Achieving flow isolation in 802.11 networks with the DTT scheduler

Rosario G. Garroppo, Stefano Giordano, Stefano Lucetti, and Luca Tavanti

Dipartimento di Ingegneria dell'Informazione, Università di Pisa
Via Caruso, 16 - Pisa, 56122 - Italy
Email: {r.garroppo, s.giordano, s.lucetti, luca.tavanti}@iet.unipi.it

Abstract. Though the IEEE 802.11 standard has reached wide acceptance, its main access function, the Distributed Coordination Function (DCF), still suffers from some relevant problems coming from the specific features of the wireless channel. By means of simulation, we analyse the performance anomaly and the "inter-flow blocking" problems, highlighting the mechanisms that generate them. Starting from these insights, we propose a simple centralized channel aware scheduling algorithm, named Deficit Transmission Time (DTT). The basic principle under the DTT is measuring the channel quality in terms of frame transmission times. This measurement is then used to take scheduling decisions that guarantee each downlink flow an equal time share of the channel. The proposed scheduler has been developed and deployed in a Linux-based prototype AP to experimentally evaluate its performance. The results clearly show the improvements introduced by the DTT in terms of flow isolation and reduction of the effects of the performance anomaly.

1 Introduction

The IEEE 802.11 standard [1] has by now become the most popular technology for broadband wireless local area networking. As a corollary, we have seen a steep rise in the efforts to increase system capacity and, more lately, a growing interest in supporting service differentiation. Nevertheless, one of the most critical factors driving the efficiency of 802.11 networks is still the ability to overcome the hurdles imposed by the wireless channel. The capacity of the links is highly variable in both time and space, thus leading to unpredictable frame delivery ratio and times. The same 802.11 DCF mode of operation, coupled with the simple "First-In First-Out" (FIFO) strategy employed at the access point (AP), does not deal effectively with this problem. In fact, the multi rate capability and the induced uplink/downlink unbalancing [2] are two aspects that contribute in hindering network performance.

The 802.11 Distributed Coordination Function (DCF) has been designed to offer the same long term channel access probability to all stations. In ideal conditions, this translates into an equal portion of the overall effective bandwidth for each user [3]. On the other hand, as soon as some frame is not correctly received, 802.11 provides for the retransmission of the frame, with the possibility

of applying a rate fall-back algorithm. This kind of algorithms are freely designed by the manufacturer of each device, that defines the rules for employing more robust but less efficient modulations for the successive transmissions (e.g. the 1 Mbps DBPSK). Such retransmission attempts occupy the channel at the expenses of all other stations. Consequently these stations experiment a sensible reduction in their available bandwidth, regardless of the state of their own links. The performance of the network is thus driven by the station under the worst channel condition. This phenomenon, known as the "performance anomaly" of 802.11 [4], is more general, and also occurs if two (or more) stations simply uses two different transmission rates (e.g. 2 and 11 Mbps, or 11 and 54 Mbps in a mixed 802.11b/g network).

The space-time variation of link quality also has another consequence. Most commercially available APs use a single FIFO queue: all the frames stored after the currently served one must wait for it to be dequeued. This event occurs after either a transmission has been completed successfully or the retransmission limit has been reached. In both cases, these periods are influenced by the state of the channel, or, more precisely, of the link toward the destination station. Thus, the transmission delay experienced by each frame does not depend solely on the number of frames enqueued in front of it, but also on which station those frames are addressed to and in what condition those links are. This issue, which we refer to as "inter-flow blocking", becomes a noteworthy problem especially when there are large or bursty transfers of data, e.g. FTP. These packets, addressed to the same station, might build a continuous bulk in the queue of the AP, hence monopolizing the medium for a long time.

Many researchers agree that a way to overcome the performance anomaly and the inter-flow blocking problems might be a smart scheduling algorithm to be deployed at the AP. Indeed, several schedulers for centralized wireless networks have already been proposed in literature [5][6][7]. Most of them rely on a model of the wireless channel. The links between the AP and the user devices are independent of each other and subject to bursty errors according to a two-state Markov model: link quality is either good (i.e. error-free) or bad (faulty). Unfortunately, this model is sometimes far from the actual channel behaviour, and a system based on it can easily lose its gain. For that reason, a more reliable solution would be centering scheduling decisions on a real measure of the link state.

Starting from these observations, we have designed a centralized scheduling algorithm able to take into account the actual channel behaviour. The main innovation of our scheduler is the way it measures the link quality. This is not appraised with usual metrics, such as signal-to-noise ratio, but is quantified as the time needed to deliver a frame to the destination. Hence, the resource to share is not the total capacity of the channel, but the time the channel is in use. We will show that this approach leads to the important property of flow isolation.

The paper is organized as follows. The scheduler is described in the next Section and evaluated in Section 3. In particular, after an outline of the testbed

configuration, Subsection 3.2 introduces the simulation methodology and the obtained results and Subsection 3.3 presents the experimental trial. A thorough discussion of the work and its outcome can be found in Section 4. Finally Section 5 concludes the paper.

