T^2-Fair: A Two-Tiered Time and Throughput Fair Scheduler for Multi-Rate WLANs

Kwan-Wu Chin
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
kwanwu@uow.edu.au

ABSTRACT

Low throughput due to unfairness is a key problem in multi-rate wireless local area networks. To promote fairness and hence throughput, T^2-Fair groups flows according to their average data rate, provides each group fair time allocations and ensures throughput fairness for flows in each group. Since each group is allocated transmission times fairly, T^2-Fair isolates high and low rate groups and prevents system capacity from being degraded by low rate flows. We have derived T^2-Fair's performance bounds analytically and investigated its performance using the *ns-2* simulator in various scenarios with a mix of high and low rate flows. Our results show that T^2-Fair is effective in isolating and providing proportional throughput fairness to these flows.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer Communications Networks

General Terms

Wireless LANs, Scheduling, QoS

Keywords

Wireless Fair Queueing, Multi-Rate, Deficit Round Robin

1. INTRODUCTION

Current wireless local area network (WLAN) technologies offer a multitude of data rates. For example, the IEEE 802.11a [7] standard offers up to eight data rates; 6 to 54 Mbps. The best data rate to use depends on the channel condition before transmission and is affected by various factors. In particular, path loss and interferences from other access point (APs) or devices operating on the same frequency. Other factors include multi-path, device mobility and obstructions, e.g., people [13]. The dynamic link variations thereby require different data rates to be used such that packet error is low and throughput is maximized.

A key problem in multi-rate WLANs is the performance anomaly [2] [16] that occurs due to the disparate data rates used to combat link variations, which result in the throughput of a high-rate flow converging to that of the flow with the lowest data rate. The key reason is due to lower data rate flows taking up an unfair and significantly higher share of the wireless channel. Consider two flows transmitting a 1024 bytes packet. If the flows have a data rate of 54 and 6 Mbps, each flow will occupy the channel for 151 and 1365 μs respectively. Thus, the 54 Mbps flow's throughput will be reduced to 5.4 Mbps since the 6 Mbps flow takes approximately 10 times longer.

A significant factor that determines a device's link quality is path loss. Given a WLAN with devices connecting from different locations, and hence distances to the AP, devices at a given distance or location from the AP will more or less experience similar link conditions. An observation here is that when high and low rate devices compete, traditional wireless fair queueing (WFQ) algorithms cannot be used to guarantee fairness because a device that is far away from the AP is unlikely to catch up in terms of bytes sent since the data rate used is likely going to be low. Unless the low rate flow's link quality improves, by moving closer to the AP for example, it will lag behind high-rate flows. Therefore, for this reason, conventional WFQ algorithms cannot achieve long term throughput fairness in multi-rate WLANs.

To address the above problem, T^2-Fair provides time and proportional throughput fairness at the system and flow level. It makes use of two observations. First, in a WLAN, at a given point in time, there will be devices with similar link quality or data rate. Since their data rate is similar, this group of devices can be viewed as operating in a single-rate WLAN, thus conventional WFQ algorithms can be used to provide throughput fairness. Secondly, to address the performance anomaly reported in [2], it provides time fairness between flows and ensures high-rate flows are compensated according to the time occupied by low rate flows.

To make use of the observations above, we design T^2-Fair to be a two tiered scheduler. At the first tier, the scheduler tracks and groups devices based on their average data rate or signal-to-noise value. T^2-Fair then allocates each group the same "air-time", calculated using the lowest data rate group in the current round. For each group, T^2-Fair runs a throughput wireless fair scheduler called DRR^w, an exten-

MSWiM'06, October 2–6, 2006, Torremolinos, Malaga, Spain.

sion of the well-known deficit round-robin (DRR) scheduler [15] revised to operate in wireless networks. Each device is serviced according to their deficit counter as long there is sufficient time left in the group. Otherwise, operation is suspended until more time is allocated.

To justify the performance of T^2-Fair, we have extended the analysis of DRR [15] to consider error prone channels and derive new performance bounds. Also, we have conducted extensive experiments using the *ns-2* simulator which we have extended to simulate an IEEE 802.11a WLAN. Our experiments involving a mix of high and low rate flows conducted over a realistic radio propagation model show that T^2-Fair effectively prevents low rate flows from dominating the wireless channel and ensuring flows in the same group are throughput fair.

In the next section, we will elaborate further the problems and observations discussed above. Then in Section 3, we present how T^2-Fair groups devices and how it guarantees time fairness, and proportional throughput fairness using DRR^w. In Section 4, we discuss the *ns-2* implementation of T^2-Fair and MAC modifications needed to achieve good performance. We present our results in Section 5 from experiments involving a realistic radio propagation model in scenarios consisting of a high-rate flow competing with two low rate flows. In Section 6 we highlight our contributions to existing literature before concluding in Section 7.

2. THE PROBLEM

Figure 1 shows an example IEEE 802.11a [7] WLAN with data rates ranging from 6 to 54 Mbps. To ensure the best data rate is used, the link adaptation algorithm is responsible for probing the channel and selecting an appropriate data rate that yields the lowest bit error rate (BER). For example, in RBAR [6], a receiver measures the signal-to-noise (SNR) value of the incoming request-to-send (RTS) message, and returns the measured SNR in the clear-to-send (CTS) message. From the SNR value, the sender selects the best data rate that yields the lowest BER to send the outgoing data packet.

