFiRR, the collision rate with WiFiRR (6s slot and 5 active clients) was 2-3%.

We also compare our solution WiFiRR (with 6s slot duration, 5 active clients) to WiFox [5], another solution that aims to improve TCP download performance in dense scenarios by addressing the asymmetry between uplink and

[2]Making the WiFiRR approach practical would require devising a mechanism to determine the slot duration as a function of the RTT.

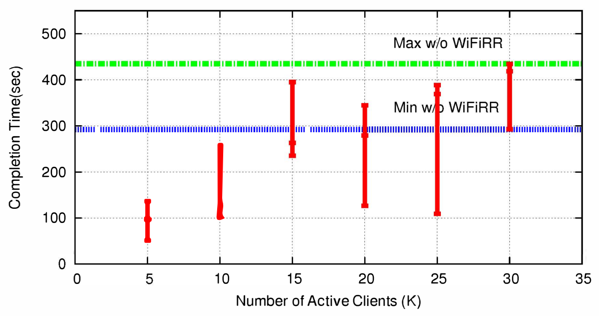

Fig. 11: Min, max, median download completion times vs K (T fixed at 6s).

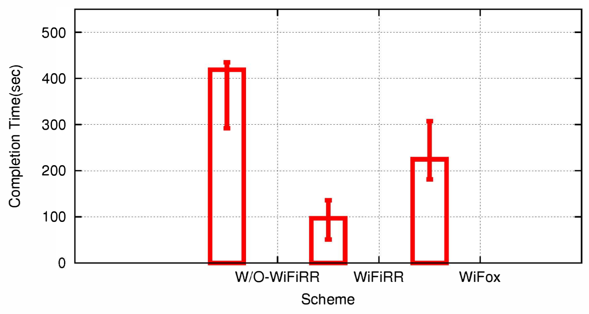

Fig. 12: Min, median and max download completion times.

downlink traffic (see Section II). We implemented WiFox in ns3, as per the specification in [5]. We made AP use 802.11e, and implemented the high and default priority channel access settings as described in [5]. In WiFox, time is divided into intervals of size T', each of which is divided into n slots. In every time unit T', the AP accesses the channel in a prioritized fashion for $k \leq n$ slots. The mapping between the priority level k and the queue size of the AP can be logarithmic, exponential, linear and logistic. We use the logistic mapping as it gave the best results for WiFox. We also tuned the maximum queue size (from 50 mentioned in the paper to 512) as it resulted in better completion time for WiFox in our simulation setting. We set T to 100ms, n to 10 slots, and set the k range to be 0-10. Fig 12 shows the download completion times with WiFox, as compared to the cases with WiFiRR and without WiFiRR. While WiFox does lead to improved performance over the base case without WiFiRR, WiFiRR outperforms WiFox by 2.25$\times$ in our scenario of large TCP downloads.

VI. Conclusion and Future Work

This paper presented measurements of TCP download performance in a dense WiFi scenario of WiFi-enabled classroom, where students download quizzes and instruction material over WiFi. Our results show that TCP download performance degrades significantly with increased user density, much more beyond what is to be expected from prior work. We analyze the reason for this poor performance and find that the small amount of background upload traffic that coexists with the TCP download traffic in real life causes an increase in contention on the wireless channel. The subsequent collision losses trigger undesirable behavior in other protocols: the bit rate adaptation unnecessarily lowers its bit rate, TCP gets confused by the highly variable RTTs and performs spurious retransmits, and device drivers perform unexpectedly under such losses. We also propose a solution, WiFiRR, that improves the performance of large TCP downloads in a dense scenario. Our solution operates as a scheduler at the AP buffer, and restricts the number of active clients contending for the channel at any instant by selectively transmitting packets to different subsets of active clients over different slots.

Going forward, we will make WiFiRR adaptive to short or interactive flows. We plan to build and deploy a prototype of WiFiRR, to make our classroom teaching using WiFi more effective. Finally, we believe that future standards of WiFi must also focus on addressing this very real performance bottleneck of contention on the wireless channel in dense usage scenarios, in addition to just improving the peak data throughput at the physical layer, to provide a better experience to the end user.

VII. Acknowledgement

This work was supported in part by AirTight Networks.

References

[1] R. Bruno, M. Conti, and E. Gregori, "Modeling TCP Throughput Over Wireless LANs," in *Proc. 17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation*, 2005, pp. 11–15.

[2] G. Kuriakos, S. Harsha, A. Kumar, and V. Sharma, "Analytical Models for Capacity Estimation of IEEE 802.11 WLANs using DCF for Internet Applications," *Wireless Networks*, 2009.

[3] M. A. Ergin, K. Ramachandran, and M. Gruteser, "An experimental study of inter-cell interference effects on system performance in unplanned wireless LAN deployments," *Computer Networks*, 2008.

[4] S. Choi, K. Park, and C.-k. Kim, "On the Performance Characteristics of WLANs: Revisited," in *Proc. SIGMETRICS*, 2005.

[5] A. Gupta, J. Min, and I. Rhee, "WiFox: Scaling WiFi Performance for Large Audience Environments ," in *Proc. CoNEXT* , 2012.

[6] E. Lopez-Aguiler, J. Casademont, J. Cotrina, and A. Rojas, "Performance Enhancement of WLAN IEEE 802.11 fot Asymmetric Traffic," in *Proc. The International Symposium on Personal, Indoor and Mobile Radio Communication*, 2005.

[7] X. Wang and S. A. Mujtaba, "Performance enhancement of 802.11 wireless LAN for asymmetric traffic using an adaptive MAC layer protocol," in *Proc. VTC*, 2002.

[8] D. Malone, D. J. Leith, A. Aggarwal, and I. Dangerfield, "Spurious TCP Timeouts in 802.11 Networks ," in *Proc. Wiopt*, 2008.

[9] P. A. K. Acharya, A. Sharma, E. M. Belding, K. C. Almeroth, and K. Papagiannaki, "Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach ," *SECON*, 2008.

[10] K. V. Cardoso and J. F. de Rezende, "Increasing throughput in dense 802.11 networks by automatic rate adaptation improvement," *Wireless Networks*, 2012.

[11] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function ," *IEEE Journal on Selected Areas in Communications*, 2000.

[12] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New Insights From a Fixed-Point Analysis of Single Cell IEEE 802.11 WLANs ," *IEEE/ACM Transactions on Networking*, 2007.