2 Description of the DTT scheduler

The architecture of the DTT scheduler is depicted in Figure 1. The whole framework is inserted on top of the MAC layer, which is in no way modified. A classifier splits outgoing traffic into several queues, on the basis of the destination MAC address. Hence all frames held in the same queue are addressed to the same station. A "bucket" is associated to each queue to account for the over-the-air time of the previous transmissions. As an example, consider a network with three users (refer again to Figure 1). The scheduler will therefore create a queue and a bucket for each associated station (left, center, right). At the end of every frame transmission, the scheduler computes the Cumulative Frame Transmission Time (CFTT). The CFTT comprises all the time spent for the transmission, including all retransmission attempts, backoff and idle periods. The CFTT is also produced when the retry limit is reached with a transmission failure. Given this definition, it can be seen that the CFTT, embracing all the factors that limit the maximum frame rate, is a deterministic and real measure of the link state.

The CFTT is then used to drain the bucket associated to the destination of the transmitted frame. Let us assume that the MAC layer has just completed a transmission of a frame for the station associated to the queue on the right. Then the rightmost bucket is drained with CFTT tokens. Next, this same value is equally divided by the number of non-empty queues and the result loads the related buckets. The bucket whose frame has just been sent, if non-empty, is included in this count. This is needed to give all the queued frames the same treatment. All the buckets coupled with an empty queue are cleared (set to zero) to avoid that some stations, idle for a long period of time, keep a credit/debit that would reduce short-term fairness once they have some new frames to transmit. In our example all queues are non empty, hence the CFTT is divided by three and each bucket is added CFTT/3 tokens. Afterwards, the scheduler picks the next frame from the queue whose associated bucket is the fullest (the one on the left of Figure 1) and hands it over to MAC layer. Only one frame at a time is sent to MAC, and the next frame is not sent until the previous transmission is over. This allows the scheduler to make a precise computation of the CFTT and avoids the MAC buffer to hold any other frame but the one under delivery.

A few things are worth noting. First, the tokens are not related to transmission times with complex formulae, but with a simple and direct one-to-one relationship. Thus they are exactly a transmission time, or a fraction of that. Second, the tokens are an actual measure (not an estimate, nor a prediction) of the channel quality. More retransmissions, possibly at lower bit rates, are needed to deliver the frame to the stations whose channel quality is poor. As a consequence, such destinations get their buckets emptied by larger amounts of

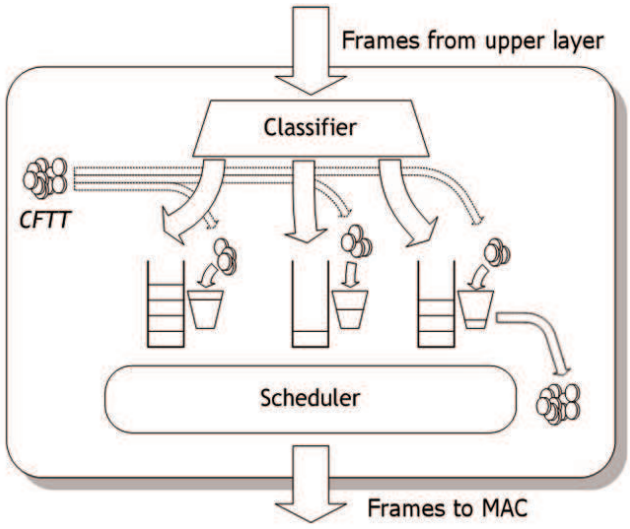

Fig. 1: The architecture of the DTT scheduler.

tokens, thus having to wait longer before being chosen for the next transmission. On the contrary, easily reachable destinations get their buckets drained slightly and so they need to be idle for shorter intervals. Under these rules, it is clear that the bucket with more tokens is also the one whose associated queue has transmitted less. The name of the scheduler, Deficit Transmission Time (DTT), aims at reminding just this concept.

3 Performance evaluation

The proposed scheduler has been evaluated via simulation and experimental trials. The same general scenario, described in the next Subsection, has been used for both kinds of test. The simulation mainly addressed aspects that are difficult to reveal or analyse through a real testbed. In particular, Subsection 3.2 concentrates on the relation between time on air and retransmissions. A short outline of the simulation tool is also given. Then Subsection 3.3 presents the experimental results.

3.1 Description of the test-bed

All the trials have been performed with the reference scenario of Figure 2. One access point is connected with a wired link to a server that generates UDP and TCP traffic. The AP then transmits data packets toward two or three stations, depending on the specific try. Traffic is only in the downlink direction, from the AP to the stations. The only exceptions are TCP control packets sent back from the stations to the server. To create various channel conditions, stations

are placed at different distances from the AP and in most cases move around. The experimental trials have been run in a typical office environment, whereas the simulation assumed free space propagation.

