Deploying Ubiquitous Connectivity with Commodity Wireless Routers

Charles Reis Karthik Gopalratnam

{creis, karthikg}@cs.washington.edu

Abstract

Commodity wireless routers have become widely deployed, providing a new opportunity to provide inexpensive Internet connectivity in broad areas. We present a software architecture to promote sharing connectivity, using easily upgraded wireless routers and unmodified client devices. We focus primarily on aspects of security and resource allocation, addressing several issues specific to our domain, including appropriate authentication and router queuing mechanisms. We evaluate our queuing mechanisms on an experimental wireless testbed to show improved fairness and utilization compared to alternatives. Overall, our design is appropriate for both small and large deployments of upgraded wireless routers, supporting widespread growth of wireless connectivity.

1 Introduction

As mobile computing devices have grown in popularity, and as we have become more dependent on network services in general, there has been a similar rise in demand for network connectivity regardless of one's location. Many approaches have already been pursued to provide more ubiquitous connectivity, from wireless "hot spots" in public locations, to network access over cell phone networks, to city-wide deployments of 802.11 networks (or similar proposals, such as WiMAX [33]).

A new solution for providing inexpensive, high bandwidth connectivity is now possible, due to the recent proliferation of 802.11 wireless access points (*APs*, also known as wireless routers) in houses and apartments. These low-cost devices have been widely used for building home networks, creating many areas with physical connectivity in and around buildings in urban areas. While such networks are often unintentionally left open for anyone to connect, a new scheme for intentionally sharing network access is very promising for achieving ubiquitous Internet connectivity. Wireless routers are already widely deployed, and they offer higher bandwidth than cell phone networks. Additionally, when compared to new long-range, high bandwidth WiMAX antennas, 802.11 access points offer less contention for network resources due to their smaller coverage area.

Most significantly, some of the most popular and inexpensive wireless routers are highly programmable, allowing owners to easily install new Linux distributions on them to provide new functionality. This presents an opportunity to use software upgrades to routers to solve many of the difficulties for sharing connectivity.

1.1 Challenges

While wide-spread sharing of wireless connectivity is a promising opportunity, several technical and non-technical challenges must be addressed to make such an idea practical. From a technical perspective, sharing an Internet connection and wireless air time with more clients creates greater competition for scarce resources, indicating a need for improved resource allocation at the wireless router. This is exacerbated by the ability of wireless clients to send traffic at different bit rates. A client may choose to send at lower rates to reduce its number of dropped packets [29], but this adversely impacts the throughput of all clients, as illustrated in Figure 1.

The presence of potentially untrusted clients on a home network presents many security challenges, as well. While careful use of firewalls and encryption can protect local resources, many standard techniques common in 802.11 networks (such as the shared keys in WEP and WPA-PSK [10, 11]) are not sufficient for this setting. Additionally, each client must be securely identified by the router in order to correctly apply any resource sharing and access control policies.

From a less technical perspective, appropriate incentives are necessary to avoid a "free rider" problem, where few people share their connections and many take advantage of them. Configuration requirements must also be minimal, since the owners of wireless routers may have low technical expertise. Similarly, maintaining backwards compatibility and minimizing the number of devices that need to be upgraded will greatly increase the impact of any solution.

Other aspects of improving ubiquitous connectivity could clearly be addressed as well, including support for seamless mobility between access points or automatic coordination between nearby wireless routers. However, we

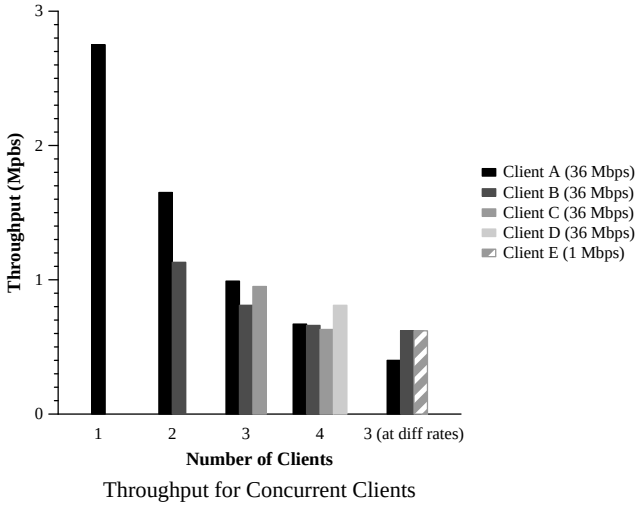


Figure 1: Measured throughput for 802.11g clients to hosts behind a 3 Mbps bottleneck, possible for a cable modem. Throughput drops off rapidly with the number of clients, and even more so when some clients send at lower bit rates.

choose to focus on the challenges above to first provide a feasible incentive to share connections, after which these enhancements could be considered.

1.2 Approach

We present an architecture for sharing Internet connectivity on small and large scales, using software modifications to commodity wireless routers. We discuss appropriate incentive models for such a system, and we describe the authentication and encryption components that are necessary for security. Finally, we propose and evaluate a resource sharing mechanism that allows different classes of wireless clients to effectively share the scarce resources of a wireless router, even when clients send data at different bit rates. Our mechanism is implemented as a Linux module for wireless routers and relies on TCP backoff at the end hosts, requiring no software changes for clients. When compared to existing queuing mechanisms, it shows potential to improve both fairness and utilization, though some implementation issues remain to be resolved. Overall, it represents a feasible approach to exploit existing wireless routers to create a great number of "hot spots" of Internet connectivity.

2 Related Work

Several aspects of our architecture have been considered in other contexts with different requirements, while some other work is complementary for our goals. In this section, we discuss similar uses of wireless networks, techniques for wireless resource allocation, and communities writing new software for wireless routers.

2.1 Sharing Wireless Connectivity

The Protocol for Authorization and Negotiation of Services (*PANS*) and the CHOICE network [2] provide an architecture for building public wireless networks. PANS provides "authorization, access, privacy, security, policy enforcement, quality of service (QoS) and accounting" for such networks, using a number of software mechanisms on clients, routers, and servers. PANS was designed for commercial deployment, and it requires infrastructure that is potentially infeasible in a home setting. Additionally, the need for clients to install software limits the scope of the system to clients with supported operating systems.

