Traffic Management and Resource Allocation in Small Wired/Wireless Networks

Christos Gkantsidis
Microsoft Research
Cambridge, UK
chrisgk@microsoft.com

Thomas Karagiannis
Microsoft Research
Cambridge, UK
thomkar@microsoft.com

Peter Key
Microsoft Research
Cambridge, UK
peterkey@microsoft.com

Bozidar Radunovic
Microsoft Research
Cambridge, UK
bozidar@microsoft.com

Elias Raftopoulos
FORTH, Greece
eraftop@ics.forth.gr

D. Manjunath
IIT Bombay, India
dmanju@ee.iitb.ac.in

ABSTRACT

We consider the problem of traffic management in small networks with both wireless and wired devices, connected to the Internet through a single gateway. Examples of such networks are small office networks or residential networks, where typically traffic management is limited to flow prioritization through port-based filtering.

We propose a practical resource allocation framework that provides simple mechanisms to applications and users to enable traffic management functionality currently not present due to the distributed nature of the system and various technology or protocol limitations. To allow for control irrespective of whether traffic flows cross wireless, wired or even broadband links, the proposed framework jointly optimizes rate allocations across wireless and wired devices in a weighted fair manner. Additionally, we propose a model for estimating the achievable capacity regions in wireless networks. This model is used by the controller to achieve a specific rate allocation.

We evaluate a decentralized, host-based implementation of the proposed framework. The controller is incrementally deployable by not requiring modifications to existing network protocols and equipment or the wireless MAC. Using analytical methods and experimental results with realistic traffic, we show that our controller is stable with fast convergence for both UDP and TCP traffic, achieves weighted fairness, and mitigates scheduling inefficiencies of the existing hardware.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Management; C.2.3 [**Network Operations**]: Network Monitoring

General Terms

Algorithms,Design,Measurement,Performance

CoNEXT'09, December 1–4, 2009, Rome, Italy.

Keywords

home networking, congestion control, rate control, weighted fairness, wireless networks, quality-of-service

1. INTRODUCTION

We consider the problem of traffic management in small networks comprising a mix of wireless and wireline hosts connecting to the Internet through one central gateway. Examples of such networks include those in small offices and residential networks, in which case connectivity to the Internet might be provided through a broadband access link. In such networks, hosts are often connected in an ad-hoc manner, typically through a single device, for example a small router and/or Access Point (AP). These devices may provide some form of minimal centralized control or could implement any desired policies (e.g., prioritization of certain types of traffic). Our aim is to significantly extend the existing management functionality to allow users and applications to exert control over the network in scenarios that currently cannot be enforced. This is especially relevant for small business networks where users have some notion of the intrinsic business value of each application's traffic.

Traffic management in these networks is problematic. Control typically translates to application prioritization which is enforced at the central gateway. This form of control is achieved crudely, through port-based filters, and can be highly ineffective or inaccurate for applications using arbitrary port numbers such as Skype or BitTorrent; in these cases, packet payload inspection becomes necessary. Low-cost devices providing payload classification functionality are rarely found in these networks. Even if available, such devices are not easily extensible, with policies that focus on specific application types (e.g., gaming traffic) and cannot be modified. Additionally, policies are enforced irrespective of the underlying technology, that is without taking into account whether hosts are wireless or wired which may lead to performance degradation for the overall network.

A traffic controller in a small network needs to address three main challenges:

- *Sharing the wireless medium.* While in wired networks network queues are shared across all participating hosts in the network, queues are distributed in the case of wireless networks. Thus, the notion of a single queue where centralized rate control might be applied is not present. Further, wireless hosts obey a predetermined MAC protocol which is built-in, cannot easily change, and is known to be inefficient [10].

- *Sharing across diverse scheduling disciplines.* Different network devices might obey different scheduling disciplines that cannot be easily modified (or even known to users and applications). For example, most devices such APs or home routers, obey FIFO schedulers that may cause unfairness in the network.

- *Application priority scheduling.* Applying application-specific policies in devices such as routers or APs is hindered by several factors such as the usage of arbitrary transport-layer port numbers, encrypted packets, etc. This is especially true in the considered networks where low-cost devices prevail and traffic may cross from wireless to wired and broadband links.

The key contribution of this paper is a practical control framework for small networks that i) allocates resources among flows, hosts or applications according to a desired policy in a decentralized manner, ii) mitigates various deficiencies of different schedulers present in today's networks (e.g., WiFi MAC, FIFO at devices), iii) is incrementally deployable and requires no modification of the existing network architecture (i.e. no changes to the MAC, network, or transport layers, except for support for rate limiting which is already provided in most current operating systems) and iv) it does not presuppose control over network devices such as APs, routers or broadband modems. Instead, our solution is based on host coordinated rate control.

We first describe our controller which is based on weighted max-min fair allocations. Resource allocations are in proportion to a per-application weight, thus enforcing a form of weighted max-min fairness. We argue that this form of control is the most flexible in small wired/wireless networks. Hard priority mechanisms, such as strict or preemptive priorities have several disadvantages, such as possible starvation or unbounded performance for lower priority traffic, and potential priority inversion when there are several resources.

