

A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks

Joseph Dunn, Michael Neufeld, Anmol Sheth, Dirk Grunwald and John Bennett
University of Colorado
Department of Computer Science
Boulder, CO 80309
{josephd,neufeldm,sheth,grunwald,jkb}@cs.colorado.edu

Abstract

Many companies, organizations and communities are providing wireless hotspots that provide networking access using 802.11b wireless networks. Since wireless networks are more sensitive to variations in bandwidth and environmental interference than wired networks, most networks support a number of transmission rates that have different error and bandwidth properties. Access points can communicate with multiple clients running at different rates, but this leads to unfair bandwidth allocation. If an access point communicates with a mix of clients using both 1mb/s and 11mb/s transmission rates, the faster clients are effectively throttled to 1mb/s as well. This happens because the 802.11 MAC protocol approximate “station fairness”, with each station given an equal chance to access the media. We provide a solution to provide “rate proportional fairness”, where the 11mb/s stations receive more bandwidth than the 1mb/s stations. Unlike previous solutions to this problem, our mechanism is easy to implement, works with common operating systems and requires no change to the MAC protocol or the stations.

1 Introduction

The 802.11 wireless network standard allows for the simultaneous usage of multiple data rates on a single wireless LAN (WLAN). This is called a multi-rate wireless network. Multi-rate is achieved by using a single, low bitrate encoding technique for the *header* portions of packets while allowing individual nodes to use higher bitrate encodings on the *payload* portions when noise conditions permit. This feature has greatly eased the deployment of 802.11 networks by allowing incremental introduction of newer and higher bitrate equipment, e.g. 802.11b at 11Mb/s and 802.11g at 54Mb/s, without having to immediately abandon existing, slower equipment.

More importantly, multi-rate physical layers make wire-

Table 1. Transmission range by bitrate in a semi-open environment

Transmission rate	Semi-open range
1 Mb/s	110m
2 Mb/s	90m
5.5 Mb/s	70m
11 Mb/s	50m

less networks more robust. The 802.11b standard offers four different transmission speeds – 1, 2, 5.5 and 11 Mb/s. The higher transmission rates use encodings which are faster to transmit, but are also more susceptible to error and hence require a better signal to noise ratio for successful reception. A multi-rate network provides graceful degradation as stations move away from an access point. This is often expressed in terms of distance in open space, *i.e.* an unobstructed, open field with minimal outside RF interference. Table 1 shows the specified ranges of communication at different transmission rates for an Orinoco Gold 802.11b card.

Unfortunately, there is a key property of the 802.11 MAC which greatly reduces the bandwidth available to higher rate nodes when operating in an environment alongside lower rate nodes. The 802.11 MAC attempts to enforce access fairness on *station* basis. In other words, each station has an equal probability of acquiring the wireless media for each transmission. This has the unfortunate side effect of allowing low bitrate nodes to monopolize the medium for a far greater percentage of the time than high bitrate nodes. Because faster nodes are able to transfer *individual* packets much more quickly than slow nodes they yield the medium much more often than slower nodes transmitting the same number of bytes per packet. The fast nodes must then wait a relatively long time for the slow nodes to finish transmitting, greatly reducing their achievable bandwidth. Our tests with bulk data transfers showed that when placed

alongside a 1Mb/s node the performance of an 11Mb/s node dropped down to that of the 1Mb/s node; this “performance anomaly” has been noted and analyzed by other researchers [3]. Worse yet, the encoding scheme used at 11Mb/s is less resistant to noise than at 1Mb/s, potentially increasing the rate of packet loss and retransmission; this means that the 11Mb/s station may actually receive *less* bandwidth than the slower connection.

Prior approaches to addressing this issue have proposed altering the 802.11 MAC protocol. Unfortunately, this solution doesn’t solve the problem for existing equipment. Altering the MAC layer of existing client hardware is very difficult or impossible. Even requiring all clients to download new drivers or software is an undesirable logistical and support problem. With this in mind we propose a *cross-layer* scheme that exploits IP path MTU discovery [8] to reduce the number of bytes per packet sent by nodes connected at lower bitrates while allowing higher bitrate nodes to send full size packets. This results in an effective control mechanism that can be used by a bandwidth allocation policy to implement a range of allocation policies, including a “rate proportional fairness” allocation.

In the remainder of this paper, we first describe the problem with the 802.11 MAC protocol in more depth and survey the prior solutions to the problem. Then in §3, we describe how the path MTU can be used to automatically control the bandwidth allocation for a number of clients. We describe three possible implementations of the mechanism. In §4, we conduct an experimental evaluation using our mechanism. We then conclude in §5 with a summary of the paper and possible future work.

2 Background

In this paper, we are primarily concerned with *bridging infrastructure networks*, or networks that connect access points through a *wired distribution network*. Clients, normally called stations, *associate* with an access point using an association and authentication protocol. Access points coordinate with each other to communicate the arrival and departure of stations at specific access points. Traffic from stations is bridged to the wired network and pruning spanning-tree protocols are used to direct the transmission of messages destined for specific stations through the access points to which they are associated. This is the most common deployment scenario for wireless networks, including popular “hotspot” networks and home wireless networks.