There are also a significant number of communities organized within cities to provide or share wireless network connectivity, using current technologies. These include NoCatNet [19] and Bay Area Research Wireless Network [4] in California, Seattle Wireless [26], NYCwireless [20], and many others. These groups often maintain a knowledge base of wireless technologies, and they help members set up and connect to public networks. Our work is complementary to theirs, and we provide an additional opportunity for greater financial incentives for participation.

2.2 Wireless Resource Allocation

A great deal of previous research has addressed quality of service and resource allocation issues for wired networks, and we focus specifically on Weighted Fair Queuing (*WFQ*) [8] to provide isolation between classes of flows with different priorities. Much work has tried to adapt fair queuing to a wireless setting, both for packet cellular networks [16, 18] and 802.11 networks [31, 3]. Of the latter, most work has focused on extensions or modifications to the 802.11 MAC protocol to support improved fairness. Our work attempts to improve fairness and utilization without modifying the MAC protocol or client software, using TCP flow control to adjust sending rates in the common case.

Other work has considered the notion of "time-based fairness" to improve utilization of wireless networks in the presence of clients using different bit rates. Tan and Guttag [29] describe an algorithm to regulate packets at an AP with similar goals as our approach. Their implementation requires driver changes on the AP, which is not feasible for routers with binary-only drivers (such as the Linksys WRT54G), while our implementation operates above the driver in the Linux kernel. Tan and Guttag

have also considered MAC modifications to provide long-term time-share guarantees [30].

Other tools have been proposed to address quality of service in wireless contexts, including Frottle [9] and WiCCP [22] for eliminating the "hidden node" problem. As above, these tools usually require special software at clients (e.g., Frottle requires Linux clients with modified firewall settings).

2.3 Router Development Communities

Finally, it is worth noting the considerable communities that have developed around current programmable wireless routers. The Linksys WRT54G [14] and similar routers (such as the Dell TrueMobile 2300) have general purpose processors that run the Linux operating system, and several community-based efforts have provided software distributions to enhance the functionality of these routers. These efforts include freely available distributions with new routing and administrative features [7, 12, 13], frameworks for easily building new router distributions [21], and even companies such as Sveasoft [28] with subscription-based models for enhanced distributions. The communities mentioned in Section 2.1 are often involved in this work, especially Seattle Wireless. The numerous projects based on modifying the WRT54G router indicate widespread interest in adapting such devices for new uses.

3 Architecture

To support greater ubiquitous Internet connectivity, we propose modifying commodity wireless routers to improve their effectiveness for safely sharing network access. Our changes are limited to the routers themselves, allowing unmodified clients to leverage the new features. On the router, we address security and authentication concerns in Section 4 and resource sharing mechanisms in Section 5.

However, we must first consider the incentives a router owner might have for sharing connectivity, to ensure our architecture is feasible in both the short term and the long term. While some number of owners may be willing to share their connectivity out of altruism, a "free rider" problem is likely to develop, requiring few owners to support large numbers of clients.

3.1 Local Deployment

In the short term, few modified routers will be deployed in any area, so the full benefits of ubiquitous connectivity will not be available. However, many owners of wireless routers already choose to share Internet access with neighbors, offering an appealing monetary incentive to split the bill among many clients. Note that these neighbors may be partially or wholly untrusted in terms of accessing local network resources or eavesdropping on traffic, and their devices may be susceptible to virus or worm attacks which could threaten the local network. Similarly, the router owner would want to ensure no one client could monopolize the connection. These issues illustrate the need for improved security and resource sharing mechanisms even for sharing on a small scale.

To support these goals for non-expert users, it should be straightforward to add usernames and passwords for new clients, as well as to define classes of clients for applying security and resource allocation policies. With appropriate authentication mechanisms in place, this can be accomplished with a web-based interface on the wireless router.

3.2 Global Deployment

As greater numbers of routers are adapted for sharing connectivity, there will be more opportunities for clients to connect to wireless routers in new locations, which would be problematic if each client must be added manually to each router's configuration. At the same time, Internet Service Providers (*ISPs*) will likely become interested in any trend of sharing connectivity, perhaps for contractual reasons regarding their service fees or the opportunity to gain new customers.

These two observations can be combined to envision a global deployment of "home-based hot spots." ISPs could charge customers a small fee to connect to the Internet through any upgraded wireless router, and they could provide service discounts to customers who share their connections, to encourage widespread coverage. This model is not without precedent: SpeakEasy already offers a "WiFi NetShare Service" [27] for customers to share connectivity with their neighbors at a reduced cost. Similarly, commercial hot spot services such as T-Mobile [34] and Boingo [5] indicate the willingness of consumers to pay for Internet connectivity on the road. By combining these models, ISPs can encourage home consumers to convert their wireless routers into hot spots for the ISP.

In this scenario, clients would securely authenticate to unknown wireless routers via their ISP, which would provide credentials to the router to allow the client to connect. Thus, the authentication service could be centralized, while the base of upgraded wireless routers could continue to grow in a decentralized manner.

4 Authentication

In this section, we discuss the requirements and design of an authentication system to support the architecture described in Section 3.

4.1 Requirements

We first note the requirements for any authentication mechanism to provide ubiquitous wireless connectivity, to meet the goals of access control and policy application.

First, any mechanism must be able to associate all inbound and outbound flows with particular users, each of whom may belong to a particular resource class. Note that any one user may have multiple devices and multiple active flows, but that policies should be applied based on a user's class, regardless of device. Thus, user-based authentication, as opposed to identifying devices, is preferred.

Second, the authentication mechanisms must be able to scale to a global service. Supporting ubiquitous connectivity involves allowing users to connect to unfamiliar APs, and each AP cannot be pre-configured with all valid users. Instead, APs must communicate with a global authentication service, such as the ISP model discussed in Section 3.2, to determine the privileges of unknown users.