All stations are equipped with IEEE 802.11b network cards, providing a maximum theoretical throughput of 11 Mbps. For the simulation, a bit rate adjustment strategy has been implemented at MAC layer. In case a frame transmission fails, the bit rate is decreased to 5.5 Mbps, then 2 Mbps and, finally, 1 Mbps. This last value is kept until an ACK is received or the retransmission limit is reached. An analogous strategy is also running on the laptop computers used in the experiment. The RTS/CTS mechanism is disabled in all cases.

Further details are then given when describing each specific experiment.

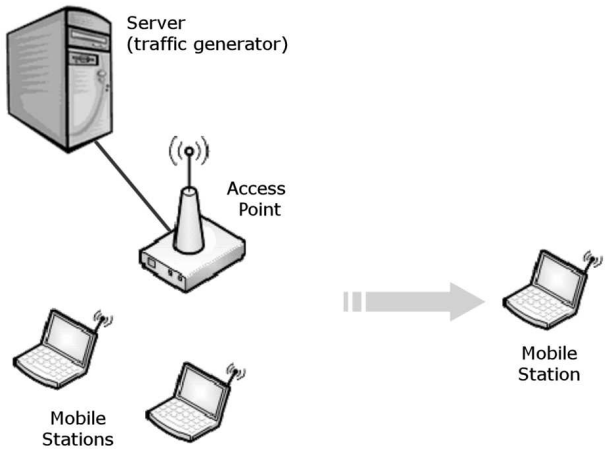

Fig. 2: Reference test-bed for simulation and experimental trials.

3.2 Simulation analysis

We have chosen to employ the OMNeT++ simulator, version 3.0b1 [8], which we have integrated with the Mobility Framework (version 1.0a3) developed at the Technical University of Berlin [9], and with an accurate 802.11b MAC layer, which we have built from scratch. Given that both the simulator and its parts are not yet well established in literature, we carried out some validation tests, comparing the results with known models. Specifically, our baselines were an analytic formula for the maximum achievable throughput with a single transmitting station [10] and Bianchi's performance study [11].

As to the first model, the maximum observed difference between the theoretical value and the simulation is in the order of 0.1%, which can be considered negligible. A very similar behaviour has been observed with regard to Bianchi's

model. Figure 3 presents the results for a network of 10 stations working in saturation conditions. As it can be seen, the simulator matches pretty closely the theoretical values. Therefore we can conclude that the accuracy level of the employed tool is enough to guarantee reliable trials.

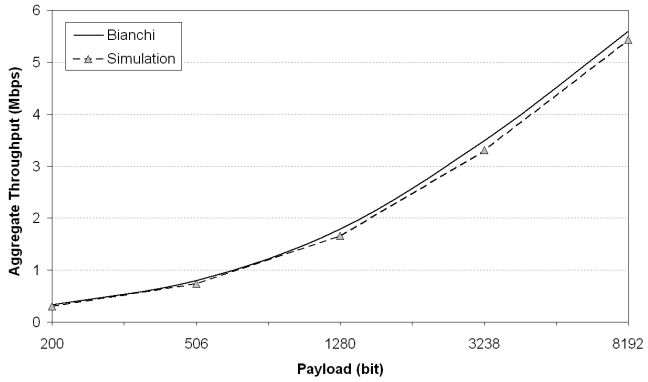

Fig. 3: Validation of the simulation tool: throughput versus payload size for a network of 10 stations transmitting at 11 Mbps.

Simulation has been used to examine the correlation between time on air and number of frame retransmissions, for both a simple FIFO based AP and our scheduler. The test was run with two stations, both static, and each receiving a continuous UDP stream of 5 Mbps, with packets of 1440 bytes (same traffic of the experimental tests). The first station (say "A") is very close to the AP and is almost always reached at the first attempt; the other station (say "B") suffers poor channel conditions, and often needs a high number of retries. The time on air is updated after the completion of each transmission cycle, being each increase computed in the same way as described in Section 2 for the CFTT. In other words, each increase is the CFTT of the last frame. The retransmission limit has been set to 4, as suggested in the standard for the used frame length.

Figures 4 and 5 plot the total time on air (lines, left axis) and the number of retries (points, right axis). The simulations run for 60 seconds, but, to have a better insight of the events, only the first 250 ms are shown. In both figures, the time on air looks like a flight of stairs, whose steps are very high for B and very low for A. In correspondence of each step, there is a point (cross or diamond) referring to the number of retries for that frame. The bigger the number of retries, the worst the quality of the link, the higher the step. This is a confirmation of the goodness of the CFTT in accounting for the channel behaviour. By the way, when the number of retries is 4, the frame has been dropped. Note that the relationship between the height of the step and the number of retries is not linear. This is because the CFTT also includes the backoff periods, which are aleatory.