The 802.11 MAC layer uses two protocols to arbitrate media access. The Point Coordination Facility (PCF) uses a centralized arbiter for media access. This arbiter is typically an access point. Under PCF, the access point allocates time slots to each station. Stations are polled for messages

by the point coordinator. This is a “contention free” protocol since arbitration is handled by the point coordinator. Following a contention free period, the point coordinator uses another protocol called Distributed Coordination Facility (DCF) to allow unscheduled stations to communicate. While PCF provides many features, it is optional and typically not implemented by most access points; thus, we don’t discuss it any further.

The second arbitration mechanism is the Distributed Coordination Facility (DCF). We present a shortened introduction to the DCF protocol and only approximate the delays involved; interested readers should consult [3]. DCF uses a carrier-sense, multiple-access with collision avoidance MAC protocol (CSMA/CA). Unlike the more familiar CSMA/CD protocol used in wired ethernet, the CSMA/CA protocol attempts to *avoid* packet collision because wireless radios can not both transmit and receive at the same time; thus, collision detection is impossible. The DCF protocol relies on a series of timing intervals and exponential backoff mechanisms to arbitrate access. Figure 1 shows a timeline for two stations accessing the same media. The MAC protocol relies on specific delays between events. A *SIFS* (short inter-frame spacing) period is defined as a $10\mu s$ delay. A *DIFS* (distributed inter-frame spacing) is defined as a $50\mu s$ delay. The normal media access by station #1 is shown in Figure 1. The station waits for one DIFS period; if no carrier is detected, it waits for a random backoff period. If still no carrier is detected, the station transmits the Physical Layer Convergence Protocol (PLCP) preamble and header. The PLCP consists of a 144 bits preamble that is used for synchronization to determine radio. The preamble is 128 bits of synchronization, followed by a 16 bit pattern 1111001110100000. This sequence is used to mark the start of every frame. The next 48 bits contain the PLCP header, which contains four field: **signal**, **service**, **length** and **header error check**. The signal field indicates how fast the payload will be transmitted (1, 2, 5.5 or 11 Mbps). The service field is currently not used. The length field indicates the length of the ensuing payload, and the header check is a 16 bit CRC of the header. The PLCP is always transmitted at the *base rate*, typically 1 Mbps. Thus, 24 bytes of each packet are sent at 1 Mbps, no matter what rate is used for the payload. The PLCP introduces 24 bytes of overhead into each wireless Ethernet packet. This is followed by a 30 byte MAC header, the actual data payload and then CRC checksum. After the packet is received, the receiver waits one SIFS period and then transmits another PLCP preamble, header and 14-byte ACK as an acknowledgement that the packet was received correctly.

Contention is handled by an exponential backoff and a mechanism we will not describe called the *Network Activity Vector* (NAV), which is used to estimate when the media will be available for the next access. In Figure 1, station

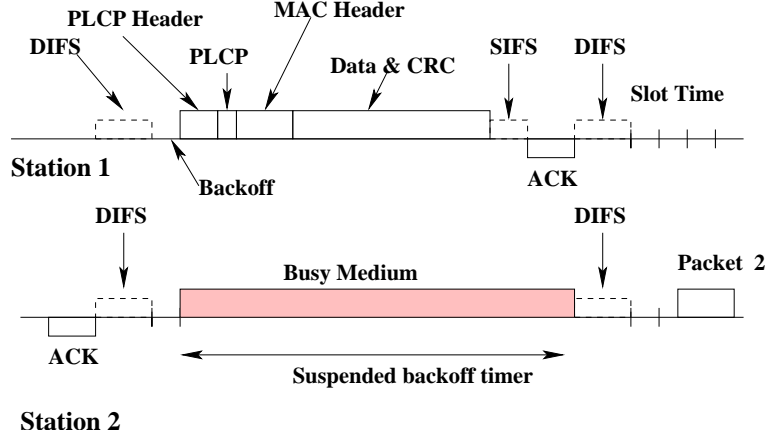


Figure 1. Schematic representation of the MAC CSMA/CA protocol

two detects that station one has acquired the media because it had a shortened backoff period, and station two backs off until the media is available.

This contention protocol provides long-term fair access to the media [2], although recent studies have shown that it is unfair in the short term [5]. When the media access is unfair in the short term, it impacts the performance of TCP because packets arrive in bursts rather than in a smooth stream regulated by the TCP ACK's. Most of the previous work on fairness has only examined single-rate wireless networks. Heusse *et al* [3] provide simulation and measurement studies showing that the 802.11 MAC protocol also provides station fairness when using multi-rate networks.