To achieve allocations in WLANs, we derive a practical model for the capacity of a wireless network as a function of the access rates of the associated devices and the rate limits imposed. It is based on existing analytical models (see [5] and follow-up works) and provides a simple and accurate approximation of the feasible rate set of a wireless network, that can be easily parametrized and implemented in a controller.

We implement the proposed controller in a decentralized system and evaluate its performance in three sample scenarios. In particular, we show how our controller can i) provide prioritization of wireless flows irrespective of the MAC scheduler, ii) improve network performance by rate-limiting wireless hosts experiencing bad channel quality, and iii) improve network responsiveness by controlling network queues created by bandwidth intensive applications such as peer-to-peer. In each scenario, our controller is fast to converge to the desired policy, is stable irrespective of the application traffic demands and does not under-utilize the network. This is achieved despite the dynamic nature of the network (e.g., WiFi, broadband) and the variable rate of applications such as video streaming.

To our knowledge, our work is the first to provide a practical resource allocation framework that jointly optimizes wired and wireless networks. We further provide experimental evidence that simple extensions of previous theoretical work to characterize achievable wireless throughput regions are applicable in practice. This is even operating at significantly different timescales (e.g., aggregating information at the second granularity) and with hosts being associated at different rates.

2. PROBLEM DEFINITION

In this section, we present simple cases of traffic management that are difficult to support in today's networks, and illustrate the problem we want to solve. We then provide a more formal definition of the problem space under consideration.

2.1 Sample Scenarios

We present three sample scenarios that our control framework enables: i) prioritizing wireless flows, ii) improving wireless network performance, and iii) ensuring network responsiveness by reducing queues in the network. Realizing similar scenarios currently requires modification of network devices and/or protocols (such as the wireless MAC) and some form of coordination among devices. Instead, the proposed framework provides flexible control mechanisms that facilitate the management of a variety of scenarios under the same solution. In Section 5, we demonstrate how each of the examined scenarios may be realized under the proposed controller.

Scenario 1: Prioritizing wireless flows. Our controller facilitates prioritization of wireless flows irrespective of their direction (upstream, towards the AP, or downstream). Consider, for example, the scenario presented in Fig. 1 (left), where two flows (f_1 and f_2) compete for the wireless capacity. Enforcing a higher priority for a flow (say f_2) is currently not feasible even after applying per flow priorities at the AP. This is due to the fact that the wireless MAC is distributed and the scheduler is fair, and hence allocates equal resources for the two flows by design. Such a scenario today could only be achieved with 802.11e which is rare in low-cost APs. 802.11e has also other disadvantages discussed in the following section.

Scenario 2: Improving network performance. In a related scenario, consider the same setup, with the additional complication of the wireless hosts experiencing different channel qualities. For example, in Fig. 1 (middle), the host sourcing flow f_2 is associated at rate 12Mbps with the access point, while the other wireless host is associated at 54Mbps. In such a case, network performance is affected overall due to the properties of the MAC layer; the total network capacity drops to 12Mbps penalizing the host associated at 54Mbps [10]. Providing higher priority for f_2 would improve the total capacity of the network at the same time (of course, at the expense of f_1's rate).

Scenario 3: Improving network responsiveness. Broadband queuing delays can be significant (for example, roundtrip times up to seconds are not unusual [7]), which may have a dramatic impact on network responsiveness. Consider the scenario in Fig. 1 (right), where flow f_1 competes with two other flows (f_2 and f_3), and flows f_2 and f_3 are fast, long file transfers or bandwidth intensive applications (e.g., p2p), while f_1 represents short flows (e.g., a web session). $\mathcal{I}$ denotes the Internet. Assuming that the bottleneck is the last mile link (represented here by typical broadband speeds), queues may build up in two places in the network, namely at the broadband downstream link, as well as at the downstream wireless link. In such a scenario, web packets will be queued after a series of f_2 and f_3 packets which will result in poor user experience, observable, for example, by web pages taking significant time to load. Removing such queuing from the network to ensure responsiveness today is feasible only by modifying the AP architecture. Similarly, removing queues building up in downstream broadband link requires modification of the first hop upstream router of the broadband provider. Finally, reducing network queues results in less packet losses which in turn increases the goodput (see Section 5.4 for an example).

Additionally, several network devices such as NICs, routers, modems or APs often implement a FIFO scheduler. Assuming that this is