Third, policy decisions must be effectively applied to every packet in every flow, to ensure the available bandwidth is partitioned fairly and appropriately, and to ensure security policies are not breached. This rules out potential mechanisms which rely on periodic checks for identification.

Fourth, any authentication technique must require minimal changes to client software, to ensure that the system can be immediately deployed with low impact for unsophisticated users. This implies a strong dependence on existing mechanisms for achieving our goals.

4.2 Potential Approaches

Several approaches are possible for authentication in wireless networks, but the above requirements restrict which solutions are appropriate.

MAC address registration is currently widely used for deployed wireless networks in institutions such as universities, as it provides per-packet user identification and recourse in the event of an attack. However, such an approach is not robust against MAC address spoofing, which can be easily performed by adversaries.

Other per-packet identification techniques are appealing, but usually require low level software changes on client devices. For example, PANS [2] uses encrypted tokens in a new packet header to authenticate each packet, involving software updates to clients.

Instead, we can consider certain existing encryption and authentication technologies to avoid changing client software. Using per-user encryption keys, we can securely associate flows with a user, and thus with a resource class. However, current common practice employs Wired Equivalent Privacy (*WEP*) [10], which is insufficient for two reasons. First, it uses a shared secret for encryption, making it difficult to securely differentiate traffic from different clients. Second, WEP has been shown to use encryption in critically flawed and insecure ways [6].

WiFi Protected Access (*WPA*) [11] offers two approaches to improve on WEP. WPA-PSK ("pre-shared key") continues to use shared secrets, but it improves on the flaws of WEP using TKIP [32] for encryption with longer and time-varying keys, with a planned transition to AES encryption. WPA-Enterprise builds on this to eliminate shared secrets, using 802.1x technology [1] and a RADIUS server [24] for user-based authentication and per-user keys. As the name implies, this approach is targeted towards enterprises rather than homes, as configuring an authentication server may be daunting for home users.

It is worth noting that all modern wireless clients support WPA-Enterprise, as of Windows XP Service Pack 2 and Mac OS X 10.3. Additionally, using an externally designed and scrutinized security mechanism minimizes our own chances for introducing security flaws. This approach is thus very appealing for our architecture.

4.3 Authentication Solution

Using WPA-Enterprise, we can support username and password based authentication and per-user encryption keys to identify flows and apply policy, without requiring changes to client software. The use of per-user keys also provides increased isolation between wireless clients, preventing eavesdropping and attacks on data integrity. The primary challenge is placement of the RADIUS servers, to avoid overwhelming home users or requiring additional hardware.

In the long term, it is clear that ISPs can maintain their own RADIUS servers, and each AP can be configured to contact these servers to determine whether to admit a new client. This approach places trust appropriately, as clients need only trust their ISP with their password and not the AP.[1]

In the short term, each router owner must configure a local RADIUS server with accounts for known clients. Fortunately, the programmability of wireless routers allows self-contained RADIUS servers to be loaded directly on the routers themselves. This is demonstrated by the tinyPEAP project [13], which provides a web-based interface for administering a RADIUS server running on a Linksys WRT54G router. Using a local server, clients must be added manually and they must trust the AP, but this is reasonable on a small scale.

To transition between local and global RADIUS servers, the AP can check if a user is known locally before

[1]Using 802.1x, clients can use their ISP's public key to encrypt their password for authentication, preventing rogue APs from stealing passwords.

deferring to a global server, possibly treating local clients as part of higher priority classes. This model also allows for an altruistic policy of admitting unknown clients to use any leftover bandwidth.

Overall, the use of WPA-Enterprise can support each of the requirements for sharing connectivity, with straightforward management techniques and no software changes to clients.

5 Resource Allocation

The second fundamental building block for sharing connectivity with commodity routers is a scheme that provides robust resource allocation for clients. Effective resource sharing provides important guarantees to AP owners that allowing other clients to connect will not unfairly deprive themselves of bandwidth.

Although in this setting resources are to be allocated amongst different classes of users, this scenario is significantly different from traditional class based approaches for establishing resource allocation. This arises from the fact that the resource being shared in a wireless AP is not just the bandwidth of the link to the Internet, but also the wireless air time provided by the AP, and considering one without the other renders all classical mechanisms ineffective. This added dimension introduces a significant amount of complexity that motivates the need for more sophisticated queuing schemes that can provide quality of service guarantees in this setting.

5.1 Characterizing the Shared Resources

The typical layout of a wireless AP is shown in Figure 2. The AP is connected to the Internet via a DSL router or cable modem. The AP-modem link is generally a 100 Mbps or 1 Gbps ethernet link, while the modem typically has a lower bandwidth connection to the Internet. The AP is shared among multiple clients that may include wired machines as well. This section characterizes the various resources being shared among clients of different priorities and the issues involved in doing so.

Sharing The Internet Link As evident in Figure 2, the router is ignorant of the modem-Internet link, since this is completely masked by the (much) higher bandwidth link to the modem. Any resource allocation mechanism that fails to consider this issue is ineffective, because the modem will drop excess traffic indiscriminately, without respect to any policy implemented at the router. This problem is exacerbated by the large bandwidth of the AP-modem link, since it is unlikely queuing will ever occur at the AP.