The PLCP is always sent using the base rate (1mb/s). In a multi-rate network, the MAC header, payload, CRC and the ACK are sent using the rate specified in the PLCP header. Figure 2 shows a series of measurements we made of an 802.11b network running at an 11mb/s rate. These measurements were made between a wireless station and another system behind an 802.11 access point; we explicitly controlled the transmission rate from the 802.11 access point. We then fit a linear regression model to those measurements. The model indicates that it's reasonable to estimate the time to transmit a packet at 11mb/s as $t_{p,11}(s) = 1.18 + s \times 0.000756$. Although the 802.11b network can support larger payloads, the maximum payload in a bridged network is 1500 bytes. Thus, at 11mb/s the time to transmit the largest possible message is 2.31ms. This results in a maximum bandwidth of $\approx 648KB/s$. If a 1mb/s transmission rate is used, the corresponding model is $t_{p,1}(s) = 1.49 + s \times 0.00736$, the time to transmit a 1500 byte packet is 12.5ms and the maximum bandwidth $\approx 120KB/s$. Our experimental results closely agree with theoretical models [3]. The intercepts for the two transmission rates don't match because the duration of the PLCP for the 1Mb/s rate is slightly longer than that of rates more than

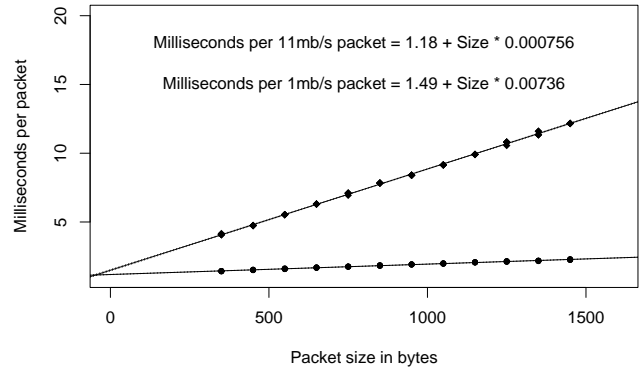


Figure 2. A linear regression model with 95% confidence intervals relating the minimum packet transmission time (in milliseconds) vs. packet size for an 802.11b wireless network at the 11mb/s rate. The coefficient of regression is $R = 0.99$.

1Mb/s ($192\mu s$ vs. $96\mu s$) and because of intrinsic measurement errors.

Mixed Rate Traffic: Assume that the 802.11 MAC provides long term fairness, and stations alternate taking turns acquiring the media. Consider a mixed rate network with one 1mb/s and one 11mb/s station with absolutely no contention. Each station will take a turn transmitting a message: the 1mb/s takes 12.5ms and the 11mb/s station takes 2.31ms, so a complete cycle is 14.8ms. During that 14.8ms, 3000 bytes are delivered, for an effective channel bandwidth of 202.7KB/s. In a perfect fair multi-rate network, each sta-

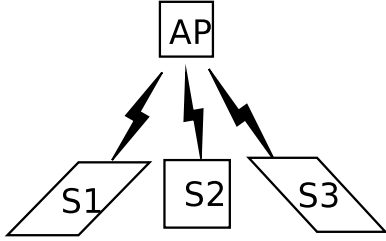


Figure 3. Single Access Point and Three Stations

tion receives half of this bandwidth, or 101 KB/s.

Thus, the faster rate station runs at the same rate as the station with the slower transmission rate. The 1 Mb/s station receives $\frac{101}{120} = 84\%$ of the bandwidth it would receive if it was the sole station using the network; however, the faster station would receive only $\frac{101}{648} = 16\%$ of the bandwidth it would receive if it was the sole station using the network. This is only an approximation based on measurements, but it agrees with the much more complicated analysis presented by Heusse [3].

The crux of the problem is that the 802.11 MAC protocol provides fair access to each station for each *packet*, independent of the time that each message transmission actually occupies. Such bandwidth disparities would also be seen in networks if the different stations transmitted messages at different sizes, but even in the presence of equal traffic flows, the different transmission rates greatly influence the delivered bandwidth.

There are a number of alternative fairness metrics that could be used instead of packet-level fairness. One alternative metric is *rate-proportional fairness*. In other words, if two stations have an achievable bandwidth of 648 KB/s and 120KB/s in isolation using different transmission rates, a rate-proportional bandwidth allocation would allocate $648/(648 + 120) = 84\%$ of the bandwidth to the 11mb/s stream. This is equivalent to giving each station an equal share of the media time – *i.e.* using a TDMA mechanism. For example, in the same period of time that a 1 mb/s station can transmit 1500 bytes (12.4ms), an 11mb/s station should be able to deliver ≈ 5 similarly sized packets.

There are a number of ways to enforce such a fairness metric, and those techniques can be applied at the PHY, MAC or network layer. In §3, we show how we can adjust the maximum transmission unit (MTU) in the IP network layer to control message size, largely accomplishing the goal of imposing a TDMA mechanism on the CSMA/CA MAC protocol. As we’ll show, this technique has the benefit of being backwards compatible with existing equipment, albeit at the expense of additional overhead and overall lower throughput.

Sender/Receiver Fairness: When used in a bridging infrastructure network, the 802.11 MAC layer introduces another form of unfairness [9] that is another special case that our general control mechanism can regulate. Figure 3 illustrates a situation where this problem can occur. Assume that station *S1* is constantly sending data – for the moment, assume it never receives any data. Likewise, assume that stations *S2* and *S3* only receive data. If all three nodes are transmitting at 11mb/s, we would expect that each receives $1/3$ of the total available bandwidth. In fact, *S1* receives $1/2$ the bandwidth and *S2* and *S3* each receive $1/4$ of the bandwidth. This happens because all data received by *S2* or *S3* must be transmitted by the access point, *AP*, and both *S1* and *AP* have “fair” access to the media. Thus, both *S1* and *AP* receive half the available bandwidth. Assuming that *AP* fairly relays packets to *S2* and *S3*, each of those stations will receive $1/4$ of the available bandwidth.