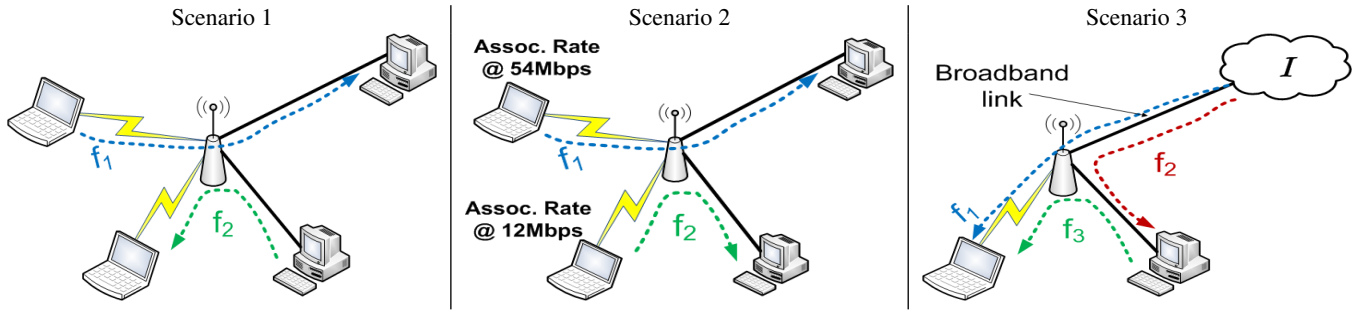

Figure 1: Illustration of the sample scenarios

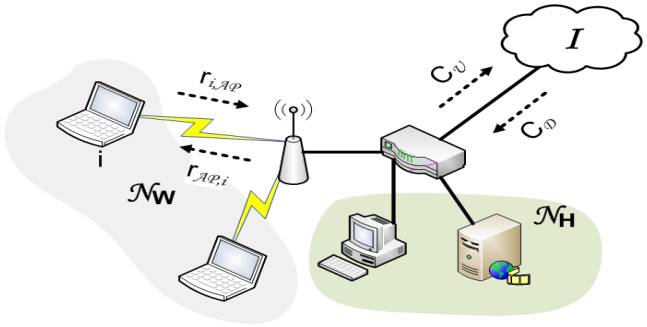

Figure 2: Example of a simple local network

the case in the preceding scenario, if flow f_3 is of a higher rate than f_1 (e.g., due to slow broadband connection), f_3 will overload the queue of the AP, resulting in most of the f_1 packets being dropped. If the difference in rates is significant, this may even result in TCP timeouts and f_1 being dropped. Instead, by reducing queues in the network and appropriately rate-limiting f_3, we can prevent this gross unfairness, without changing the existing hardware.

2.2 Problem definition

Consider a local network composed of a wireless subnetwork, a wireline subnetwork, and a connection to the rest of the Internet (see Fig. 2). We denote by $\mathcal{N}$ the set of active hosts, some of which, say $\mathcal{N}_W \subseteq \mathcal{N}$, connect using the wireless subnetwork, and the rest connect through wireline, $\mathcal{N}_H \subseteq \mathcal{N}$. For convenience, we shall assume that each host connects to the network using only one interface, hence $\mathcal{N}_H \cap \mathcal{N}_W = \emptyset$ and $\mathcal{N}_H + \mathcal{N}_W = \mathcal{N}$.

For those hosts connected to the wireless network, let $r_{ij}(t)$ be the association rate which denotes the nominal data rate that i uses when transmitting data to j at time t (either i or j is the wireless AP). For simplicity, we shall assume that the local wireline network is over-provisioned, and hence we shall ignore capacity limitations in the wireline subnetwork (hence, for brevity, we ignore the case of powerline and related technologies).

We assume a set of active unidirectional traffic flows $f \in \mathcal{F}$, with $\mathrm{Src}(f) \in \mathcal{N} \cup \{\mathcal{I}\}$ and $\mathrm{Dst}(f) \in \mathcal{N} \cup \{\mathcal{I}\}$ representing the source and destination hosts. $\mathcal{I}$ denotes an endpoint that is not part of the local network, i.e., the endpoint is in the Internet and accessed via the broadband link. We associate a weight w_f with flow f that denotes the relative importance of the flow. We shall assume that the weights w_f are known in advance.

Given the set of flows $\mathcal{F}$, their associated weights, the upload and download capacities of the Internet access link (C_U and C_D respectively), and the characteristics of the wireless subnetwork, our goal is to implement an end-host weight-based priority scheduling. Intuitively, this means that flows with higher weights should experience better data rates. More precisely, *our aim is to ensure two flows f and f' have sending rates, say x_f and $x_{f'}$ proportional to their "weights", i.e., $x_f/x_{f'} = w_f/w_{f'}$*, assuming that they share the same bottleneck (wireless or broadband) and that they have enough data to send.

We also want to control the responsiveness of the network. Since we cannot modify the schedulers inside the network, we need to control the amount of queuing in the network. We seek to find an algorithm that balances the responsive (i.e. small queue sizes) versus the utilization (i.e. large queue sizes) of the network. We discuss in detail the motivation behind our two objectives in Section 3.

Observe that this differs from the priority-based medium access for wireless technologies such as 802.11e, which ignores the effect of the association rates. Also note that we have *to jointly* optimize the use of the wireless subnetwork and of the Internet access link.

3. RATE ALLOCATIONS AND CONTROL

This section presents the design of a simple yet efficient controller that addresses the problem defined in the previous section. We first discuss the motivation and overall design goals, and then present the principles of the controller's design. Finally, we explain how the controller can incorporate allocations for WLANs by providing a model to infer the achievable capacity regions.

3.1 Motivation & design goals

The goal of our controller is to be flexible, and operate in dynamic environments where the capacity of the controlled resource fluctuates in time, as in the case of the broadband link or the wireless [15, 7]. Similarly, the type of applications that are typical in these networks further dictate a form of dynamic control that is both automatic and transparent to the user. Applications such as video streaming or peer-to-peer are characterized by variable rate. This implies that enforcing rate control with hard rate guarantees per application (such as reserving fraction of the available capacity) would lead to under-utilizing the network.