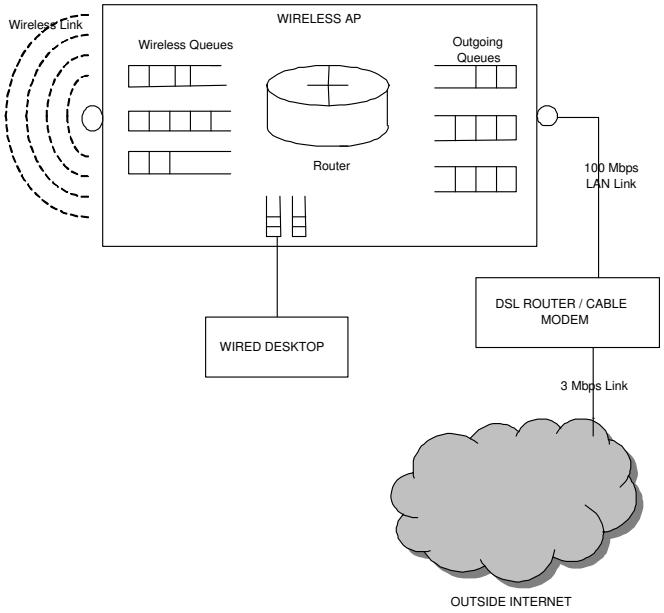


Figure 2: Resources and Sharing in the Typical Wireless AP Setup

Sharing the Wireless Link The nature of wireless transmission adds another layer of complexity to our picture of resource allocation. The problem arises because not all wireless clients communicate with the AP at the same rate. The lossy nature of the wireless link motivates the clients to adjust their sending rate so that they minimize their number of losses. This rate is chosen in a greedy manner by each client, and it typically depends on its signal strength with the AP. Clients with weaker signals will be susceptible to greater loss rates and will therefore choose to send at a lower transmission rate. This phenomenon has two very important consequences. First, as discussed by Tan and Guttag [29], this lowers the aggregate throughput of the AP on the outbound link. Second, this prevents traditional resource allocation on the outbound link, as discussed below.

Consider two clients connected to an AP and sending to the AP at different rates as shown in Figure 3. Further, suppose that the clients have reached a stable flow rate where they are each sending about the same amount of data to the AP. Since client A can send at a much higher rate than client B, it takes up much less air time to send its packets. However, the packets leaving the AP towards the Internet are going out at a fixed rate that is higher than the rate that B is using. Therefore, as illustrated in the figure, since B takes longer to get a packet to the AP than for the packet to leave the AP, A is effectively constrained by B's sending rate. Also, the outbound link can never realize its full throughput, since all senders are constrained by the slower sender, who has a slower rate

than the outbound link. Since each sender can send at rates ranging anywhere from 1 to 54Mbps, modems with speeds around 3 Mbps can be constrained to less than their full capacity.

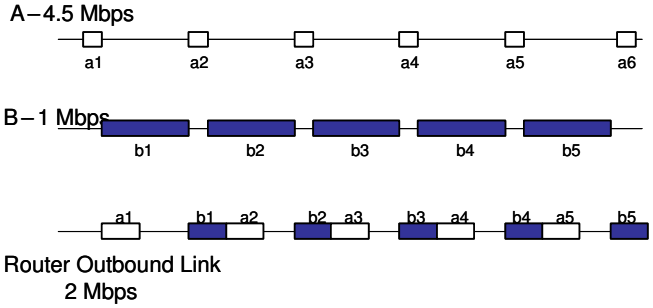


Figure 3: Effects of Variable Rate Wireless Transmission

It is important to note that in this way, slow senders can prevent queuing from occurring within the router. Thus, traditional queuing mechanisms will not be effective for resolving the problem, and we must instead take the effects of multiple bit rates into account.

Upstream and Downstream Traffic We introduce the notion of upstream and downstream traffic with respect to the AP at this stage because this distinction is an important factor in further development. We refer to traffic going from the clients to the Internet as *upstream* traffic, and traffic from the Internet to the AP as *downstream* traffic. This is necessary to reason about the fact that in the router, these flows are essentially separate, and have to be dealt with as such. However, in reality, upstream traffic competes with downstream traffic in the air, and resource allocation mechanisms have to take this into account to be effective.

Wired LAN Clients The presence of wired clients on a typical LAN increases the complexity of our resource allocation picture. The fact that wired clients can send at a rate that can potentially swamp the outgoing link very quickly makes them another factor to be carefully considered. Wired clients also can be differently prioritized and their traffic has to be treated on par with that from wireless clients in comparable classes.

5.2 Wireless Weighted Fair Queuing (WWFQ)

As discussed in Section 5.1, dealing with multiple transmission rates and classes at the same time is central to sharing resources. The resource allocation problem, based on our discussion thus far, may be characterized as follows. Consider a set of N client nodes $n_1, n_2, ..., n_N$, each of which belongs to a class $c \in C$ (the set of all classes) with priority α_c. Each node communicates with the AP at a rate r_i, for $i \in 1, 2, ..., N$. Note that a higher value of α_c denotes a higher priority class. Then, the resource allocation requirement is to ensure that the share b_i of the bottleneck Internet bandwidth for client n_i is proportional to its *normalized* class priority. Our solution to this problem is a flow-based, two-level queuing scheme called Wireless Weighted Fair Queuing (*WWFQ*), which uses both ingress and egress mechanisms. We initially discuss the solution considering only upstream flows, and then describe the minor modifications necessary to handle downstream flows, as well as flows in both directions at the same time.

5.2.1 Ingress Queuing

Our ingress mechanism is designed to throttle clients with slow rates, so that enough packets can be queued at the egress link for intelligent policy to be applied. The criterion for throttling senders is based on ensuring that each client gets a share of *air time* that is proportional to its priority. Figure 4 lays out the intuition behind this criterion. In this example, the clients operate as in Figure 3, except that the router actively drops a packet from the flow of the slower sender. After a round trip time, TCP will cause this sender to back off, thereby leaving the channel open for longer than before. This gives the faster sender a longer window through which to send more data, likely leading to queuing at the egress bottleneck. Therefore, when the slower sender does get a packet through, it has to compete with the queued packets from the faster sender, and queuing mechanisms can be applied at the egress to enforce fair sharing. Note that this is effectively the same as apportioning a larger share of the air time to the faster sender so that it can get more of its packets through. Therefore, the fundamental idea of our ingress queuing is to consider air time as a central resource to be shared, and share it proportionately among the senders.