Again, the problem is that the different stations aren’t being given “equal time” on the media. However, there is a critical difference between this scenario and the previous one – it would be very difficult for a simple MAC-level mechanism to arbitrate Sender/Receiver fairness, because a MAC level mechanism would need to be aware of the number of stations, the transmission rate of the different stations and (ideally) the amount of demand placed by each station. Indeed, one previous solution [9] to the Sender/Receiver fairness problem adjusted the TCP window size to enforce fairness.

In the next section, we describe a simple mechanism that provides a very good approximation to a rate-proportional fairness policy for both basic fairness and sender-receiver fairness.

3 Controlling Fairness Using MTU

Our solution to both of these fairness problems is to adjust the message size used by the different stations to insure that the *time* used by each station is as close to equal to that of the other stations as can be achieved. We assume the time to send a packet of size *s* at a given transmission rate for station *k* can be modeled as $T_k(s) = o_{r_k} + tr_k * s$ where tr_k is the time-per-byte for a particular transmission rate and o_{r_k} is a constant overhead at the particular transmission rate for station *k*.

Since the access point can not change the rate used by each station (recall that this is set based on the signal to noise ratio in an attempt to reduce errors), one control available to the access point would be to adjust the message size sent by each station. In other words, if an access point has *n* stations associated with it, the AP should direct the stations to transmit messages such that $T_1(s_1) = T_2(s_2) = \dots = T_n(s_n)$.

Given this goal, we need to determine the maximum

message sizes for the individual stations and then force the stations to use those maximum message sizes. It should be noted that a smaller message size will reduce overall network efficiency due to the high fixed cost of sending a packet of any size. Therefore, we need to insure that the mechanism doesn't greatly reduce bandwidth when everyone is "being fair" (e.g. when several stations are all only receiving traffic and all at the same transmission rate). To this end when all clients are transmitting at the same data rate the access point allows them all to use the maximum MTU.

Although there is slight variance in our measured data, to a first approximation, we can assume that $o_1 = o_2 = o_{5.5} = o_{11}$ – in other words, we can assume the overheads for message transmission are the same. Likewise, without introducing major error, we can assume that the different time-per-byte times are proportional to the transmission rates – in other words, we can estimate that an 11 mb/s rate actually is actually 11 times faster than a 1mb/s rate. Our measurements from the previous section indicate this is a acceptable assumption, since the per-byte transmission rate for the 1Mb/s connection was measured as 0.00736 millisecond per byte and that of the 11Mb/s connection was 0.0000756 milliseconds per byte, or 9.7 times less. This implies that the access point can simply estimate the time needed by each station as $T'_k(s_k) = o + tr_k \times s_k \propto s_k/r_k$, where r_k is the transmission rate (i.e. 1, 2, 5.5 or 11).

At this point, we have made a circular argument – we need to know s_k to set s_k such that $T'_1(s_1) = T'_2(s_2) = \dots = T'_n(s_n)$. The solution we adopt is to assume that stations would always like to transmit the maximum sized packet; an alternate solution would be to assume that we could estimate future packet sizes. Using our simplified timing model, the access point simply finds the highest transmission rate, r_{max} , and then sets the maximum message size for each station to be $1500 \times \frac{r_k}{r_{max}}$. For example, we would like a 1mb/s station to send packets no larger than $\frac{1}{11} \times 1500 = 136$ bytes. From our previous measurements, this means a 1mb/s would take 2.49ms, while a 1500 byte packet at 11mb/s would take 2.31ms (the difference of $\approx 100\mu s$ arises from assuming the o_{r_k} are identical when they are, in fact, not equal).

As we show later, implementations of the IP protocol limit our ability to control the MTU enough to achieve the desired ratio exactly.

Simply adjusting the message size addresses multi-rate fairness, but more work is needed for the sender/receiver fairness. If an access point has queued messages for m unique stations, an additional m message times will be needed to deliver those messages (assuming that station level access is perfectly fair, and taking a long-term averaging view of packet deliveries). Thus, the maximum size

for station number k out of n total stations would be

$$1500 \times \frac{r_k}{r_{max}} \times \frac{n}{n+m}$$

if there are messages for m unique stations pending. The access point could refine this estimate further by actually estimating the time needed to transmit the pending messages to the stations rather than assuming they are of the maximum size. Moreover, the access point could use additional information to estimate if a receiving stations is likely to both send and receive a message in the same "turn" – this would reduce the $\frac{n}{n+m}$ term. We assumed that a station would either be transmitting or receiving in any given "turn", which results in a fair bandwidth allocation for the example in the prior section. We left these extensions to future work since our initial implementation performed so well.

There are practical problems with having the access point direct the stations to control their message size, the most important of which is that there is no mechanism specified in the 802.11 MAC protocol that would allow this. However, since we're assuming the access point and stations are using IP networking, the access point can vary the Maximum Transmission Unit (MTU) seen by each station. With an IPv4 network, each station will (or should) use Path MTU Discovery[8]; with an IPv6 network, stations *must* use Path MTU Discovery [7].