We design our controller to adapt to traffic demands and utilize resources efficiently by re-allocating resources. Our controller is thus based on a weighted max-min fair allocation [4], and rates are allocated in proportion to a per-application weight.[1]

As discussed previously, we cannot presuppose any modifications to existing hardware, or protocols (e.g., the wireless MAC or

[1]Or per-{flow, connection, host} depending on the desired level of granularity.

APs). Instead, in order to achieve weighted fairness, rate-limiting traffic flows needs to occur at the wireless hosts. This implies a decentralized controller that is executed at each host. Therefore, our controller should enable the sharing of traffic information among hosts and apply the desired limits at the application layer.

Decentralized control is attractive because hosts have easy access to extensive information regarding the context of their own traffic. Modern operating systems allow easy association of traffic flows to process ids, user ids, and other application information which enables flexible rate control. For example, rate-limiting all traffic generated by a peer-to-peer application is simple at the host, whereas performing the same operation at a switch without host cooperation is practically infeasible. This is because associating traffic flows with specific applications within the network is not possible without deep-packet inspection or traffic classification software, which is rarely found in low-cost network equipment.

Adapting flow rates to traffic demands requires knowledge of whether the various resources (wireless, Internet access uplink and downlink) are saturated. Existing devices offer limited or no information regarding the congestion level of each of these resources. For example, it is quite unlikely for hosts and network interfaces to offer the possibility to peek into MAC level queues in order to estimate congestion (although our framework is able to incorporate that information, should it become available). Hence, we need to explicitly estimate the capacity of each resource, measure the network traffic, and from there deduce the level of congestion; these should be accomplished in a decentralized manner as discussed previously. For the purposes of this paper, we shall assume that the capacity of the broadband access link is known; this may be feasible through the provider of the link, or estimated using active or passive measurements e.g., [8], [14]. We develop a framework to estimate the capacity of the wireless network (explained in detail in are Section 3.4) and incorporate it into the controller design.

Based on the previous discussion and design goals, the following are the main steps in the algorithm of our controller and are repeated in a loop. The algorithm is implemented in each node, executes in parallel and asynchronously (implementation details in Section 6).

1. Collect instantaneous rates of flows at each host and share required information.
2. Determine the utilization of the wireless and wireline resources (Section 3.2.3).
3. Given the utilizations and the flow weights, adjust each flow's rate limit (Section 3.2.2).

The controller implicitly finds a weighted max-min fair allocation (Section 3.2.1). The algorithm is flexible and can also be used to guarantee a rate; this relates to the method of picking the weights. Since the choice of weights is not a core part of the controller, we discuss it in Section 6.

3.2 Weighted fair sharing of resources

We would like our network to behave like a weighted fair queue switch where different flows would have different priorities (weights). However, there are several potential bottlenecks in our network: the wireless access, the Internet uplink and the Internet downlink. One needs to define what is the fair share *across* all the resources. Hence, each flow should get at most the weighted fair share of each resource it uses.

The intuition behind this is as follows. Suppose for the purpose of illustration that the Internet uplink is the most congested resource. The flows that traverse the Internet uplink will share the access capacity according to their weights. Suppose some of these flows also use the wireless network. Since they are already constrained at the Internet uplink, they will get less than what would otherwise be their weighted fair share of the wireless access. The other flows, which use the wireless access but not the Internet uplink, share what remains from the wireless capacity (and get more than what would be their weighted fair share of the wireless access). We now define this weighted fair share more formally.

f	Refers to a (unidirectional) flow.
$f \ni ij$	Refers to a (unidirectional) flow that uses link $i \to j$. We use it to describe flows that use the wireless network. Either i is one endpoint of the flow and j is the wireless access point, or vice versa.
x_f	(Target) Rate of flow f.
$x_{f \ni ij}$	Rate of flow f (x_f) iff f uses (wireless) link $i \to j$, and 0 otherwise.
$\hat{x}_f$	Observed rate of flow f.
w_f	Weight of flow f.
A	Refers to a resource, $A \in \{U, D, \mathcal{W}\}$ denoting uplink, downlink, or wireless respectively.
$\mathcal{F}_A$	Set of flows that use resource A. E.g. $\mathcal{F}_\mathcal{W}$ for the set of flows that use the wireless network.
$\mathcal{X}_A$	Set of feasible rates for resource A. $\mathcal{X}_A \subseteq \Re^{\lvert\mathcal{F}_A\rvert}$
C_A	Capacity of resource A (when applicable). E.g. C_U for the capacity of the Internet uplink.
$\hat{C}_A$	Estimate of capacity of A.
x_f^A	Fair share of flow f in resource A.
x_A	Nominal fair share of resource A.
ρ_A	Utilization of resource A.
$\tilde{\rho_A}$	Target utilization of resource A.
p_A	Congestion price of resource A, typically $p_A = \rho_A^B$ for a small constant B (B=5).