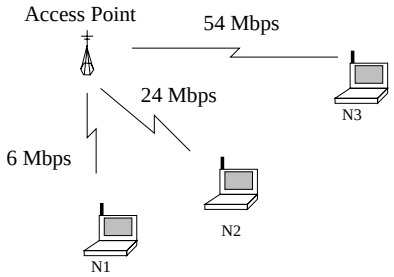


Figure 1: An example WLAN using IEEE 802.11a.

The advantage of having wide ranging data rates is that a device can operate in different channel conditions. Unfortunately, at the expense of other devices. The primary reason for this unfairness is that each device occupies the channel for varying amount of time. For example, in Fig. 1, N_1 and N_3 occupy approximately 1365μs and 152μs of "air-time" respectively. As a result, the data rate of N_2 drops to 5.4 Mbps. This effect is well documented in [2] and [16]. These papers showed via analysis and experimentation that throughput drops significantly due to unequal data rates or "air-time" consumed by each device. In fact, all devices within a WLAN will have a throughput that is equal to that of the lowest data rate device.

Another issue to consider when designing a scheduler is the path loss or long term channel condition of devices within a WLAN. This issue impacts whether compensation is feasible at all, as carried out by current WFQ algorithms, e.g, [10], that aim to provide long-term throughput fairness. This observation becomes more severe if we consider systems with wider data rate ranges, for e.g., DS-UWB [9] provides data rates that range from 28 Mbps to 1 Gbps. The wide discrepancies between supported data rates mean that a low rate flow is unable to achieve a throughput comparable to a high rate flow in the long term. In other words, the strong path-loss component means that a device that has a data rate of 6 Mbps is unlikely to attain long-term fairness if it competes with a device transmitting at 54 Mbps. Of course, one could starve the 54 Mbps device until the 6 Mbps device catches up, however, this would likely violate the 54 Mbps flow's QoS agreement. Alternatively, the AP can increase its transmission power to improve the SNR of disadvantaged devices, thus their data rate. The problem however is that radio propagation is affected by environmental factors, therefore, increasing transmission power may not necessary improve a device's link quality.

Aside from the issues above, scheduling algorithms must also take into account unfairness caused by delays resulting from retransmissions and link adaptation. The behavior of the MAC and link-adaptation algorithm will affect how much time is used to transmit a given packet. For example, the IEEE 802.11b will retransmit a packet four and seven times depending on whether RTS/CTS is used. Coupled with the link adaptation algorithm, the "air-time" consumed will be different for each device at different points in time. Further, to ensure reliability, the link adaptation algorithm may reduce a device's data rate for each retransmission. Thus, increasing the "air-time" required to transmit a packet, which consequently affects the throughput of other flows.

Conversely, the link adaptation algorithm may be able to transmit a packet at a higher data rate, thus saving "air-time". For example, assume a device has been allocated a time slot sufficient to transmit one packet at 6 Mbps. Before transmission, the link adaptation algorithm finds that the channel has improved to 54 Mbps. Therefore, the allocated time slot will be under utilized. Ideally, the scheduler should be able to salvage this remaining "air-time" and allocate it to other flows.

Following our discussions above, it is important that a scheduler is aware of the (a) long term link quality experienced by devices, and (b) the underlying link adaptation and retransmission processes. In the next section, we present a scheduler that takes advantage of these two observations to provide fairness, both time and throughput, to flows in multi-rate WLANs.

3. THE T^2-FAIR SCHEDULER

T^2-Fair provides both time and proportional throughput fairness. To achieve both fairness criteria, it groups devices based on their long term link quality or average data rate and assigns each group fair time shares. This is achieved by having a two tiered scheduling process. At tier-1, T^2-Fair is responsible for ensuring system wide fairness by grouping

devices with the same data rate into groups and allocating each group fair time allocations. Then, at tier-2, it relies on a throughput fair scheduler to guarantee proportional fairness amongst devices in each group. Note, we only consider downlink communications, that is, traffic destined to devices and we leave uplink communications as future work.

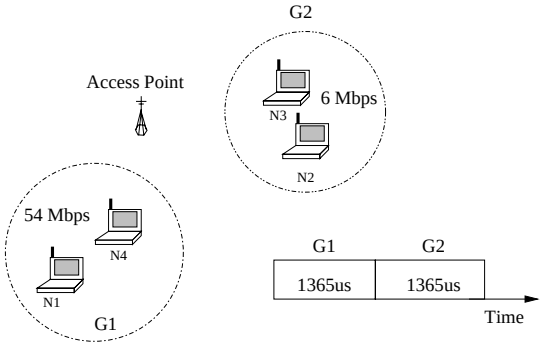


Figure 2: Overview of T^2-Fair. Groups G1 and G2 are allocated the same time allocation, thus ensuring low and high rate flows compete fairly.

Figure 2 shows how T^2-Fair achieves both time and throughput fairness. The WLAN has two groups of devices, classified according to their respective data rate; i.e., 6 and 54 Mbps. Assuming there are backlogged packets and each packet is of size 1024 bytes, the scheduler allocates equal time share to both groups. That is, both groups obtain 1365 μs of "air-time"; ignoring MAC overheads. At each scheduling round, the scheduler determines the group with the lowest data rate, calculates the required "air-time" for that group and assigns the same "air-time" to all other groups which have backlogged flows. In this example, with a time allocation of 1365 μs, G_1 and G_2 have time to transmit one and nine packets at 6 and 54 Mbps respectively.