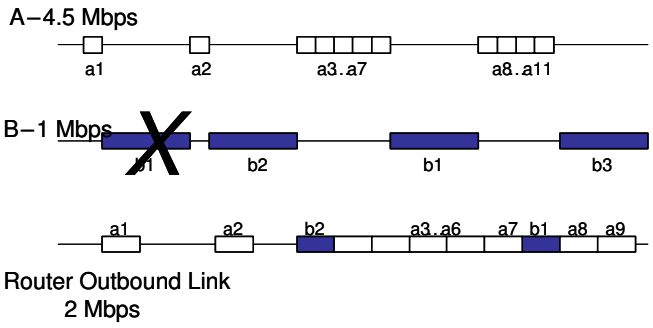


Figure 4: Dropping Packets to Handle Different Wireless Transmission Rates

This means that the share of the time t_i received by client n_i is also proportional to the normalized priority of the client's class. This is enforced by an ingress queuing mechanism which drops an incoming packet when:

$$\frac{D_i}{r_i} > \frac{\alpha_c(T - T_0)}{\sum_c \alpha_c N_c} \quad (1)$$

where T_0 is a reference point time in the past, T is the current time, N_c is the number of clients in class c, α_c is the priority assigned to class c, D_i is the amount of data sent so far by client n_i, and r_i is the rate at which client n_i is sending. In other words, the ingress queuing mechanism drops packets from any flow that exceeds the amount of data it is allowed to send in a given time interval. Note the normalized weight factor for the time share with respect to the number of flows in each class in our system. This is to ensure that *each* flow in a higher class does proportionately better than every flow in a lower class.[2] One caveat is that in practice, the rate r_i is not the same as the rate set by hardware on the flow, but is instead some observed fraction that depends on the channel conditions. For example, a client configured to send at 1 Mbps often has an observed *base rate* of 0.78 Mbps. It is this observed base rate that has to be used by the queuing mechanism. However, this can be easily estimated using an exponential average smoothing scheme for each sending client.

5.2.2 Egress Queuing

Our egress queuing mechanism operates on the outgoing interface from the AP and serves two goals. The first is to perform packet shaping to cause packets to be dropped at the router according to policy, rather than at the modem indiscriminately. The second is the egress policy itself, which is the mechanism that is used to ensure fairness of the shared link.

The first of these objectives is solved by using tools such as *pchar* [17] to estimate the actual modem bandwidth B, and then by adding a token bucket filter with that rate to the interface. Thus, all outgoing traffic leaves the router at the bottleneck rate and will not encounter additional queuing at the modem.

For the second objective, the egress mechanism chooses a packet from among the various flows in a manner that reflects the priority of the flow. At a first glance, standard Weighted Fair Queuing (*WFQ*) [8] would appear an ideal solution. However, senders with different rates again complicate the scenario, since the queue for each flow fills at the rate determined by the ingress queuing mechanism. Since the modem bandwidth B is typically smaller than the rate at which wireless clients can send to the AP, queuing will occur in the router. In the presence of senders with different bit rates, standard WFQ will distort the priorities of each flow, as illustrated in the following example.

Suppose there are two flows passing through the router: a high priority flow that sends at a fast rate, and a low priority flow that sends at a slow rate. Then, if the egress queue drains based on priority alone, the high priority flow experiences more queuing, since it is sending at a higher rate, and because it is getting more time due to the ingress filter. Therefore, it will more quickly encounter drops, forcing it to back off. This means that the low priority low rate flow gets more airtime as well as more egress bandwidth than it should, which goes against the fairness requirement.

Two Level Queuing Scheme The above example motivates a queuing scheme that not only achieves isolation of flows, but also isolation based on the rates of the various flows. We propose a two level scheme that achieves this objective. The rates are incorporated into a Fair Queuing algorithm just like class priorities are in WFQ — here packets with higher bit rates are made to look smaller to the algorithm, and they are thus chosen for transmission sooner.

The lower level of the queuing scheme establishes an isolation between flows based on rate for all the flows *within a single class*. This is accomplished using the following equation for calculating the finish time of a packet within a flow:

$$F_p^i = S_p^i + \frac{1}{r_i} L_p^i$$

where F_p^i and F_p^i are the finish time and start time of the p-th packet in the i-th flow respectively, r_i is the rate of the i-th flow, and L_p is the length of the p-th packet in the i-th flow. The lower level of the scheme uses this equation to select which packet will finish first, based on rate.

The higher level of the queuing scheme then chooses from among these candidate packets (one per class) in order to effect the desired isolation between classes. It chooses the packet from the flow with the lowest finish time as defined by the following:

$$F_p^i = S_p^i + \frac{1}{r_i \alpha_c^i} L_p^i$$

where α_c^i is the priority of the class that flow i belongs to. Note that the finish times are scaled here by the rates as well for exactly the reason mentioned before: low rate flows in a lower class should not be allowed to shut out fast flows in higher classes.

[2]Note that this assumes a certain model of fairness that is not entirely based on isolation. However, we argue that in the economic incentive model in which we envision this system, this is the most fitting type of fairness, because a customer who has paid a certain amount for a service expects to receive twice as much service as *any* other customer who has paid half that amount.

Observations The two level queuing scheme exhibits a few desirable properties. First it is work conserving — we are guaranteed to get the best possible utilization of the bottleneck link since the link is divided up between all *existing active* flows. Second, under the fairness criterion described above, it establishes the desired isolation and fair allocation of resource among flows of different classes, with respect to both the air time and bottleneck link.

5.3 Wired Clients

So far, we have only considered the case where there are only wireless clients sending data to the Internet. We propose that the above egress scheme can be easily extended to include the case of wired clients connected to the AP on the LAN. For wired clients, the only concern is that their bandwidth share should reflect solely their class priority, since they do not compete for wireless air time. To incorporate wired clients, we create another level to our egress queue. This level chooses between the result of the egress queue described above for the wireless clients and from among all the wired clients. The fair queuing finish time criterion is now computed differently for wired and wireless clients. For wireless clients, we have:

$$F_p^{i,wireless} = S_p^{i,wireless} + \frac{\alpha_c^i}{\sum_{k:k\in C} \alpha_k N_k^{wireless}} L_p^i$$

where we scale the finish time of a packet from a wireless client by its priority (α_c^i), with respect to all the wireless clients ($N_k^{wireless}$) in the various classes connected to the AP. This causes the chosen packet to be representative of all classes with active wireless flows. Similarly, the finish times for the packets from the wired clients are computed according to:

$$F_p^{j,wired} = S_p^{j,wired} + \frac{\alpha_c^j}{\sum_{l:l\in l} \alpha_l N_l^{wired}} L_p^j$$

We posit that this scheme allocates the bandwidth fairly among wired and wireless clients.