Table 1: List of symbols

3.2.1 Weighted max-min fairness

Let x_f denote the rate of flow f. Let $\mathcal{F}_A$ be the set of flows that use resource A (wireless, uplink, or downlink). Each resource A has a set of feasible rates $\mathcal{X}_A$ and rates $(x_f)_{f\in\mathcal{F}_A}$ are feasible if $(x_f)_{f\in\mathcal{F}_A} \in \mathcal{X}_A$. For example, for uplink U of capacity C_U, the feasible rate set is defined as the set of all possible flow rates such that the sum of all rates is smaller than the link capacity, $\mathcal{X}_U = \{(x_f)_{f\in\mathcal{F}_U} \mid \sum_{f\in\mathcal{F}_U} x_f \le C_U\}$. The set of feasible rates for the downlink $\mathcal{X}_D$ is defined analogously. The set of feasible rates for the wireless, $\mathcal{X}_W$, is also linear (see (8)) but slightly more difficult to define. We address this in Section 3.4.

Let w_f be the weight of flow f. A fair share of resource A for flow f is defined as

$$x_f^A = \min(x_f^{-A}, w_f X_A / \sum_{f'\in\mathcal{F}_A} w_{f'}) \quad (1)$$

where x_f^{-A} is the maximum possible sending rate of flow f assuming that A has infinite capacity (i.e., f is not bottlenecked in A), and X_A is the *nominal fair share* of resource A. X_A is defined as:

$$\max X_A$$
$$s.t. \left\{\min(x_f^{-A}, w_f X_A / \sum_{f'\in\mathcal{F}_A} w_{f'})\right\}_{f\in\mathcal{F}_A} \in \mathcal{X}_A.$$

The rate of flow f is then the smallest of the fair shares of all resources used by f,

$$x_f = \min_{A:f\in\mathcal{F}_A} x_f^A \quad (2)$$

In other words, if the bottleneck for flow f is resource A, then f will get rate

$$x_f = w_f X_A / \sum_{f'\in\mathcal{F}_A} w_{f'}.$$

As a simple example, assume a resource A with capacity C_A. By definition $x_f \le x_f^A$. If all flows $\mathcal{F}_A$ are rate limited at A, i.e., $x_f = x_f^A$, and A is fully utilized, i.e., $\sum_{f\in\mathcal{F}_A} x_f = C_A$, then

$$C_A = \sum_{f\in\mathcal{F}_A} x_f = \sum_{f\in\mathcal{F}_A} x_f^A = \sum_{f\in\mathcal{F}_A} w_f X_A / \sum_{f'\in\mathcal{F}_A} w_{f'} = X_A.$$

If a subset of the flows $\mathcal{F}_A$ are rate limited elsewhere in the network (i.e., not in A), which implies $x_f < x_f^A$ for some f, then to achieve full utilization of A, the following condition should apply:

$$C_A = \sum_{f\in\mathcal{F}_A} x_f < \sum_{f\in\mathcal{F}_A} w_f X_A / \sum_{f'\in\mathcal{F}_A} w_{f'} = X_A$$

By increasing X_A above the capacity C_A, the fair share of all flows increases, and that allows the flows that are bottlenecked at A to claim the excess capacity that is not used by flows that are rate limited elsewhere in the network.

The goal of the rate controller is precisely to estimate the nominal fair share values for all resources (i.e. X_A), and then use it to compute the rate limits for each connection (from (1) and (2)). Underestimating X_A results in very conservative rate limits and hence resource underutilization. Overestimating X_A allows flows to send excessive traffic and create congestion. During the congested periods, the system cannot guarantee allocation of the resource according to the flow weights. Observe that in a more realistic scenario X_A needs to be adapted with time, to keep track of the active network flows and their instantaneous rates. The algorithm for adapting X_A is described in Section 3.2.2. Increases in X_A and the associated rates that do not impact a low utilization is a signal that there is not enough demand for A. In that case, we can safely ignore A, until the observed utilization increases.

It is easy to verify that the resource mechanism described above is the weighted max-min fairness [4], hence the vector $(x_f/w_f)_f$ is max-min fair on the set of all feasible rate vectors in the network. Observe that in this vector, as well as in the discussion above, the absolute values of the weights w_f are not important. Indeed, the allocations depend on the ratio of the weights.

It is well known that the max-min fair allocation can be achieved using a simple water-filling approach [4]. The basis of this approach is that we gradually increase the nominal fair share X_A of each resource A until the resource is saturated. This is done in parallel and independently at all resources. The details are given next.

3.2.2 A rate control algorithm

Each resource A in the system (wireless, Internet uplink and downlink) is assigned a "price" p_A. The price is a dimensionless number between 0 and 1 and denotes how congested the resource is.