In Figure 2, notice that each group consists of two devices. This means the time allocation assigned by the scheduler will have to be shared fairly amongst devices within a group. In this example, the effective data rate for the WLAN is 30 Mbps (10 packets every 2730 μs), which equates to G_1 and G_2 having an effective capacity of 3 and 27 Mbps respectively. This means, the amount of bandwidth received by a flow is proportional to its channel condition relative to other flows, hence the term proportional throughput fairness. Also, given that the data rate of devices belonging to each group is similar, one can view each group as a single rate virtual WLAN. Therefore, existing WFQ algorithms can be used to ensure proportional long-term throughput fairness amongst devices within each group.

Another key feature of T^2-Fair is that it works closely with the MAC layer to ensure "air-time" are used efficiently. This means the MAC is given a fixed amount of time to transmit a packet where the MAC and link adaptation algorithm are required to transmit the packet using a data rate that minimizes transmission time. If the packet cannot be transmitted due to insufficient time or that a better data rate can be used, the unused time can be allocated to other devices within a group. Once all backlogged flows within a group are serviced, any remaining time is returned to the T^2-Fair for redistribution to other groups.

3.1 Time Fair Allocation

Time fairness is necessary to isolate flows transmitted at different data rates. T^2-Fair groups flows according to a system's supported data rates. This means there will be eight groups if the scheduler runs over a IEEE 802.11a WLAN. At the start of each round, the scheduler scans through these groups and determine the backlogged group with the lowest data rate.

Figure 3 shows three groups. In each group there are queues belonging to devices D_i. To ensure fairness and isolation, all groups must receive equal time share. For example, the scheduler will have to allocate 1365 μs (time to transmit one 1024 bytes packet at 6 Mbps) to all groups in order to guarantee fairness. I.e., $T_1 = T_2 = T_3 = 1365$ μs.

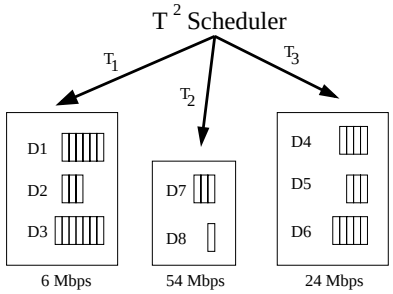


Figure 3: Groups 6, 24 and 54 Mbps. Each group has varying number of devices as reflected by the queues.

The duration of each round is determined by the "air-time" of the lowest data rate group and number of backlogged groups. It is important to ensure that group numbers are small, otherwise throughput will degrade. This is because as the number of groups scale, "air-time" will be spread across multiple groups, thus lowering throughput. The performance of T^2-Fair is therefore proportional to the number of supported data rates, in particular the number of low data rate groups.

To reduce the number of groups, the AP can use data rate ranges instead of having a group for each supported data rate. However, one has to consider the large data rate discrepancies between the supported data rates, thus using data ranges to group devices mean it will be harder for devices to be compensated and achieve long term throughput fairness.

3.2 Grouping Devices

Path loss is a dominant factor that affects a link's signal quality [13]. Therefore, T^2-Fair groups devices based on their average SNR values or data rate. The key observation here is that each device exhibits a unique path-loss and slow-fading characteristics at a given location and distance from the AP, therefore it is logical to group devices that experience similar channel condition. We like to point out that location systems such as [1] have also exploited SNR "patterns" to correlate the location of devices in a building.

When a device joins a WLAN and requests its share of the wireless bandwidth, the scheduler probes a device continuously for t seconds in order to gain an initial average SNR or data rate value. This value is then updated whenever the AP receives acknowledgment packets from the device. If the

device has been idled for δ seconds, the scheduler will probe the device for t seconds in order to refresh its estimate.

To track a device's average SNR or data rate, S^i, the scheduler computes the exponential weighted data rate average of a received packet z as follows,

$$S^i_{z+1} = \alpha S^i_{new} + (1-\alpha)S^i_{z-1} \quad (1)$$

To group devices, the scheduler computes the Euclidean distance to each supported data rates. For example, assuming an IEEE 802.11a system, if a device has an average data rate of 45 Mbps, the device will be placed in the 48 Mbps group. The value of α in Equ. 1 therefore plays a crucial role in determining how fast a device moves to a different group and frequency in which group membership changes.

An alternative method to group devices is to consider data rate variations, using the following equation,

$$S^v_{z+1} = \beta S^v_i + (1-\beta)|S^i_{new} - S^i_z| \quad (2)$$

where β controls how quickly a device is upgraded or downgraded from the current group. This is especially important when we have mobile devices since the scheduler has to quickly re-classify a device base on its link quality at different positions and points in time. Once a device's average data rate deviates ϵ times from the mean, a device is moved to the next higher or lower group, depending on whether its link quality has improved or degraded.