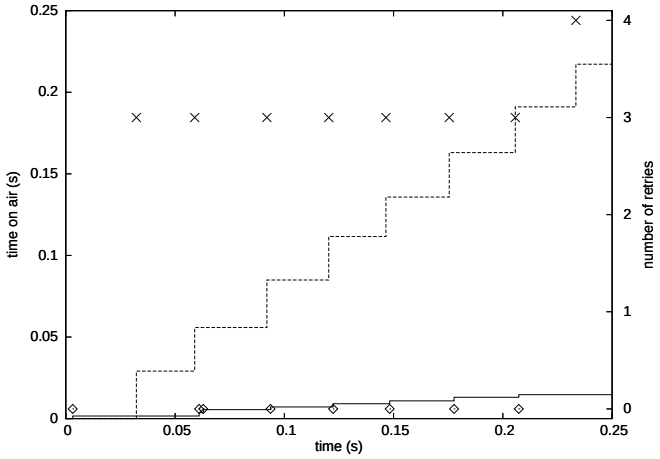

Fig. 4: Detail of time on air (lines) and number of transmission retries (points) for the FIFO discipline. The solid line and the diamonds refer to the nearest node (A), the dashed line and the crosses refer to the farthest node (B).

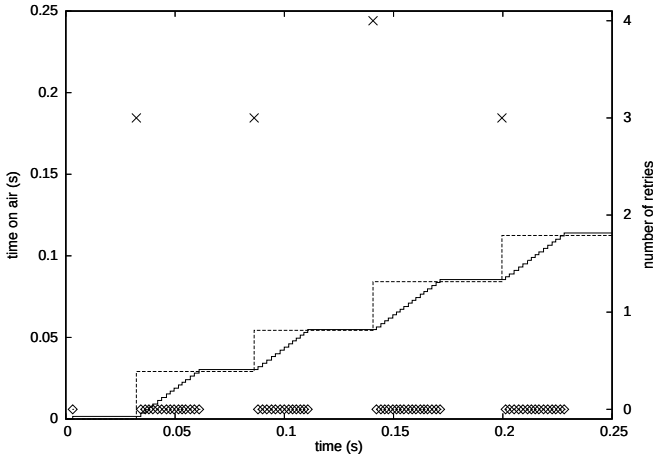

Fig. 5: Detail of time on air (lines) and number of transmission retries (points) for the DTT scheduler. The solid line and the diamonds refer to the nearest node (A), the dashed line and the crosses refer to the farthest node (B).

Some more things are worth nothing. The first is that, in Figure 4, the stairs for B are much steeper than those for A, while the number of steps is roughly the same. This means that the FIFO discipline alternates transmissions toward the two stations, but delivering a frame to B takes much longer than delivering it to A. Thus, FIFO allows the station in the bad position to occupy the channel most of the time. We can see that the time on air for B is almost 90% of the total. Clearly, since most of this time is wasted in transmission

failures and control procedures, we can only expect a very low throughput. On the contrary (see Figure 5), the DTT compensates this unfairness sending more frames to A. For example, at 0.14 seconds, a frame addressed to B is dropped after four retransmissions. This corresponds to an overall channel occupation time of around 30 ms, that includes the time on air, the expired timeouts and the backoff periods. On the other hand, delivering a frame to A takes less than 2 ms. The disparity is evident. Thus, after the far station has received its frame, the DTT scheduler may let the AP transmit around 15 frames to the near station. B can receive its next frame once the allocated times have reached the same level (at around 0.2 s). That is how A's stairs achieves the same steepness of B's, but with much more steps.

Secondly, far less points (diamonds in particular) appears in Figure 4 in comparison to Figure 5. That means that far less frames have been transmitted. Therefore, even without drawing the throughput, we can infer that the FIFO discipline produces a (much) lower aggregate throughput than DTT. This statement will be proved in the next Section, when describing the outcome of the experiments.

3.3 Experimental trials

For the experimental measures, we have built a software framework to embed in a prototype AP. The scheduler is implemented as a Linux kernel module. It intercepts the packets arriving at the device driver, and stores them internally. It dynamically creates and manages as many queues and buckets as the number of registered stations. The packets are then sent back to the driver one at a time, according to the scheduling decisions. The driver finally passes each frame to the physical device (the WLAN card), which dispatches it over the medium. The driver has also been modified to notify the scheduler of completed frame transmissions and reached retry limits. This is necessary to compute the CFTT.

The scheduler has been installed on a laptop computer, that acts as the prototype AP. For the FIFO discipline we used a commercial AP, that natively implements a single FIFO queue. In the prototype AP, we have also reduced the timeout to force disassociation to 30 seconds (from the default 6 minutes) to have a faster response of the system. This is not strictly related to the DTT scheduler, but it is an optimization that aims at quickly releasing the bandwidth of stations that have clearly become unreachable. The commercial AP is employed with no modifications. UDP traffic has been created with the MGEN traffic generator [12]. Packets are 1440 bytes long. TCP traffic has been produced via a simple file transfer.