Each host has an estimate of the nominal fair share X_A of each resource in the system. It adjusts this estimate based on the following feedback control loop:

$$X_A(t+1) \leftarrow X_A(t) + \kappa X_A(t)\,[1 - p_A(t) - p_A(t)\rho_A(t)] \quad (3)$$

where κ is a gain parameter, and $\rho_A(t)$ is the observed utilization of resource A at time t. The utilization $\rho_A(t)$ is computed as in (4) by using empirical observations. In the simple case of resources with constant capacity (such as the Internet access link), $\rho_A(t)$ equals the ratio of the aggregate rate using resource A at time t over the capacity of A. This algorithm implements a Multiplicative Increase Multiplicative Decrease (MIMD) type control, similar to the primal algorithm proposed in [11]. As explained in Section 3.2.1, the rate of flow f is then assigned to be

$$x_f(t) = \min_{A:f\in\mathcal{F}_A} w_f X_A(t) / \sum_{f'\in\mathcal{F}_A} w_{f'}.$$

It is easy to verify that the update (3) "forces" $\rho_A = (1-p_A)/p_A$; if $\rho_A > (1-p_A)/p_A$ the increase is negative, otherwise it is positive. The algorithm (3) then represents the water-filling algorithm. The nominal fair shares of all resources are increased slowly and independently, until each of them gets saturated and its nominal fair share reaches the steady point.

Observe that in principle the algorithm does not need to know the capacities of the resources, i.e. C_A. It only needs to know the resource utilization, which determined utilization $\rho_A(t)$ and price $p_A(t)$. The resource utilization can be determined by observing the queue sizes, or directly when the resource capacity C_A and the traffic loads $x_{f\in\mathcal{F}_A}$ are known, as we show in Section 3.2.3.

Here, $x_f(t)$ is the *assigned* rate of the connection. In other words, this is the rate which the system will try to enforce on f. The actual rate used by the connection, denoted by $\hat{x}_f(t)$, will be different and typically it will be smaller. For example, when rate limiting a TCP connection, the achieved rate is usually slightly smaller than x_f. As we shall see in Section 4, when we do not have control over the sender (as in the case of traffic coming from the Internet), the rate limit is applied in the receiver. In that case, it will take a few RTTs for the sender to infer that there is congestion (i.e., our rate limiter) and reduce the rate. In those cases, the actual rate may be higher than $x_f(t)$. The use of the particular feedback control loop is justified in the following section.

3.2.3 Calculating prices

The price p_A of resource A represents a congestion level of the resource A. In fact, if such information is available (e.g., by reading queue statistics from a broadband modem or a wireless NIC), it could be plugged in directly in the calculation and the algorithm would work with no modifications. However, devices rarely support this capability and henceforth we will assume this functionality is not offered.

Instead, we model each resource using a so-called *virtual queue* strategy [9, 13]. This signals early warnings of congestion for resource A with capacity C_A by studying a queue with the same arrivals but reduced capacity βC_A, for some $\beta < 1$. For a virtual wired queue A with offered load $\sum_f x_f$, and virtual capacity of βC_A, $\beta < 1$, the virtual utilization is $\rho_A = \left(\sum_{f\in\mathcal{F}_A} \hat{x}_f\right)/(\beta C_A)$. The virtual utilization of the wireless virtual queue $\mathcal{W}$ is $\rho_{\mathcal{W}} = \frac{1}{\beta}\sum_{f\in\mathcal{F}_{\mathcal{W}},ij} \hat{x}_{f\ni ij} T_{ij}$, where the summation is over all wireless channels ij and all flows f that use wireless, and depends on the wireless link(s) that each flow uses, and the performance characteristics of each link (T_{ij}). The wireless utilization is derived from (8) in Section 3.4, where the characterization of achievable wireless regions is discussed in detail. Consequently, we use

$$p_A = \rho_A^B, \quad \rho_A \stackrel{\text{def}}{=} \begin{cases} \left(\sum_{f\in\mathcal{F}_A} \hat{x}_f\right)/(\beta C_A), & A \text{ is wired,} \\ \frac{1}{\beta}\sum_{f\in\mathcal{F}_A,ij} \hat{x}_{f\ni ij} T_{ij}, & A \text{ is wireless,} \end{cases} \quad (4)$$

where B is a small number (e.g., $B = 5$). For example, if we were to model resource A as an M/G/1 queue with utilization ρ_A, p_A is the probability that an arriving packet finds at least B packets in the queue. Every host simulates the operation of a *virtual queue* for every constraint resource, and learns the flow rates from other hosts.

We choose our estimate of capacity, $\hat{C_A}$ such that $E[\hat{C_A}] = \beta C_A$, to align with this price calculation, where for example $\beta = 0.95$. We simulate the virtual queue by using the observed rates $\hat{x}_f(t)$ of the flows $\mathcal{F}_A$ to determine the utilization ρ_A of the resource, and hence the marking probability p_A that a queue with the same average ρ_A would observe.[2]

We do not know the instantaneous load at the resource; instead, the hosts determine the utilization of the resources every M seconds (typically M=2 seconds in our implementation, see Section 4), which includes measuring the rates of the local connections and broadcasting them). It is then straightforward to show (using control theory techniques [21], [19], [18]) that the delayed feedback algorithm (3) is locally stable about the equilibrium point provided that κ is sufficiently small, i.e., provided that

$$\kappa \leq \min\left(\frac{1}{B+1}, \frac{M}{RTT}\right).$$

We omit the details here due to lack of space. Given our large observation window (M), we just need to pick $\kappa < 1/(B+1)$, and then we can show that the recursions are indeed stable.