3.3 Proportional Throughput Fairness

Within each group, for each backlogged device, the scheduler determines whether the device should receive service in the current round. The aim here is to provide long term throughput fairness where lagging devices are compensated by leading devices. Here, we denote a flow as lagging if it has not used its allocated transmission opportunity due to channel error. Note that at the group level, given that devices have approximately the same data rate and similar error characteristics, time and throughput fairness are equal.

We extend the Deficit Round Robin (DRR) algorithm [15] to handle wireless errors. The resulting algorithm called DRR^w does not rely on a reference system like existing wireless fair scheduling algorithms to determine whether a flow is lagging or leading.

We begin by introducing some notations. Let F_q and F_q^{HOL} denote the queue for flow q and its head-of-line packet respectively. Each flow is associated with a deficit counter, DC_q, that is incremented by Q_q bits at each round. A round is defined as one iteration of service across all backlogged flows. As in [15], we define $Q = Min_q(Q_q)$, and F_q's share of the channel capacity to be $f_q = \frac{Q_q}{Q}$. Lastly, we denote all backlogged flows as $B(k)$, where k denotes the round number.

Algorithm 1 shows the pseudo-code for DRR^w. DRR^w relies on five functions, $CHN(D_i)$, $DST(p_i)$, $SIZE(p_i)$, $TX(p_i)$ and $PUSH(B(k),k)$. These functions are responsible for determining channel conditions, extracting the packet's destination address, determining packet size, transmitting a packet and saving the current round onto the stack once the time allocated to the group, Q_g, runs out respectively. The $PUSH(.)$ function is required since a round may not complete within Q_g, therefore requiring the current round to be suspended until the group is given more time by the T^2-Fair at a later period.

We make three modifications to DRR. First, DRR^w preserves the deficit counter if a flow experiences channel error; *line-11*. This retains a flow's share of the channel while waiting for its channel condition to improve. Secondly, we limit each flow's deficit counter value to $MAX + \omega_q\widehat{Q_{max_g}}$; *line-3*. This bounds the maximum amount of time a flow uses within each round, thereby ensuring a lagging flow does not lock out other flows when it receives service. The last modification ensures each flow is serviced within time, Q_g; a value allocated by T^2-Fair. It is important to note that T^2-Fair allocates time based on the lowest data rate group, thus the value of Q_g may not be sufficient for a group to finish a round of service. As a result, when Q_g runs out, DRR^w has to suspend its operation and saves the current iteration number k and the set of backlogged flows waiting to be serviced, $B(k)$; *line-8*. Once T^2-Fair provides more time to the group, the current state is retrieved from the stack and DRR^w resumes service.

DRR^w requires the AP determine whether a link is experiencing a fade. To measure link quality, the AP can employ the RTS/CTS exchange to obtain channel information at a target device [6]. If the channel is unable to support the group's data rate, R_g, a flow does not receive service in the current round. Alternatively, the AP can send pilot signals periodically to gain channel information, as in [17].

```
   input: Q_g - time based deficit counter for group g
 1 for F_q ∈ B(k) do
 2     DC_q = DC_q + Q_q;
 3     ζ = MAX + ω_q Q̂_max_g;
 4     if DC_q > ζ then DC_q = ζ;
 5     while DC_q > 0 AND F_q != NULL do
 6         q_size = SIZE(F_q^HOL);
 7         T_time = TXTIME(F_q^HOL, R_g);
 8         if T_time > Q_g then
 9             PUSH(B(k), k);
10         end
11         if q_size ≤ DC_q then
12             if CHN(DST(F_q^HOL)) == GOOD then
13                 T_used=TX(F_q^HOL, T_time) ;
14                 DC_q = DC_q − q_size ;
15                 B(k) \ F_q;
16                 Q_g = Q_g − T_used;
17             end
18         else
19             break;
20         end
21     end
22     if F_q == NULL then DC_q = 0;
23 end
```

Algorithm 1: Pseudocode for DRR^w.

3.4 Link-Layer Considerations

At line 13 of Algorithm 1 we see that the transmit function, $TX(.)$, accepts T_{time} as a second parameter and returns any unused time. The reason for this modification is that the MAC and corresponding link adaptation algorithm causes unfairness. Take for example the Auto Rate Feedback (ARF) [8] link adaptation algorithm. Upon a packet failure, it reduces the current data rate followed by the MAC re-

transmitting the packet at a lower data rate. This results in the packet using significantly more "air-time", where packets of other flows are locked out from transmitting. Thereby, resulting in the well known head-of-line blocking problem [3] which leads to unfairness and reduction in system capacity.

Henceforth, T^2-Fair requires the link adaptation algorithm minimize the use of T_{time}. This means in good channel conditions, a higher data rate must be used, thereby reducing "air-time". The amount of used time is then returned, which T^2-Fair then uses to subtract from Q_g, the time allocated to the group. Note that, given a fixed T_{time}, the MAC cannot retransmit a failed packet multiple times. In other words, T^2-Fair overwrites the default 4-7 retransmission attempts carried out by the MAC in standard IEEE 802.11 systems.

From the above, it is clear that T^2-Fair needs to control how the MAC and link adaptation algorithm use the wireless channel. Without this control, an unfair amount of "air-time" will be used by the channel adaptation process. This is the key reason why T^2-Fair needs to control the MAC using the T_{time} parameter, since doing so provides control of the time allocations given to "bad" devices without sacrificing the QoS of "good" devices.