In a first try, the AP casts UDP traffic towards two stations. Each UDP downlink flow is at 5 Mbps, therefore the network works in saturation conditions. At the beginning all stations are in a good position. Then, each of them alternatively moves away from the AP, and the quality of the link starts degrading until the AP decides to disassociate it. After a while, the station re-enters in the AP coverage area and reassociates.

Figure 6 plots the throughput seen by two stations when the commercial AP is employed. In this figure the anomaly of 802.11 is clearly visible. As long as the two stations are both close to the AP, they equally share the available bandwidth. However, once one station starts moving away (at around 70 s), the other, that still has a good link, is dragged into bandwidth shortage. Only when the far station is disassociated (at 180 s), the other can exploit the full potential of its link. Note the 6 Mbps peaks in the throughput: these are present in the following graphs as well, and are merely due to the backlogged packets.

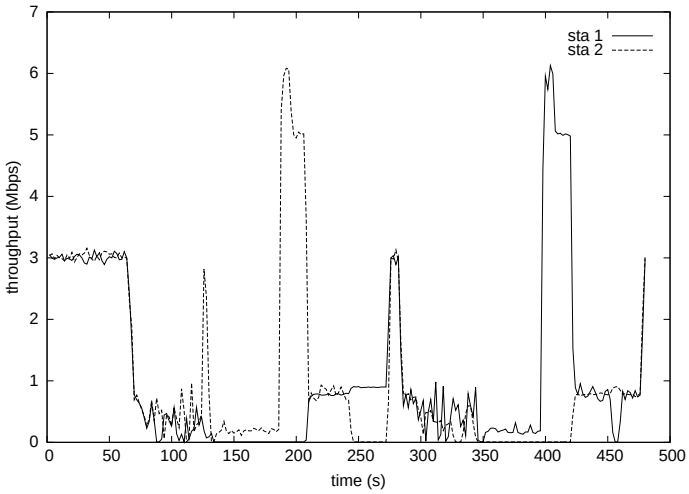

Fig. 6: Throughput seen by two stations receiving UDP traffic with the commercial AP.

The improvement introduced by the DTT scheduler is visible in Figure 7. The scenario is the same as before. Again, no difference exists while the stations are close to the AP. Then, when a station is brought progressively farther, the other does not suffer in any way and its throughput remains constant. It can only be noticed a small increment in the fluctuations of throughput that occur when the far station is about to be disassociated (at around 180 s and 380 s). In that case, all frames reach the retransmission limit, thus occupying the channel for the longer time possible. As it happens for the FIFO policy, once the far station is disassociated, the close one gets all the available bandwidth.

Figure 8 presents the outcome for a slightly more complex network. Three terminals have been moved alternatively closer and farther from the AP. As it can be noted, the behaviour of the system is similar to what previously described. The scheduler constantly keeps the throughput of nearby stations to high levels, penalizing the stations that go away. This result definitely increases the confidence on the goodness of DTT for more complex scenarios. Proving large scale

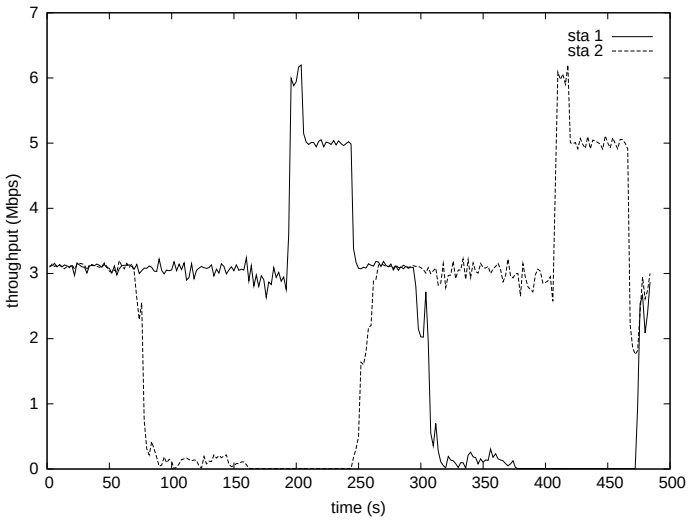

Fig. 7: Throughput seen by two stations receiving UDP traffic with the DTT scheduler.