5.4 Upstream and Downstream Flows

Our discussions above have derived the WWFQ algorithm considering upstream flows alone. However, it is easy to see that the egress scheme alone can be applied to downstream wireless flows, as it controls which packets are transmitted to wireless clients based on rate and priority. Ingress queuing is not necessary, since all packets are received from the Internet at the same bit rate.

Additionally, as previously mentioned, the wireless air time is really being shared between upstream and downstream traffic at once, since wireless clients cannot send and receive at the same time. We can easily extend Equation 1 in Section 5.2.1 to include *all* packets from flows in both directions (i.e., the data either sent or received by a client on a wireless link counts towards its received time share). Ideally, this is akin to maintaining a common pool of queues on the wireless link from which packets are chosen to either be dropped, transmitted to the clients (downstream), or forwarded to the egress queuing scheme (upstream).

5.5 Note on Related Work

It is worth noting that Tan and Guttag devise a similar scheme for air time fairness in [29]. We unfortunately were not fully aware of this work until after independently deriving and evaluating our own mechanisms. While coming to similar conclusions regarding methods of sharing air time, their scheme does not address different classes of service, and it requires driver modifications (preventing its deployment on Linksys routers), as we note in Section 2.2. Essentially their scheme is a subset of the ingress mechanism described in Section 5.2.1.

6 Evaluation

In this section, we describe the performance of our resource sharing mechanisms, as implemented in the Linux kernel on an actual wireless testbed.

6.1 Experimental Topology

We implemented the egress scheme as a new queuing discipline for the Linux kernel, allowing us to load it dynamically using the *tc* command. While our ingress mechanism could be implemented similarly, for simplicity, we chose to approximate it using ingress token bucket filters for our experiments. Equation 1 can be used to compute token bucket filter rates appropriate for each experiment, given a static set of flows.

We deployed our implementation on a PC running Linux, rather than directly on a Linksys WRT54G router. This machine was configured to behave the same as a wireless router, allowing us to avoid debugging troubles with the router itself.

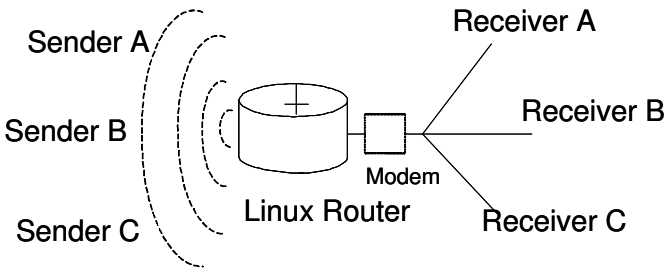


Figure 5: Topology Used to Test WWFQ

Figure 5 shows the topology that we used to test WWFQ. The router has three wireless clients sharing the wireless link, and each client connects to a separate wired receiver behind a simulated Internet bottleneck of 3 Mbps. The rates and classes of each client were varied for the experiments.

To ease the evaluation, we made a number of other simplifications. Notably, our test code uses pre-configured knowledge of rates and classes, rather than detecting this information dynamically. We also do not consider wired clients connected to the wireless router.

Additionally, we chose to only test heavy upstream traffic, both because downstream traffic is fairly symmetric, and because of a quirk in configuring experiments. That is, clients can be reliably configured to send at fixed bit rates, but traffic from the AP to clients do not respect these configured rates. Instead, the AP appears to send as fast as the client actually supports, making it difficult to set up experiments with specific downstream bit rates. Note that this quirk should not be a problem in practice, as long as the AP can observe its downstream rates.

Our implementation does not currently support heavy simultaneous upstream and downstream traffic, since we did not immediately see an easy way to apply queuing policy to flows in opposite directions at the same time in Linux. However, this may later be accomplished using special queuing mechanisms or our own global data structure, as discussed in Section 7.

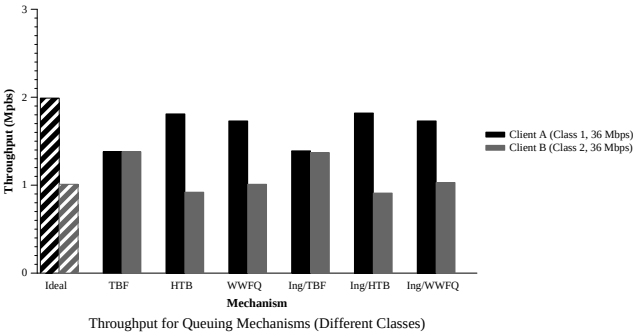
6.2 Results

We conducted several simple experiments to evaluate the fairness and link utilization of our queuing mechanisms, compared to other standard mechanisms in the Linux kernel. For each experiment, we used Netperf [25] to measure the TCP throughput for each concurrent connection, using each queuing mechanism. We report the average of three 10 second trials for each result.

All experiments used the equivalent of a Token Bucket Filter (*TBF*) with a rate of 3 Mbps as the root queuing discipline, shaping packets for a simulated cable modem. Beyond this shaping, we considered cases with no additional queuing (marked "TBF"), a hierarchical token bucket filter (marked "HTB")[3], and our egress Wireless Weighted Fair Queuing mechanism (marked "WWFQ"). Both HTB and WWFQ trials were configured to give clients in Class 1 twice the priority of clients in Class 2. Additionally, we ran each of TBF, HTB, and WWFQ with and without our approximated ingress mechanism, which we here refer to as "Ing" or "Ingress".

Figure 6 shows the throughputs of two clients in different classes sending at the same bit rate. The ideal ratio of

[3]HTB provides similar functionality as class based queuing, with simpler configuration requirements.