3.5 Putting It All Together

With DRR^w specified, we now provide a description of T^2-Fair. The basic idea of T^2-Fair, as shown in Algorithm 2. First, the algorithm finds the group with the lowest data rate, G_g. Then, it iterates through all groups with backlogged flows, $G(t)$, and assigns each of them a time slot T^{min} that corresponds to the lowest data rate group. Each group is associated with a deficit counter Q_g that tracks the amount of time given and used. At each round, T^{min} is added to flow g's deficit counter Q_g. Note that the deficit counter is similar to the one used in DRR^w, but counts "air-time" instead of bytes. Further, the quantum value added to Q_g depends on the lowest backlogged data rate group. Lastly, a group with no backlogged flows is removed from $G(t)$.

```
1  while G(t) do
2      G_g = max_{g∈G(t)} SIZE(G_g^HOL)/R_g;
3      T^min = SIZE(G_g^HOL)/R_g;
4      for g ∈ G(t) do
5          Q_g = Q_g + T^min;
6          DRR_g^w(Q_g);
7          if g not backlogged then
8              Q_g = 0;
9              G(t) \ g;
10         end
11     end
12 end
```

Algorithm 2: Pseudocode for T^2-Fair.

Before presenting our simulation methodology, we like to mention that we have derived the upper and lower bounds of T^2-Fair analytically by extending the work of [15] to consider error-prone channels. Due to space limitation, we have omitted these analytical results. Interested readers are referred to [4].

4. SIMULATION

We modified *ns-2* (v2.28) [11]'s IEEE 802.11b implementation to use values from the IEEE 802.11a [7] specification, thus allowing us to simulate a multi-rate WLAN with eight data rates; 6 to 54 Mbps. We use the shadowing radio propagation model included in the *ns-2* simulator. Except for Section 5.3, all our simulations use a standard deviation value (*std_db*) of 4.0 and path-loss exponent of 2.0.

Our experiments are conducted using the network topology shown in Figure 1. In these experiments, N_1 remains stationary and is positioned 5 meters from the AP. Due to its close proximity to the AP, N_1 is able to transmit at a high data rate; i.e., 48 or 54 Mbps. However, for N_2 and N_3, except for experiments outlined in Section 5.3, we increasingly move them away from the AP. This has the effect of lowering their data rates and increase their error rates since their link quality is more likely to drop below the required receive threshold. In effect, N_2 and N_3 will occupy the channel longer as they move away, thus unfairly taking away N_1's "air-time".

In all simulation runs, the AP originates three UDP flows, queued separately, to each of the three devices. All three flows start at the same time. The UDP flow has a rate of 4 Mbps or 64 Kbps depending on experiments. Each flow transmits a maximum of 1000 packets, each 500 bytes in size. For all experiments, we conducted 100 simulation runs and calculated each flow's average throughput and delay. Lastly, at the start of each simulation run, a new seed is used to initialize the random number generator.

4.1 T^2-Scheduler Implementation

In Section 3.4 we argued that the existing IEEE 802.11 MAC suffers from the head-of-line (HOL) blocking problem. To avoid this problem, we made several modifications to the IEEE 802.11 MAC implementation in *ns-2*. We like to point out that T^2-Fair is better suited to TDMA based MACs since each device is allocated strict time slots. Given the wide install base of IEEE 802.11 systems, we will first report on our experimentations conducted using the IEEE 802.11 MAC.

First, we use the RBAR [6] implementation from [14], where receivers inform the AP an appropriate data rate to use via CTS messages. The AP then use the reported data rate to transmit the awaiting data packet. Once a packet is received, the physical layer calculates the difference between the SNR value which the packet is received at and the predefined receive threshold for the data rate which the packet is transmitted at. The difference along with the packet size, and the complementary error function, $erfc(.)$, are then used to determine whether the packet is received in error. Secondly, we modify the values of *ShortRetryLimit* and *LongRetryLimit* to one. This affectively prevents the MAC from retransmitting any packets; RTS or data packets.

Lastly, T^2-Fair buffers a packet for five retransmission attempts. This ensures that when a device is experiencing a bad channel, for example no CTS is received or a data packet fails, the scheduler gives the MAC another go at transmitting the packet in the next round, up to five times. Notice that our implementation does not assume that the AP is able to predict the channel condition of each device. Ideally, if the AP can predict the channel accurately, more time will be saved, thus improving system capacity.

5. RESULTS

5.1 The HOL Blocking Problem

In this experiment, we run a simple round-robin scheduler without consideration for time and throughput fairness. In other words, the scheduler does not compensate a flow if it suffered a channel error nor isolate low and high data rates flows from each other. The main goal here is to quantity the performance degradation due to the HOL blocking problem and performance anomaly due to the use of multiple data rates. In the experiments to follow we will denote Flow-1 to 3 as flows going to devices N_1, N_2 and N_3 respectively.