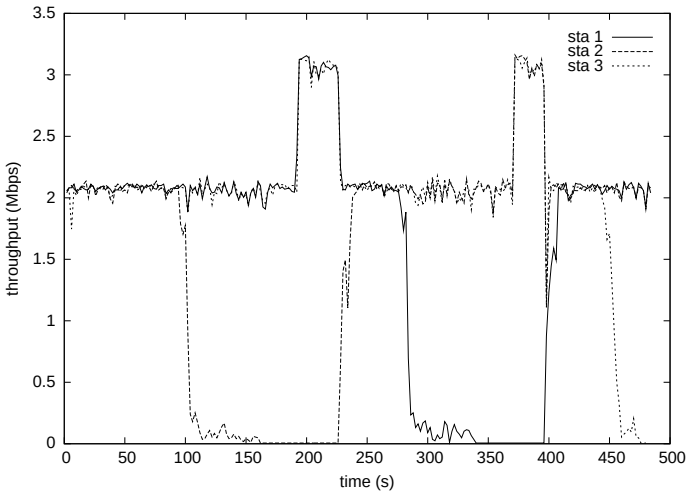

Fig. 8: Throughput seen by three stations receiving UDP traffic with the DTT scheduler.

scalability, however, is outside of the scope of the paper (also consider that the set of possible patterns of movements, timings and traffic load is extremely vast). The interested reader may refer to [14], which reports the results of a simulation analysis for a large number of VoIP connections.

We have then examined the behaviour of the system with TCP traffic. Two stations are still the sink of two UDP flows, while a third station is the destination

of a file transfer. As usual, all the stations go in turn far from the AP. Figure 9 and 10 refers to the commercial AP and the DTT scheduler, respectively. Note that for the commercial AP we had to lower the rate of the UDP flows to 1 Mbps, i.e. well under the saturation threshold. This is because the TCP session could not even start when injecting the usual 5 Mbps UDP traffic per station. The reason is that when the unique queue gets saturated, TCP starts the congestion control procedures, thus reducing its throughput. UDP traffic however keeps its packet rate constant and quickly monopolizes the queue, eventually causing TCP to starve.

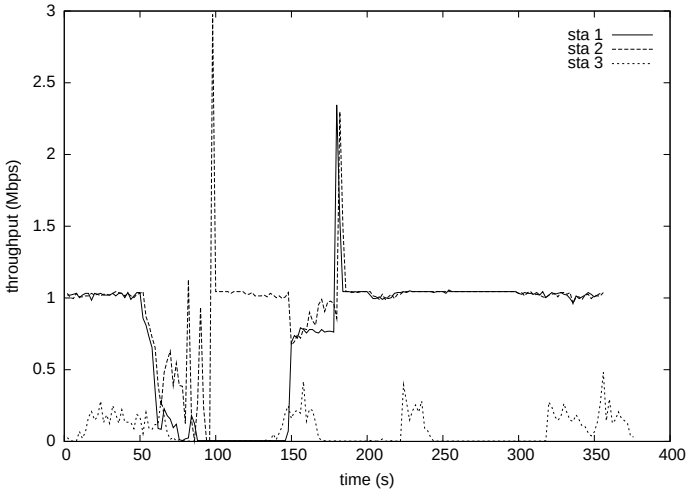

Fig. 9: Throughput seen by two stations receiving UDP traffic (1 & 2) and one station receiving TCP traffic (3) with the commercial AP.

In spite of the reduced load, the TCP session still has troubles even in sustaining itself. Furthermore, when one of the two UDP-receiving stations moves away, the throughput of TCP falls to zero. In this scenario the impact of our scheduler is even more impressive. The network can handle the original 5 Mbps UDP flows, granting each of them one third of the available bandwidth. Thus each UDP flows settles at slightly less than 2 Mbps, while the TCP settles at around 1.25 Mbps. Moreover, as expected, the TCP session is not influenced by the movements of the other stations and vice versa.

The last thing to point out about this test is the presence of a hidden uplink flow. This flow is made of all the TCP acknowledgement packets sent by the TCP station to the AP. The effect of this flow is twofold. It reduces the overall available bandwidth (the sum of all the flows is clearly less than the 6.2 Mbps registered in the previous tests) and produces some more fluctuations in the

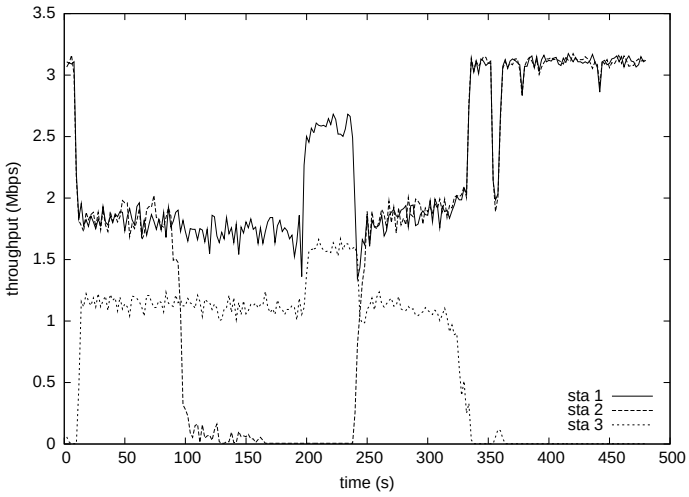

Fig. 10: Throughput seen by two stations receiving UDP traffic (1 & 2) and one station receiving TCP traffic (3) with the DTT scheduler.

throughput. Both effects disappear as soon as the TCP station moves away (at around 320 s) and the throughput stabilizes at the same levels of Figure 7.