Since Path MTU Discovery is a little arcane, and we are applying it in a non-traditional manner, we briefly review its function. In an IPv4 network, different links may have a different maximum physical transmission units. An IPv4 router should fragment any incoming datagram that is larger than the MTU of the outgoing link. The fragments are delivered to the destination and reassembled. If any fragment is lost, the entire datagram is discarded. Since fragmentation is expensive for routers and causes increased packet loss, most systems implement *path MTU discovery*. In this mechanism, messages are sent with the "DF" (don't fragment) bit set in the IP header. If a router needs to fragment a message with a DF bit set, it drops the packet and returns an ICMP (internet control message protocol) datagram with "Unreachable error – fragmentation required" to the datagram source address. That ICMP datagram contains the MTU of the outgoing link on the router that would have otherwise fragmented the datagram and identifying information for the dropped datagram. When the source receives the ICMP error message, it reduces the MTU for outgoing datagrams on that connection. For UDP datagrams, the source would adjust the outgoing MTU and fragment future messages at the source. For TCP datagrams, the TCP stack adjusts the segment size to insure that datagrams aren't fragmented. In 1994, few operating systems implemented path MTU discovery; Stevens [10] indicates that only Solaris 2.x had the feature at that time. Today, most operating systems implement path MTU discovery.

RFC 1191 [8] recommends that a source should repeat path MTU discovery every 10 minutes, in case a new route was selected; Stevens [10] (p156) found that Solaris 2.x tries to re-estimate the path MTU every 30 seconds by sending another packet with the “DF” bit set. In IPv6, intermediate routers do not fragment packets, and every source is required to use path MTU discovery.

Normally, a router would announce the same MTU to each source accessing a link. We are changing that assumption by using the MTU to control the transmission rate of individual stations on a wireless network. Our scheme relies on giving the impression to different sets of nodes on the same physical wireless link that they have different maximum packet sizes.

In order to control the MTU on a host-specific level we constructed a pair of Perl scripts that control routing. The routing script enables IP level forwarding in the Linux kernel, and constructs routing tables with explicit MTUs. The `dynamicRoutes` script combines information from the Linux `/proc` file system with the `ip route` and `ip rule` commands to dynamically alter the MTU for specific hosts. Currently the scripts total 381 lines of Perl.

Based on a list of MTUs the routing script constructs one routing table for each MTU using the Linux policy based routing framework. These tables each contain two rules with equal, explicitly specified MTUs. The first rule moves data from the wireless network interface to the wired network. The second rule moves data from the wired network interface to the wireless network. In this way connections made to or from a wireless host are forced through a route with the specified MTU. When packets larger than the specified MTU move along these routes the Linux IP stack enforces the MTU by either issuing the appropriate ICMP message or fragmenting the packet.

The `dynamicRouting` script constructs appropriate routing rules after collecting data from both the `/proc` file system and the `ip rule list` command. Through the `/proc` file system the HostAP driver provides statistics about each 802.11 association. From this information we use the transmission rate of the most recently received packet to approximate the overall transmission speed for that association. Alternatively, we could also use the transmission rate that the access point is using to communicate packets back to the wireless. However, since the access point could have a more powerful signal, the access point’s transmission rate could be significantly higher than the host’s transmission rate.

Our current implementation is written specifically to take advantage of features present in the HostAP driver; specifically, information about the transmission rate for a specific client, as well as the mapping between MAC addresses and IP addresses. However, this information, can also be retrieved from some access points via SNMP; specifically,

Cisco access points provide this information.

Since the HostAP driver’s associations are recorded via the MAC addresses of the associated cards we use the system ARP cache to create a mapping from IP address to transmission rate. Each transmission rate is then mapped to a specific routing table created earlier. The current routing rules are then examined and any stale host-specific rules are removed. Finally, the `dynamicRouting` script uses the `ip rule add` command to insert host-specific routing rules that force traffic from the host through a routing table with an explicit MTU for all packets using that routing table.

The dynamic routing script reads all the current association information and recomputes the routing rules only once per second, thus incurring minimal overhead. In our current implementation, we do not have a way of estimating m , the number of unique stations scheduled to receive messages, and we adjust this manually in our experiments.

In a little more detail, an actual routing table for short packets would be:

```
%ip route list table short
10.1.2.0/24 dev wlan0 scope link mtu 350
default dev eth0 scope link mtu 350
```

The rules involved in moving traffic to and from a specific machine through the “short” table are:

```
% ip rule list
0: from all lookup local
1000: from 10.1.2.2 iif wlan0 lookup short
1000: from all to 10.1.2.2 iif eth0 lookup short
32766: from all lookup main
32767: from all lookup 253
```

There are complications with this implementation. In particular, it is not possible to enforce a 11-fold change in message sizes. For example, if an 11Mb/s connection is to be allowed a 1500 byte packet, the 1Mb/s connection *should* be forced to use an MTU of 136 bytes. However, we were unable to lower the MTU before ≈ 350 bytes without complication.