Throughput for Queuing Mechanisms (Different Classes)

Figure 6: Throughput for two wireless clients in different resource classes to hosts behind a 3 Mbps bottleneck, for various queuing mechanisms. Classes 1 and 2 have a 2:1 priority ratio.

throughputs (based on air time fairness) is depicted with striped bars, and in this case it matches throughput-based fairness. As one would expect, both HTB and WWFQ easily achieve fairness, with and without ingress filtering. While the average throughputs for TBF alone are approximately equal for each flow, the actual throughputs for each flow vary from trial to trial. This is expected, given TBF's ignorance of individual flows.

We measure throughputs of two clients in the same class sending at different bit rates in Figure 7. The three mechanisms without ingress filtering each clearly suffer reduced utilization due to the low bit rate client. Even worse, the fair queuing schemes provide greater throughput to the slower sender, since the faster sender backs off TCP's multiplicative decrease more often. Using ingress filtering, however, the slower client backs off and allows greater throughputs for the faster client. While the ideal rates are not achieved by any mechanism, WWFQ comes closest to the proper ratio, and it is the only mechanism to queue up and drop the slower client's packets to achieve greater air time fairness.

Figure 8 combines the above scenarios, with a fast and slow client in Class 1 and a fast client in Class 2. The slightly unexpected ideal throughputs are a result of a 2:2:1 air time ratio for clients A, B, and C; client B simply has a very low throughput even when using 40% of the air time, because it sends at a much lower rate. As in Figure 7, an ingress filter is necessary to cause the slow client to back off initially, and the WWFQ egress mechanism again is the only one to drop packets from the slow client, closely matching the ideal throughputs.

Finally, in Figure 9 we look at the aggregate throughputs for each queuing mechanism, for each of the above scenarios. Trials using ingress filtering are shown with striped bars, clearly indicating higher uplink utilization in the presence of mixed bit rates. Slight decreases in aggre-

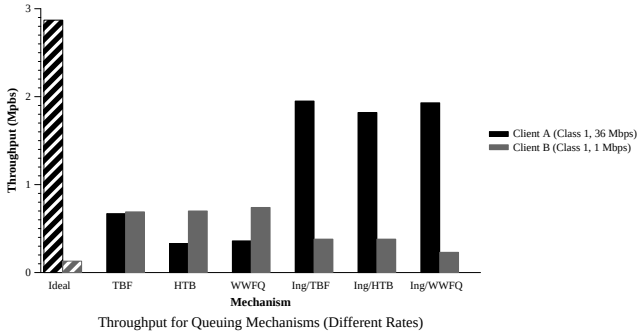


Figure 7: Throughput for two wireless clients using different bit rates (36 Mbps and 1 Mbps) to hosts behind a 3 Mbps bottleneck, for various queuing mechanisms. An ingress filter is needed to approach the ideal throughput ratio.

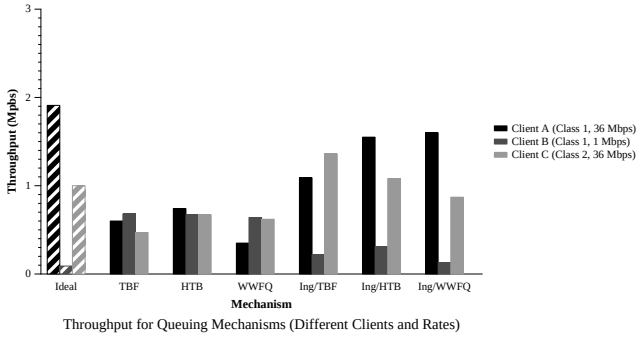


Figure 8: Throughput for three wireless clients with different classes and different rates to hosts behind a 3 Mbps bottleneck, for various queuing mechanisms. Both ingress and WWFQ egress mechanisms are needed to achieve the ideal throughput ratio.

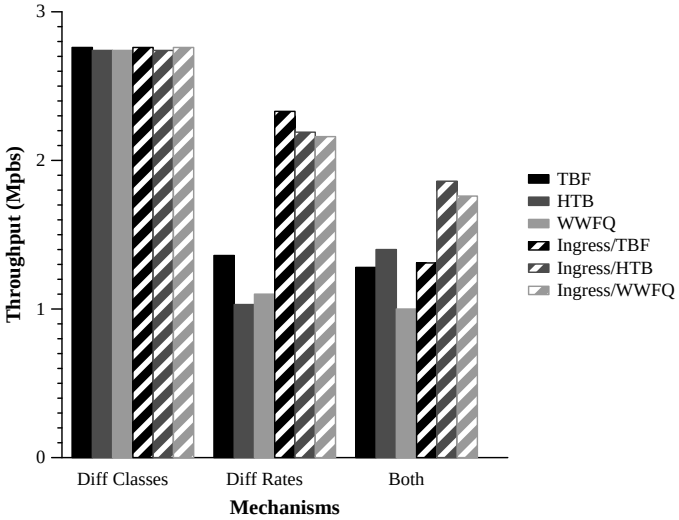


Figure 9: Aggregate throughput for wireless clients using different queuing mechanisms. An ingress filter significantly improves utilization in the presence of clients using different rates.

gate throughput from Ingress/HTB to Ingress/WWFQ are likely the result of the approximations used to calculate ideal air time. While such inaccuracies would likely be present in deployed solutions as well, they do not represent a significant difference in aggregate throughput.

Overall, we see that our proposed ingress and egress queuing mechanisms provide fair class-based sharing of air time and high link utilizations, compared to existing techniques. While the results shown are for upstream throughput, we expect similar results for downstream. The combination of heavy upstream and downstream traffic requires additional mechanisms, though, as discussed in Section 5.4.

7 Future Work

While experimental results confirm that WWFQ provides fair sharing and good utilization for upstream flows, we would like to devise experiments that could show similar results for downstream flows, given the unexpected behavior of APs described in Section 6.1.

More significantly, we need to devise Linux-based implementations of the ingress filter, as well as the mixed upstream/downstream solution. Our existing mechanisms would be mostly effective if traffic is light in at least one direction, but proper coordination is necessary to handle heavy traffic in both directions at once. There is perhaps promise in using the Intermediate Queuing Device (*IMQ*) [15] queuing mechanism for Linux, which allows policy to

be applied to multiple devices, or an alternative of using a global kernel data structure protected by locks.