Finally we present the results for a "side test" showing how the presence of multiple queues is not sufficient to prevent the performance anomaly. In the prototype AP, we substituted the DTT scheduler with a simple round robin discipline. Queues are always served in the same order, regardless of the state of the channel and of previous history. Figure 11 plots the throughput for a three station scenario. If we compare this plot with Figure 6 for the commercial AP, we can note just a marginal improvement for the stations near the AP. The undesirable anomaly induced by the far node is still unequivocally present.

4 Discussion

From the trials, it has clearly emerged that the presence of just one station with a poor link to the AP damages all the stations in the network. As mentioned in Section 1, the FIFO-driven AP tries to supply all stations with the same bandwidth. Unfortunately this try gives no benefits neither to the farthest users, whose connection is hampered by the propagation conditions, nor to the other users, whose throughput is brought to the values of the worst station. This is the well known "performance anomaly". We have shed light on it from two different points of view: simulative and experimental (see Figures 4 and 6). From the first, we have highlighted that letting the worst user control the channel results in a high degree of unfairness. In particular, it can happen that, in the worst conditions, the poorest link uses (and wastes) up to 90% of the bandwidth.

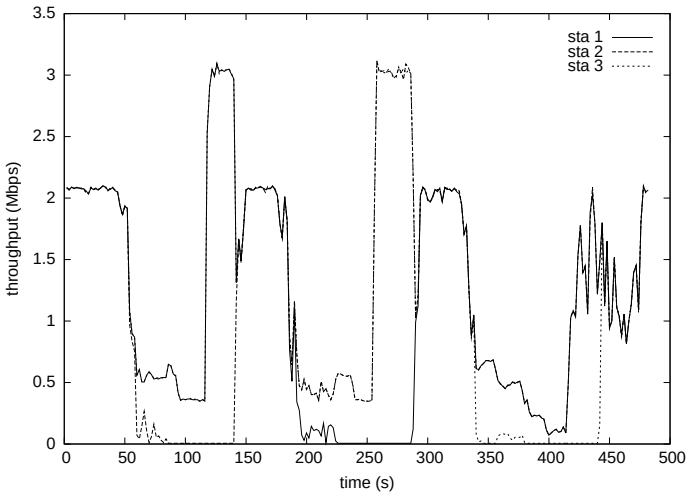

Fig. 11: Throughput seen by three stations receiving UDP traffic with a round robin scheduler.

Hence it was rather easy to infer that network efficiency could only be reduced. The experiments confirmed our deduction. Finally, we have also seen that TCP traffic is very sensitive to this problem (Figure 9).

We have then shown that using multiple queues does improve the performance of the system. However, if the algorithm to choose the next frame to transmit is not smart enough, the improvement is quite marginal. This is the case of the round-robin discipline (Figure 11). The increased complexity does not yield much better results than the original system. Consequently, inserting an "intelligent" scheduler on top of the 802.11 card is a necessary step to significantly improve network capacity. The DTT is such a scheduler.

Figures 7, 8, and 10 prove that the DTT can separate the links toward each station. Therefore, while the far users see their bandwidth severely reduced, the closer stations can still perceive the channel in a good state, thus being able to fully exploit their links. The improvement is noteworthy, and becomes impressive when applied to TCP traffic (Fig. 10). A TCP connection can be kept at high throughput levels from the start to the finish, even if other bandwidth-hungry flows are loading the network. This is true not only under the performance anomaly scenario, but also in ideal conditions, i.e. when all stations have a perfect link to the AP (and the same transmission rate). As mentioned in Section 3.3, the inter-flow blocking brought by the unique FIFO queue induces the TCP to starve under high network loads. Since the DTT splits the traffic in several queues, it can equally divide the bandwidth among all stations. The TCP benefits from this strategy as it is not influenced by the traffic injected by the other flows.

We have already mentioned that, when employing TCP traffic, some packets (the acknowledgements, ACKs) go in the uplink direction. The number and size

of ACKs is limited, and the traffic is strongly unbalanced towards the downlink (that is much more crowded than the uplink). This model is also pretty close to real conditions, as most applications (e.g. browsing the Internet, downloading files) generates much more traffic in the downlink direction. We have seen that the DTT still works fine. The confirmation is in Figure 10. The effect of the ACK packets is in the fluctuations in the throughput (besides the reduction in the overall bandwidth, of course).

The remarkable performance of DTT can be explained with the principle that drove its design. Radunovic et al. [13] have found that a desirable objective for network planning should be achieving "proportional fairness" in bandwidth usage. Proportional fairness is a metric that represents a good compromise between two extreme goals: maximum network efficiency (i.e. only the best link is used) and maximum fairness (i.e. equal average rate to all flows). By the way, maximum fairness is the goal of the basic 802.11 strategy. Furthermore, in a multi rate environment like the one generated by different quality of the links, proportional fairness is realized when all the stations use the channel for the same time [3]. A corollary of this proposition is that the flows are isolated: each station gets the maximum of its bandwidth share regardless of the others. And this is exactly what the DTT achieves.