This problem is worse for networks with more transmission rates. For example the 802.11g protocols provides transmission rates from 6 to 54 Mb/s. The differences between the 6Mb/s and 54Mb/s rates are similar to those for the 802.11b network we measured in Figure 2; the 54 Mb/s data rate can be approximated as $t_{p,11}(s) = 0.227 + s \times 0.000158$, and the 6 Mb/s rate is approximately 9 times that of the 54Mb/s rate. The 802.11g protocol uses a different timing model to control media access, which is why the time to transmit any packet drops from 1.18ms to 0.227ms. If an 802.11g access point is servicing only 802.11g clients, our ability to control the MTU should provide approximately the same factor-of-four in controlling transmission time.

A “mixed network” of 802.11b and 802.11g stations would appear to be more difficult to control, since there would appear to be a 54-fold difference in transmit speeds. However, when an 802.11g access point services both 802.11b and 802.11g stations, a “preamble packet” is sent that is understandable by the 802.11b stations as a “protection mechanism” [11]. Additionally, the 802.11g MAC layer is designed to allow 802.11g stations to have twice the number of transmit opportunities as 802.11b stations. This extra overhead and MAC changes result in an effective maximum bandwidth of $\approx 12\text{Mb/s}$ for mix networks. Thus, the needed control is actually comparable to that needed by the 802.11b network. Again, since our implementation can only effect a four-fold change in the MTU (from 1500 bytes to 350 bytes), more precise control is not possible.

4 Evaluation

For our tests, we used three 2.4GHz Pentium 4 systems running Redhat Fedora Linux with the 2.4.22 kernel. Each system had an 802.11b wireless card for connectivity and used the HostAP [4] driver. We configured one system to act as an access point, and the other two as clients. We used our user-space daemon implementation for all tests.

4.1 The Cost of Sharing

As expected, when a node operating at 11Mb/s shares a link with a node operating at 1Mb/s it pays a heavy price in throughput. Figure 4 shows the times required to transfer a 10MB file using `scp` over a wireless link for nodes operating at 11Mb/s and 1Mb/s individually (“alone”) as well as when they are sharing the medium with one other node. The 11Mb/s station can transmit the file ≈ 4.6 times faster than the 1Mb/s station, slightly less than the rate predicted by our regression model when comparing the time to transmit 1500 byte packets; this is expected since an actual file transfer involves a combination of short and long packets as well as TCP-level delays. When a station shares the transmission link (*i.e.* either the *homogeneous* or *heterogeneous* measurements), the other station is transmitting at the *maximum rate*.

When sharing the link with a node of the same speed (“homogeneous”), each node takes approximately twice as long to perform the transfer. This confirms the expected result that the MAC protocol provides station fairness – each station is receiving about half the available bandwidth.

However, when sharing the link with a node of different speed (“heterogenous”) the results are quite different. In this experiment, both the 11Mb/s and 1Mb/s stations would finish in approximately the same time (148 second *vs.* 132 seconds)¹. The 11Mb/s node drops to a performance level

¹Recall that the measurements are conducted when the companion sta-

tion is transmitting at the maximum rate. Thus, the time values are not additive, and would indicate the time when a file transfer is completed given the maximum use of the media.

which is not just substantially worse than when it shares with another 11Mb/s station, but is *worse than the 1Mb/s node in that scenario*. As discussed earlier, we believe this occurs because the MAC layer in enforcing station fairness (which should result in a 50/50 split of bandwidth) and the fact that the 11Mb/s rate is more prone to errors.

From Figure 4 one can see that in the heterogeneous case there is significantly more variance in the performance of the 11Mb/s stream. We believe the reason that both the variance and the fact that rate of the 11mb/s station is less than that of the 1mb/s station is because the 11mb/s stream is more susceptible to errors from environmental noise. The 1Mb/s node does well when sharing with an 11Mb/s node, suffering only a 20% increase in its transfer time compared with when it has uncontested access to the medium, and slightly less than 60% of the time it would take if it had to share with another 1Mb/s node – this agrees well with our estimate that the 1Mb/s station would receive 84% of the bandwidth it would normally receive.

4.2 Enabling Our Control Mechanism

Figure 5 shows the results of running our adaptive MTU control mechanism. We contrast the transfer time of the non-adaptive mechanism against that of the adaptive mechanism to show that our mechanism adds little overhead – there is little difference between the two samples when run in a “homogeneous” organization (*i.e.* two concurrent 1 mb/s connections or two concurrent 11mb/s connections). This is as expected and desired – it indicates that rate-proportional fairness degenerates to station fairness when the same modulation rate is used.

However, in the “heterogeneous” case the adaptive scheme has a substantial effect. The copy running at 1 mb/s takes about 1/2 the time of the copy running at 1 mb/s when the adaptive mechanism is used, resulting in a bandwidth twice that of the 1mb/s connection. In this experiment, the MTU for the 1mb/s connection was set to 350 bytes, because smaller MTUs caused application problems.

Although our results should be independent of the specific IP implementation, we felt it prudent to run similar tests with a mix of Windows and Linux clients. The results are indistinguishable from the earlier tests when only Linux clients were used, and therefore are not presented here. This shows that our mechanism works well with commonly available platforms and is practical to use in commercial wireless networks.