Finally, we would like to port our mechanisms to the Linksys WRT54G wireless router to deploy a usable solution. Most of the infrastructure for supporting our mechanisms is already in place on the router, and porting our code to the router is straightforward, thanks to the distributions provided by Linksys and Sveasoft. However, it may be more difficult to tie together other loose ends, such as observing the bottleneck and client bit rates or securely identifying packets based on authentication. These tasks would be straightforward with a modifiable driver, but the WRT54G's binary-only driver requires us to explore clever alternatives. Nonetheless, we feel a practical and effective implementation is feasible.

8 Conclusion

We have outlined the key aspects of an architecture to use commodity wireless routers to promote ubiquitous Internet connectivity, together with possible incentives for sharing network access. By leveraging the ability to upgrade router software, our architecture addresses the security and resource allocation challenges that arise when sharing network access with unknown clients. We use appropriate authentication and encryption techniques, along with a new set of queuing mechanisms that provide fair sharing and high link utilization for wireless TCP flows. Although certain aspects of the queuing mechanisms remain to be tested, they offer a promising solution that does not require client-side software changes. As a result, we hope to promote the use of such upgraded routers for sharing connectivity on a large scale.

9 Acknowledgments

We are very grateful to Mike Swift and Muthu Annamalai for their help in setting up VMWare and sharing kernel debugging expertise. We would also like to thank David Wetherall and Valentin Razmov for their helpful feedback throughout the project.

References

[1] "802.1x - Port Based Network Access Control." IEEE 802 Standard.
(URL: *http://www.ieee802.org/1/pages/802.1x.html*)

[2] P. Bahl, A. Balachandran, and S. Venkatchary. *Secure Wireless Internet Access in Public Places. Proceedings of the IEEE Conference on Communications*, Helsinki, Finland, June, 2001.

[3] A. Banchs, X. Perez. *Distributed weighted Fair queuing in 802.11 wireless LAN. ICC*, 2002.

[4] "Bay Area Research Wireless Network."
(URL: *http://www.barwn.org*)

[5] "Boingo Wireless."
(URL: *http://www.boingo.com*)

[6] N. Borisov, I. Goldberg, D. Wagner. *Intercepting Mobile Communications: The Insecuriy of 802.11. Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, July, 2001.

[7] J. Buzbee. "Linux on the WRT54G."
(URL: *http://www.batbox.org/wrt54g-linux.html*)

[8] A. Demers, S. Keshav, S. Shenker. *Analysis and Simulation of a Fair Queueing Algorithm. ACM SIGCOMM*, Austin, TX, 1995.

[9] "frottle."
(URL: *http://frottle.sourceforge.net*)

[10] J. Geier. "802.11 WEP: Concepts and Vulnerability."
(URL: *http://www.wi-fiplanet.com/tutorials/article.php/1368661*)

[11] E. Griffith. "WPA: New Protection for 802.11."
(URL: *http://www.wi-fiplanet.com/news/article.php/1491771*)

[12] T. Jans. "HyperWRT."
(URL: *http://www.hyperdrive.be/hyperwrt/index.php?page=homepage*)

[13] B. Lee, T. Takahasi, W. Lee, R. Lipton. *TinyPEAP.*
(URL: *http://www.tinypeap.com/docs/TinyPEAP_White_Paper.pdf*)

[14] "Linksys WRT54G - Wireless-G Broadband Router."
(URL: *http://www.linksys.com/products/product.asp?prid=508&scid=35*)

[15] "Linux IMQ."
(URL: *http://www.linuximq.net*)

[16] S. Lu, V. Bharghavan, R. Srikant. *Fair scheduling in wireless packet networks. IEEE/ACM Transactions on Networking*, 1999.

[17] B. Mah. "pchar: A Tool for Measuring Internet Path Characteristics."
(URL: *http://www.kitchenlab.org/www/bmah/Software/pchar/*)

[18] T.S.E. Ng, I. Stoica, H. Zhang. *Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors INFOCOM*, 1998.

[19] "NoCatNet."
(URL: *http://nocat.net*)

[20] "NYCwireless."
(URL: *http://www.nycwireless.net*)

[21] "OpenWRT."
(URL: *http://openwrt.org*)

[22] "Patras Wireless Network: Wireless Central Coordinated Protocol."
(URL: *http://www.patraswireless.net/software.html*)

[23] C. Perkins. "RFC 2002: IP Mobility Support."
(URL: *http://www.ietf.org/rfc/rfc2002.txt*)

[24] L. Phifer. "Using RADIUS for WLAN Authentication, Part I."
(URL: *http://www.wi-fiplanet.com/tutorials/article.php/3114511*)

[25] "The Public Netperf Homepage."
(URL: *http://www.netperf.org*)

[26] "Seattle Wireless."
(URL: *http://seattlewireless.net*)

[27] "SpeakEasy NetShare."
(URL: *http://www.speakeasy.net/netshare/learnmore/*)

[28] "Sveasoft."
(URL: *http://www.sveasoft.com*)

[29] G. Tan, J. Guttag. *Time-based Fairness Improves Performance in Multi-rate WLANs. Proceedings of USENIX*, Boston, MA, June 2004.

[30] G. Tan, J. Guttag. *Long-term Time-share Guarantees are Necessary for Wireless LANs. SIGOPS European Workshop*, Leuven, Belgium, September, 2004.

[31] N.H. Vaidya, P. Bahl, S. Gupta. *Distributed fair scheduling in a wireless LAN MOBICOM*, Boston, MA, 2000.

[32] "What is TKIP (Temporal Key Integrity Protocol)?" (URL: *http://www.tech-faq.com/wireless-networks/tkip-temporal-key-integrity-protocol.shtml*)

[33] "WiMAX Forum." (URL: *http://www.wimaxforum.org/home*)

[34] "Wireless Internet Access - T-Mobile HotSpots." (URL: *http://www.t-mobile.com/hotspot*)