5 Conclusions

The paper has firstly presented a simulation analysis that brought to light the phenomena leading to the performance anomaly and the inter-flow blocking problems of 802.11 networks. Then, starting from these findings, we have proposed the DTT, a simple centralized channel-aware wireless scheduler. The scheduler is based on an indirect but reliable measure of the channel quality: gross frame transmission times. This measure is used to take decisions on the frame to be transmitted next. The final goal is to grant each flow the same amount of channel occupancy time. This allows the DTT to isolate the different traffic flows, letting each flow obtain its downlink channel time occupancy share irrespective of the radio channel quality experimented by the others. In order to prove its feasibility, we have developed a Linux-based prototype AP based on the DTT scheduler. The outcome of the experimental trials has confirmed the efficiency of the proposed solution. In particular, the experimental results pointed out that a system based on the DTT can brightly overcome the well-known performance anomaly of IEEE 802.11 networks and reduce the negative effects of the inter-flow blocking problem at the AP queue.

The presented scheduler has therefore multiple advantages. It runs a simple algorithm, which does not need complex channel modelling or heavy computations. We can therefore envision that it could be hosted on all 802.11 based devices. Moreover it is transparent to both the applications and the network interface cards, thus allowing to use all the deployed hardware without modifications. A simple software/firmware upgrade would be enough to make the APs far more efficient.

Finally, with the introduction of some weights when distributing the tokens, the DTT may offer a suboptimal solution to the downlink traffic differentiation issues of legacy 802.11b APs. Further studies are also planned to change the nature of DTT from centralized to distributed. This will allow an extension of its appealing features to all traffic flows, i.e. in both the downlink and uplink directions. This can be very useful when non-802.11e systems are expected to support delay-sensitive services, such as VoIP.

Acknowledgements

This work has been carried out within the framework of the TWELVE Project, funded by the Italian MIUR Ministry by means of the COFIN research program.

References

1. ANSI/IEEE Std 802.11, 1999 Edition. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.
2. S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, P. Sinha, *Understanding TCP Fairness over Wireless LAN*, Proc. of IEEE Infocom 2003, San Francisco, April 2003.
3. L. B. Jiang, S. C. Liew, *Proportional Fairness in Wireless LANs and Ad Hoc Networks*, IEEE Wireless Communications and Network Conference, March 2005.
4. M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, *Performance Anomaly of 802.11b*, Proc. of IEEE Infocom 2003, San Francisco, April 2003.
5. Y. Cao, V.O.K. Li, *Scheduling algorithms in broadband wireless networks*, Proceedings of the IEEE, Volume 89, Issue 1, Jan. 2001, pp. 76-87.
6. P. Bhagwat, A. Krishna, and S. Tripathi, *Enhancing throughput over wireless LANs using channel state dependent packet scheduling*, Proc. of InfoCom 1996, March 1996, pp. 1133-1140.
7. X. Liu, E. K. P. Chong, N. B. Shroff, *Opportunistic Transmission Scheduling With Resource-Sharing Constraints in Wireless Networks*, IEEE Journal on Selected Areas in Communications, Vol. 19, Issue 10, Oct. 2001, pp. 20532064.
8. The OMNeT++ discrete event simulation system. Available at *http://www.omnetpp.org*.
9. The Mobility Framework for OMNeT++. Available at *http://mobility-fw.sourceforge.net/hp/index.html*.
10. R. G. Garroppo, S. Giordano, S. Lucetti, F. Russo, *IEEE 802.11b Performance Evaluation: Convergence of Theoretical, Simulation and Experimental Results*, Networks 2004, Wien, Vol. 1, pp. 405-410.
11. G. Bianchi, *Performance analysis of the IEEE 802.11 distributed coordination function*, IEEE Journal on Selected Areas in Communications, Volume 18, Issue 3, March 2000, pp. 535547.
12. B. Adamson and H. Greenwald, *MGEN User's and Reference Guide*, 2004. Available at *http://mgen.pf.itd.nrl.navy.mil/mgen.html*.
13. B. Radunovic, J. Le Boudec, *Rate Performance Objectives of Multihop Wireless Networks*, IEEE/ACM Trans. on Mobile Computing, Vol. 3, Issue 4, pp. 334349, Oct. 2004.

14. R. G. Garroppo, S. Giordano, S. Lucetti and L. Tavanti, *A measurement-based channel aware scheduler to lessen VoIP capacity degradation in 802.11 networks*, submitted to ICC 2006 and available at: *http://netgroup-serv.iet.unipi.it/reports/dtt-voip-icc06.pdf*.