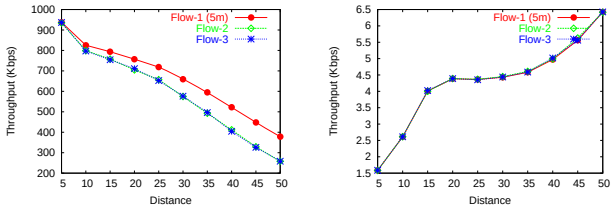
Figures 4(a) and 4(b) show the performance degradation that occurs when using the MAC's default retransmission values, and Figures 5(a) and 5(b) show the benefits of limiting the MAC's retransmission attempts to one. Comparing the results in Figures 4(a) and 5(a), at distance of 50 meters, a 12% improvement in throughput is recorded. This becomes more significant when a flow's link condition degrades, meaning as more transmission attempts are required the HOL problem becomes more prominent. From these figures it is clear that the default MAC parameters result in low throughput and high delay. In particular, Flow-1's throughput degrades along with flows 2 and 3. The main reason is the time incurred due to retransmission attempts and channel contention, especially as N_2 and N_3 move farther away, they consume increasingly more "air-time" due to their lower data rates.

Figures 5(a) and 5(b) show the results from our experiment where we do not allow the MAC to retransmit any packets. Recall that the scheduler will buffer a packet for five retransmission attempts. The advantage here is that if a MAC finds a device is having a bad channel, due to a previously unsuccessful transmission, it gives another device, which may have a good channel, a chance to transmit. In this scenario, since N_1 (Flow-1) is stationary and in close proximity to the AP, its channel is good most of the time. However, for N_2 and N_3, as they move farther away, they will have to use lower data rates to ensure good performance, meaning different receive threshold will be used to determine whether a packet is received in error. If the wrong data rate is used and the packet is received with insufficient signal-to-noise ratio, the packet will have errors and be discarded. Depending on the severity of the channel variation, as determined by *std_db*, both of these flows will experience different packet error rates. In contrast, Flow-1 has sufficient margin to absorb these channel variations since it is close to the AP.

The degradation seen on Figures 5(a) and 5(b) is due to packet errors and low data rates. Flow-1 is unaffected by the misfortunes of N_2 and N_3, thus it is able maintain its throughput. However, as N_2 and N_3's data rate decreases, meaning more "air-time" is required, we see Flow-1's throughput begin to degrade. Similarly, in Figure 5(b) we see a gradual rise in delay. Therefore, Flow-1 is not perfectly isolated if there is no time fairness. However, from these figures it is clear that removing the MAC's default retransmission attempts yield significant performance gains.

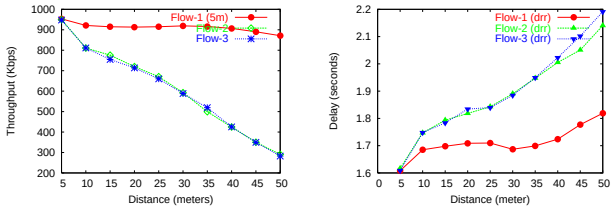
5.2 Throughput and Delay

We now move to experiments involving T^2-Fair. Figures 6(a) and 6(b) show the throughput and delay recorded for all three flows. From these figures, Flow-1's throughput is significantly higher, i.e., 40% higher compared to using only a simple round robin scheduler. In fact, the additional "air-time" occupied by flows 2 and 3 result in an increase of Flow-1's throughput since it is allocated more time as other flows' channel degrade and occupy the channel longer due to lower data rates. From these figures we also see that the throughput obtained by flows 2 and 3 is fair; more accurately, proportionally fair.



(a) Throughput (b) Delay

Figure 4: Simple Round-Robin: Throughput for all flows with MAC retransmission parameters set to the default value of 7 and 4 depending on whether RTS/CTS is used.



(a) Throughput (b) Delay

Figure 5: Simple Round-Robin: Throughput for all flows when MAC retransmission parameters are set to one.

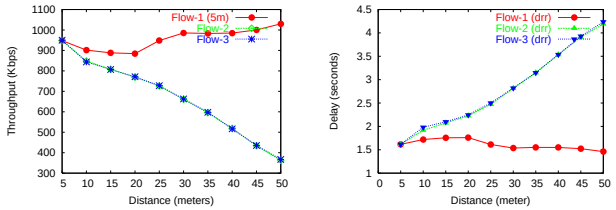
Observe that in Figure 6(a), Flow-1 observes a slight degradation in throughput. The reason is because around 5-20 meters, other flows have data rates that are similar in range to Flow-1. As a result, Flow-1 will often be grouped with either Flow-2, 3, or both. As a result, Flow-1's throughput drops since the available bandwidth is shared amongst two or three flows.

In a similar experiment, see Figures 7(a) and 7(b), but using 64 Kbps flows, T^2-Fair is able to sustain the throughput of all flows. After 40 meters, due to Flow 2 and 3's low data rates, their throughput begin to drop, however Flow-1 is unaffected. As errors become more prominent at larger distances, there is a difference between the throughput received by flows 2 and 3.

5.3 Effects of α on Device Grouping

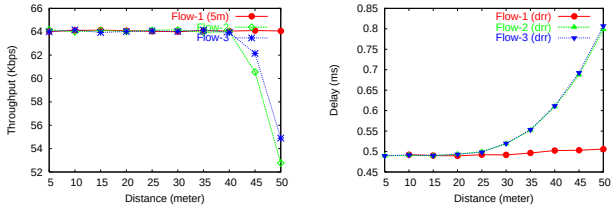
In this experiment we investigate the effect of α, as used in Equ. 1, on throughput. The value of α directly affects group membership and hence throughput since more devices will be sharing the capacity of a group. We position N_2 and N_3 20 meters from the AP. N_1 remains at 5 meters. We then record the throughput and delay of all devices. In addition to varying the value of α, we also experimented with *std_db* values of 2.0, 6.0 and 10.0.