No.	Time	Source	Destination	Protocol	Info
1837	5.575833	10.1.3.10	10.1.2.2	TCP	ssh > 1103 [ACK] Seq=2793456834 Ack=846766940 Win=63712 Len=0
1838	5.579232	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=1448
1839	5.583798	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=1448
1840	5.583874	10.1.3.10	10.1.2.2	TCP	ssh > 1103 [ACK] Seq=2793456834 Ack=846769836 Win=63712 Len=0
1841	5.739402	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
1842	5.740666	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=575
1843	5.749777	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
1844	5.749833	10.1.3.10	10.1.2.2	TCP	ssh > 1103 [ACK] Seq=2793456834 Ack=846772157 Win=63712 Len=0
1845	5.753407	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=575
1846	5.756073	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
1847	5.759132	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=575

(a) tcpdump sequence showing the MTU being adjusted from 1500 to 1000 bytes

2006	6.076332	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
2007	6.078508	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
2008	6.078550	10.1.3.10	10.1.2.2	TCP	ssh > 1103 [ACK] Seq=2793456882 Ack=846848751 Win=63712 Len=0
2009	6.081311	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
2010	6.089520	10.1.2.2	10.1.3.10	SSH	Encrypted request packet len=873
2011	6.089608	10.1.3.10	10.1.2.2	TCP	ssh > 1103 [ACK] Seq=2793456882 Ack=846850497 Win=63712 Len=0

(b) tcpdump sequence showing the datagrams eventually stabilize at 873 bytes

Figure 6. Example of MTU automatic adjustment

4.3 Sender/Receiver Fairness

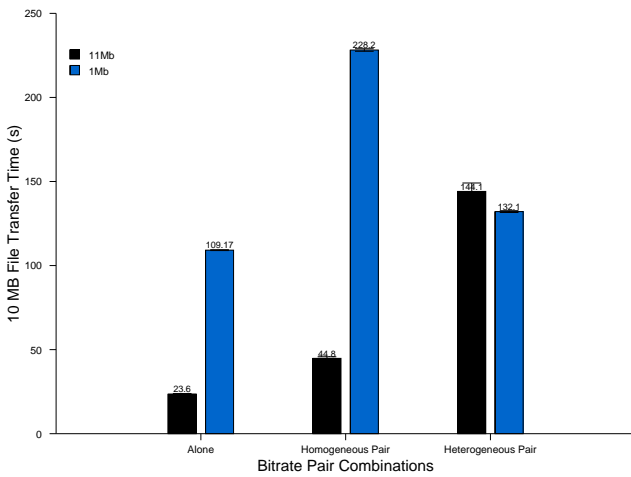


Figure 4. Transfer time required for a 10MB file using “scp” for stations operating at 1 Mb/s and 11 Mb/s with the default MTU. “Alone” shows the time required for a single connection not sharing the link. The “homogeneous” combination shows the time required when there are two stations at the same bitrate (1 Mb/s and 1 Mb/s or 11 Mb/s and 11 Mb/s) sharing the link, and the “heterogeneous” when there is a single other station operating at the *other* bitrate (1 Mb/s and 11 Mb/s) sharing the link. Smaller is faster.

Figure 6(a) shows a `tcpdump` trace from the server to which the file is being copied by the wireless station. When the MTU is changed by the access point, at frame 1841, the station sends a series of TCP segments at 873 and 575 bytes; this would appear to be a 1448 byte segment being split into two datagrams. Eventually, the upper layer protocol appears to learn the actual MTU size and consistently sends frames of 873 bytes, as shown in Figure 6(b). When we increased the MTU for the sender at the access point, we did not actually see the sender increase the TCP segment size. However, we did not run the experiment for the 10 minutes that is the period recommended by RFC 1191 for re-probing the MTU. Moreover, studies of wireless network usage shows that most connections are short-lived TCP connections commonly associated with HTTP traffic [1, 6]. Balachandran [1] showed that more than 90% of wireless traffic was TCP traffic, and about 1/2 of the traffic was HTTP, which consists of multiple short-lived TCP connections. In any case, the inability to increase the MTU of an existing connection implies that it may make sense to be conservative about reducing the MTU, particularly in environments where the bitrate may change frequently during a single connection.

Figure 7 shows our results for a statically-configured sender-receiver fairness experiment. This experiment corresponds to the description in §2, where a single sender is continuously sending a stream of data to a server *via* the access point and two receivers are continuously receiving data from data from a source *via* the access point. In this case, we statically set the MTU to 750 bytes for all stations except the access point when the adaptive mechanism was being used. We found that controlling the MTU of the sender could be used to flexibly control the fraction of bandwidth