3.3 Resource utilization

Our aim is to achieve a high target resource utilization, $\tilde{\rho}^A$, that satisfies

$$\tilde{\rho}_A = \alpha \frac{1-p_A}{p_A} = \alpha \frac{1-\tilde{\rho}_A^B}{\tilde{\rho}_A^B} \tag{5}$$

where α is a small scaling parameter that gives us flexibility in determining the desired rate. If we view our rate controller as a feedback control system, then the fixed point solution of (5) expresses the desired stable point for the system.

From a practical point of view, (5) expresses a quantifiable operational target as a function of configuration parameters B and α. For example, when $\alpha = 1$ and $B = 5$, the target utilization is $\tilde{\rho} \approx 0.88$. The higher the α is, the larger the queuing delays are which impacts user experience (see the discussion in Section 2.1). We find that $\tilde{\rho} = 0.88$ is a good compromise between efficiency and network responsiveness (see also Section 5.4).

3.4 Wireless capacity

As discussed in Section 3.2.3, in order to estimate the congestion prices for wireless hosts, we need to characterize the service rate of the wireless medium and calculate the feedback. We now provide a simple characterization of the throughput region for an access-point based 802.11 network. Our characterization captures additionally cases where the host association rates are not all equal. This is described via an approximate conservation law for the rate allocation region in the 802.11 network.

The model presented here is not intended to be yet another performance model of the 802.11 protocol. Rather, it is a simplified yet accurate model of the wireless resource. Its main contribution is in its simple parametrization and as such it can easily be incorporated in the controller design, without incurring large overhead.

[2]A refinement is to use an Adaptive Virtual Queue [13] which adapts $\hat{C}_A$ dynamically to increase utilization.

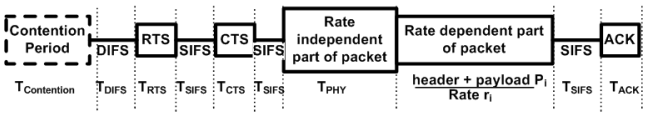

Figure 3: Typical timeline of a packet

Table 2: 802.11g parameter values used in the numerical results and experiments.

Parameter	Value
T_S (for Basic service)	9μs
T_{SIFS}	10μs
T_{DIFS}	28μs
T_{phy} (preamble and header, OFDM)	20μs
Control Frame rate	Rbs^{-1}
T_{ACK}	$T_{phy} + 112/R$
b_{MAC}	36 Bytes
CW_{min}	15
L	1500 Bytes

3.4.1 Packet Transmission Times

Host i transmits to host j using packets of average size L_{ij} bytes and has wireless association rate r_{ij} (either i or j is the AP). Consider the channel activity on the WLAN. The average transmission time T_{ij} has three components—(a) $T_{\mathcal{C}}$, the contention period when hosts are competing for the channel (including backoff times and time lost to collisions), (b) $T_{\mathcal{O}}$, the MAC protocol overhead, and (c) the MAC packet that contains the fixed MAC overhead of $b_{\rm MAC}$ bytes and a payload of L_{ij} octets that is transmitted at rate r_{ij}, if host i is successful. Fig. 3 illustrates these times, for the case of 802.11g with RTS/CTS enabled; it is easy to construct similar timelines for other variations of the protocol. (Unless explicitly stated, our discussion throughout the paper will be for 802.11g without RTS/CTS). $T_{\mathcal{C}}$, depends on the load (number of active hosts) and we will discuss its characterization in Section 3.4.3. The MAC protocol overhead consists of the time required to transmit the RTS, CTS and the ACK packets and also the interframe spacings; thus, $T_{\mathcal{O}} = T_{\rm DIFS} + T_{\rm SIFS} + T_{\rm ACK} + T_{\rm PHY}$ (an additional $T_{\rm RTS} + T_{\rm CTS} + 2\,T_{\rm SIFS}$ is required when RTS/CTS is used). The average packet transmission time is $(b_{\rm MAC} + L_{ij})/r_{ij}$. Hence, the average transmission time T_{ij} is

$$T_{ij} = T_{\mathcal{C}} + T_{\mathcal{O}} + (b_{\rm MAC} + L_{ij})/r_{ij}. \tag{6}$$

The values of 802.11g parameters used in the controller are given in Table 2.

3.4.2 Sharing the Medium

Let $x_{f \ni ij}$ be the rate of flow f, i.e. $x_{f \ni ij} = x_f$, iff f uses the wireless link from host i to host j, and 0 otherwise. For example, if i is a wireless node and j is the wireless access point, then $x_{f \ni ij} = x_f$ for all flows f that leave i and are destined to a node on the wired network or to the Internet. Similarly, $x_{f \ni ji} = x_f$ for traffic destined to i coming from the wired network or the Internet. If k is another wireless node and there are some flows from i to k, then $x_{f \ni ij} = x_f$ and $x_{f \ni jk} = x_f$ for each such flow f (assuming the typical case that the traffic is relayed by the access point j). Observe that the flows of the last example use the wireless network twice. We want to characterize the "maximal" set of $\{x_f\}$ that can be achieved, conditioned on the capacity of the wireless channel, and the characteristics and usage of the wireless links ij.