Figures 8(a) and 8(b) show the throughput of N_1 and N_2. For all α values, no significant impact on throughput is ev-



(a) Throughput (b) Delay

Figure 6: 4 Mbps Flows: Achievable throughput and delay when using the T^2-Fair scheduler.

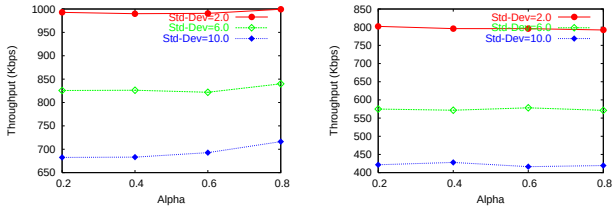


(a) Throughput (b) Delay

Figure 7: 64 kbps Flows: Achievable throughput and delay when using the T^2-Fair scheduler.

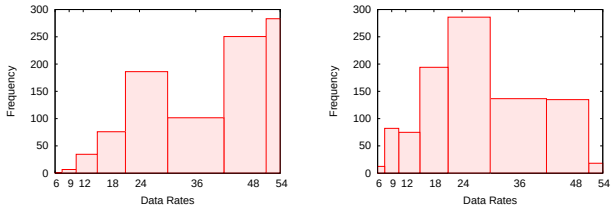
ident. However, from Figures 9(a)−10 we see that different α values have an impact on group membership, especially when std_db is high. For example, when $\alpha = 0.2$ a flow remains within a group more frequently, however at $\alpha = 0.8$ a flow is likely to switch between all groups or data rates.

Nevertheless, given the relative stable throughput obtained for all flows using different α values, we see that T^2-Fair is insensitive to group membership dynamics. This is because T^2-Fair allocates time fairly to all groups in each round. If the link quality of a device, e.g., N_2, that is far away from the AP improves, thereby taking shorter transmission time, all other flows will benefit. On the other hand, if N_2's link degrades, then N_1 will be allocated more "air-time" to compensate. Therefore, in both scenarios, N_1 is not affected by N_2 changing groups. Alternatively, if all flows' link quality degrade, then throughput will degrade, due to increased service time. However, because of channel variations, there will always be a mix of high and low rate flows, thus gains and losses in time will normalize and given the strong path-loss component, the system will converge to its long term throughput.



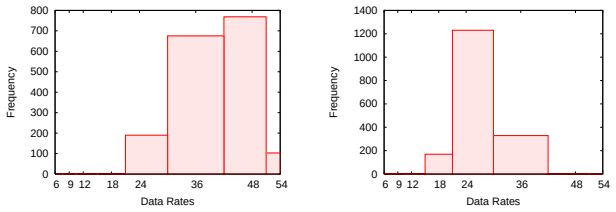
(a) Flow-1 (b) Flow-2

Figure 8: Effect of α on throughput.



(a) Flow-1 (b) Flow-2

Figure 9: $\alpha = 0.8, std\_db = 10.0$. The frequency in which Flows 1 and 2 is in a group.



(a) Flow-1 (b) Flow-2

Figure 10: $\alpha = 0.2, std\_db = 10.0$. The frequency in which Flows 1 and 2 is in a group.

6. RELATED WORK

Fair queueing (FQ) algorithms aim to isolate traffic and ensure all backlogged flows receive their share of the link capacity. In addition, these algorithms also provide delay and throughput guarantees. In the recent past, researchers have extended wired FQ algorithms to consider wireless errors; see [12] and its references. However, most of these works have not considered multiple data rates and are limited to compensating lagging flows due to wireless errors. Further, they model the wireless channel using a two-state Markov model, a simplistic model where the channel can either be in a good or bad state, thus does not consider the possibility of using different modulation schemes or data rates to compensate for errors. To address these shortcomings, researchers are beginning to develop new multi-rate aware schedulers.

Tan et al. [16] propose a scheduler that uses a token bucket to ensure time fairness. Here, tokens are time-units and bucket depth is used to control the burst of air-time given if no other flows can transmit. The scheduler chooses the queue with positive tokens in a round-robin manner. Each token is consumed by the MAC after a packet transmission, thus it includes overheads such as backoff. Moreover, failed transmissions also contribute to the consumption of tokens. A flow's fairness is ensured by adjusting its token rate.

Wang et al. [17] consider a channel with six data rates. They assume a physical layer technology that uses a feedback channel to monitor link quality. They have also considered proportional fairness where the service received by a flow is in accordance to its link quality. Their scheduler relies on the start-time FQ [5] (SFQ) algorithm as a reference system to track leading and lagging flows. The scheduler will pick a lagging flow that has the least amount of service before choosing one from the set of leading sessions. If the flow chosen is different to that selected by SFQ, the lag for the flow which should have been chosen is updated.

Yuan et al. [19] extended the work of [10] to provide *virtual* temporal fairness and omitted throughput based fairness and compensation. Their scheduler also uses the SFQ scheduler as a reference system to determine lagging and leading flows. A leading flow decides the amount of time it can relinquish based on its rate and maximum lead. This relinquished air-time is then distributed to lagging flows in proportion to their respective lag.