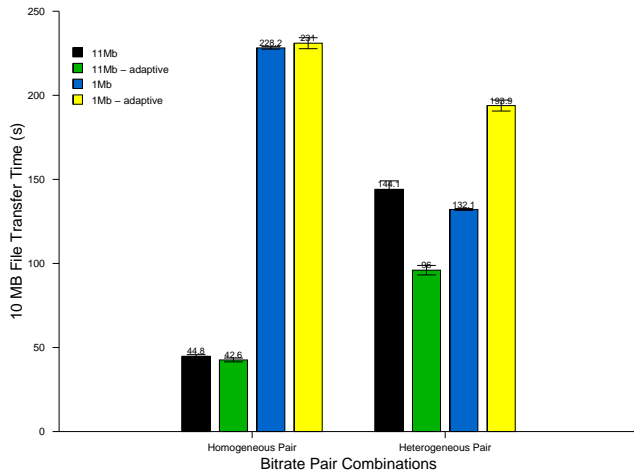


Figure 5. Transfer time required for a 10MB file using “scp” for stations operating at 1 Mb/s and 11 Mb/s with the default MTU compared with our adaptive mechanism operating. The “homogeneous” combination shows the time required when there are two stations at the same bitrate (1 Mb/s and 1 Mb/s or 11 Mb/s and 11 Mb/s) sharing the link, and the “heterogeneous” when there is a single other station operating at the *other* bitrate (1 Mb/s and 11 Mb/s) sharing the link. Smaller is faster.

devoted to the sender. In this experiment, as in the previous ones, the overall bandwidth decreases. This occurs because the larger number of packets induced by the reduced MTU causes more contention.

5 Conclusions and Future Work

Our results show that setting the MTU on a per-station basis is an effective control mechanism, and can be used to address the fairness issues encountered in practical wireless network deployments. To our knowledge, this is one of the few fairness mechanisms for wireless networks that has actually been implemented, because it relies on simple modifications at an access point rather than a change to the MAC layer.

For a bridging infrastructure network, it is possible to implement this mechanism at a central router. In most wireless networks, multiple access points bridge traffic from the wireless media to the wired media. A single system attached to the wired network is used to route traffic to the internet. Our mechanism simply knowing the transmission rate of individual stations associated with an access point.

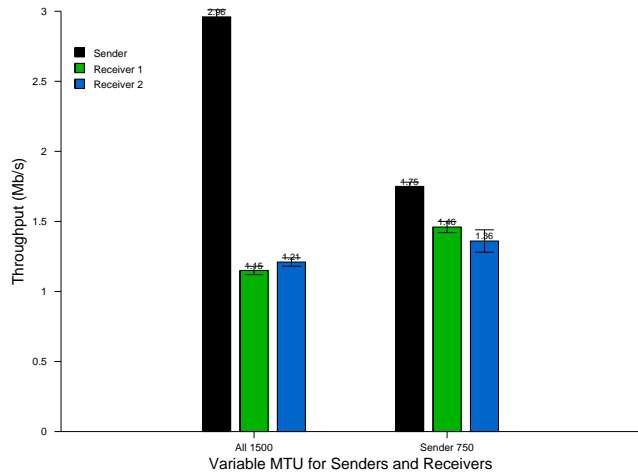


Figure 7. Bandwidth control in sender-receiver test. A single transmitting AP client receives an unfair share of the medium when sharing the AP with two other receivers. Adjusting its MTU to a lower length reduces its bandwidth and increases that of its peers.

Currently, commercial access points provide information about the MAC address of associated stations and the signal quality measured for those stations *via* SNMP. It would be a small change to have the access point also report the transmission rate *via* SNMP. Since the access points are bridges, the intermediate router would have access to the MAC addresses of the individual stations. All other information for the control algorithm (*e.g.* number of unique stations at a specific access point) can be determined by the intermediate router. Most existing wireless networks have a complex intermediate router that also enforces access control policies; the additional logic needed to implement our fairness mechanism is a minor addition.

The implementation described in this paper has numerous limitations, but it illustrates the utility of cross-layer control for emphasizing fairness. We are currently re-implementing the technique using our wireless router infrastructure, which combines the Click modular router and the HostAP wireless access point. In our more advanced implementation, we are also experimenting with mechanisms to actually measure delivered bandwidth to the different clients and adjust the message sizes based on that rather than the estimated ratios we discussed. Likewise, that implementation can use information about the number of outstanding packets to be delivered to individual stations. We believe this should provide a robust and very effective fairness control mechanism for wireless access points.

References

- [1] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of ACM SIGMETRICS*, 2002.
- [2] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, 1989.
- [3] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, USA, March-April 2003.
- [4] HostAP. Hostap. <http://hostap.epitest.fi/>.
- [5] C. E. Koksal, H. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless media access protocols (poster). In *Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 118–119, 2000.
- [6] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 107–118. ACM Press, 2002.
- [7] J. McCann, S. Deering, and J. Mogul. Path mtu discovery for ip version 6. IETF RFC 1981, Aug 1996.
- [8] J. C. Mogul and S. E. Deering. Path mtu discovery. IETF RFC 1191, Nov 1990.
- [9] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, , and P. Sinha. Understanding tcp fairness over wireless lan. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, USA, April 2003.
- [10] W. R. Stevens. *TCP illustrated : volume I*. Addison-Wesley publishing co., 1994.
- [11] M. Wentink, T. Godfrey, and J. Zyren. Overcoming 802.11g's interoperability hurdles. <http://www.commsdesign.com/showArticle/?articleID=16501220>, May 2003.