In [18], Wang et al. consider a multi-rate WLAN. A novel aspect is the control of flows' transmission rate according to $\frac{lag_i}{w_i}$, where lag_i and w_i denote the lag and weight for flow i. If a flow is transmitting too fast, then it can be slowed down. On the other hand, if a flow has a big lag or a high weight, then the flow is allowed to transmit using a wider range of data rates, thus allowing it to catch up and avoid delaying other flows. Also, their scheduler provides graceful degradation thus allowing lagging flows to catch up. Lastly, their scheduler separates flows into real-time and non real-time flows and keeps track of excess bandwidth. This bandwidth is then redistributed according to the weight of flows.

The ultimate goal of T^2-Fair and the aforementioned works is the same. That is, to provide a fair scheduler for multi-rate WLANs. However, our approach is different and has the key advantage of providing both time and throughput fairness. Arguably, it is simpler since it is built on the well-known DRR scheduler which has a work complexity of O(1). Further, it addresses the issue of long term fairness in multi-rate WLANs. In summary, T^2-Fair has the following advantages over existing works, (a) no reference system is used to track leading and lagging flows, (b) no computation of virtual time, and by extension, no sorting of packets according to their finish or start time, (c) considers both time and throughput fairness, and (d) works closely with the MAC and link-adaptation algorithm.

7. CONCLUSION

In this paper we have highlighted the key problems in multi-rate WLANs and motivated the two-tiered design of T^2-Fair. From our extensive simulation studies we found that T^2-Fair can indeed provide both time and throughput fairness. Moreover, it effectively isolates low and high rate flows; an important consideration in multi-rate WLANs.

The main limitation of T^2-Fair is the lack of consideration for uplink flows, which require the gathering of queue information from devices and informing them when and how long to transmit. Another future work is to investigate T^2-Fair's performance using a TDMA based MAC. This would allow strict control of transmission times and reduce overheads that would lead to unnecessary delays. We are also interested in efficient channel prediction algorithms, which have the potential to minimize wastage due to packet errors resulting from the mismatch between data rate and link quality.

8. ACKNOWLEDGMENT

Many thanks to Darryn Lowe and Ricardo Gandia-Sànchez for the lively and fruitful discussions that mature a lot of the ideas in this paper.

9. REFERENCES

[1] P. Bahl and V. Padmanabhan. RADAR: An in-building rf-based user location and tracking system. In *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, 2000.

[2] G. Berger-Sabbatel, F. Rousseau, M. Heusse, and A. Duda. Performance anomaly of 802.11b. In *Proceedings of IEEE INFOCOM'2003*, San Francisco, USA, 2003.

[3] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. Tripathi. Enhancing throughput over wireless lans using channel state dependent packet scheduling. In *IEEE Proceedings of INFOCOM'96*, San Francisco, USA, Mar. 1996.

[4] K.-W. Chin. T^2-Fair: A two-tiered time and throughput fair scheduler for multi-rate WLANs. Technical Report, SECTE-1-2006, University of Wollongong, Australia, 2006.

[5] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *IEEE Proceedings of INFOCOM'94*, Toronto, Canada, June 1994.

[6] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM MOBICOM'2001*, Rome, Italy, Oct. 2001.

[7] IEEE. Part 11a: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: High speed physical layer in the 5 GHz band, standard specification. IEEE Std 802.11a-1999, 1999.

[8] A. Kamerman and L. Monteban. WaveLAN II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, pages 118–133, 1997.

[9] R. Kohno, M. McLaughlin, and M. Welborn. DS-UWB physical layer submission to 802.15 task group 3a. IEEE 802.15 Working Group Document - 802.15-04-0137r4, 2005.

[10] S. Lu, V. Bharghavan, and R. Srikant. Fair scheduling in wireless packet networks. *IEEE/ACM Transactions on Networking*, 7(4), Aug. 1999.

[11] S. McCanne and S. Floyd. ns network simulator-2. http://www.isi.edu/nsname/ns/.

[12] T. Nandagopal, S. Lu, and V. Bharghavan. A unified architecture for the design and evaluation of wireless fair queueing algorithms. *ACM/Kluwer Wireless Networks*, 8(1):231–247, Feb. 2002.

[13] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice-Hall, 1996.

[14] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad-hoc networks. In *ACM MOBICOM*, Atlanta, Georgia, USA, Sept. 2002.

[15] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.

[16] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate wireless LANs. In *Proceedings of USENIX*, Boston, USA, June 2004.

[17] L. Wang, Y.-K. Kwok, W.-C. Lau, and V. K. Lau. Efficient packet scheduling using channel adaptive fair queueing in distributed mobile computing systems. *ACM/Baltzer Mobile Networks and Applications (MONET)*, pages 297–309, 2004.

[18] Y.-C. Wang, Y.-C. Tseng, and W.-T. Chen. MR-FQ: A fair scheduling algorithm for wireless networks with variable transmission rates. *Simulation: Transactions of The Society for Modeling and Simulation International*, 81(8), 2005.

[19] Y. Yuan, D. Gu, W. Arbaugh, and J. Zhang. Achieving fair scheduling over error-prone channels in multirate WLANs. In *IEEE VTC'2004*, 2004.