For each vector of *feasible* rates $\{x_f\}$ (measured in MAC frames per second) when the system is stable, the probability that a packet being scheduled at the wireless medium is from flow f at node

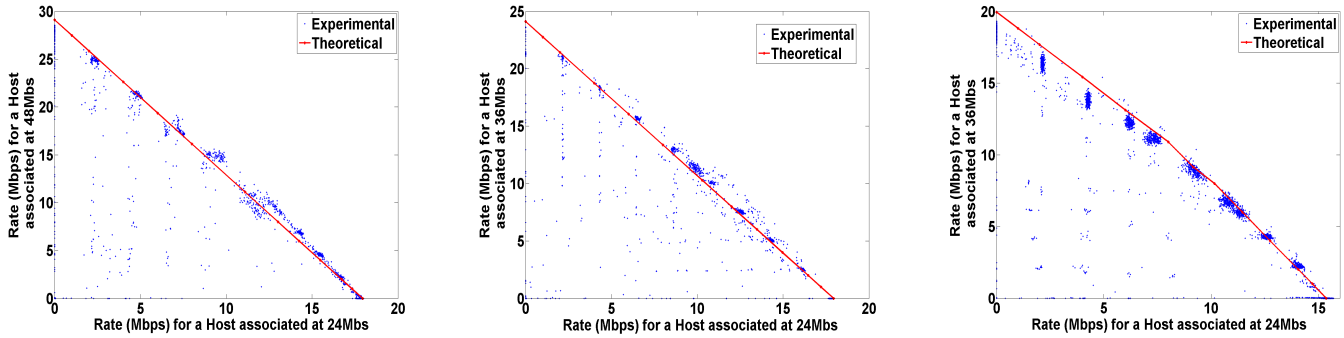

Figure 4: Conservation law in practice: Achieved throughputs (averaged over 2sec) of the two hosts transmitting simultaneously, and each of them is rate limited at a time with 0, 2, 6, 8 Mbps, and with no rate limits. In the left and middle figures two hosts send UDP traffic to a wireline host. In the right figure, the hosts send TCP traffic to a wireline host. For TCP, the conservation law does not result in an exact straight line due to the association rate adaptation performed for the packets (ACKs) sent by the AP.

i is $x_{f\ni ij}/\sum_{f',i'j'} x_{f'\ni i'j'}$. Note that this holds *regardless* of the scheduling policy among wireless nodes, or the policy used at each individual node because the network is not saturated, hence all packets will be served.

The average time between two consecutive transmissions is

$$T_{\mathcal{R}} = \sum_{f,ij} (x_{f\ni ij} T_{ij}) / \sum_{f,ij} x_{f\ni ij}, \tag{7}$$

and the service rate of the wireless network is $\mu = 1/T_{\mathcal{R}}$. Observe that in sharp contrast to the wired networks, the service rate of the wireless network depends on the offered load, assuming that the wireless links ij differ in performance (i.e. not all T_{ij} are equal). (7) suggests, for example, that the service rate is inversely proportional to the rate carried by the slow wireless link (those with large T_{ij}) [10]. The network is stable if $\sum_{f,ij} x_{f\ni ij} < \mu$ (the total load is smaller than the service rate) hence the feasible rate region is characterized by

$$\sum_{f,ij} x_{f\ni ij} T_{ij} < 1, \tag{8}$$

where T_{ij} is defined in (6). The left hand side of (8), $\rho_W = \sum_{f,ij} x_{f\ni ij} T_{ij}$, is the utilization of the wireless medium. Note that (8) includes the case when there is cross-traffic among the wireless nodes.

Equation (8) was derived under the assumption that the rates x_f are given in MAC frames per second. It can applied to the rates given in bytes per second under either of the following conditions: (a) the packets are of similar sizes, i.e. $L_{ij} \approx L$, or (b) the transmission overhead is substantial smaller than the packet transmission time. We shall assume that both conditions are met.

In practice, the left part of (8) should be bounded away from 1 to protect against traffic fluctuations that may lead to saturation. (Recall that during saturation the rate allocations depend also on the MAC protocol and are not strictly determined by the flow weights.) We protect against such cases by scaling the utilization by a factor $1/\beta$ as in (4) (we use $\beta = 0.95$).

We have verified (8) in a simple setting with 802.11g hosts using backlogged traffic to a wireline host. Fig. 4 presents such an example for an experiment with two wireless hosts for visualization purposes. We have experimented with both UDP and TCP. In the experiment, a rate limit was applied to one of the hosts, with the other host not being rate limited (i.e., we controlled x_f), and vice versa. The figure shows scatter plots of the throughputs that the two hosts achieved against each other, for experiments that lasted 5 minutes for each of the rate limits (points show averages over 2 seconds). The rate predicted by (8) is also shown; observe good agreement with experimental data. We have verified that (8) is in agreement with experimental data for experiments using up to 4 wireless hosts of varying association rates.

3.4.3 *Accounting for Collisions*

The analysis above ignores the effects of MAC-level packet collisions. We now obtain the collision overhead when there are n active hosts. Since the objective is to obtain a simple characterization for use in (4), we make a simplifying assumption, that the packets experience very few collisions. This is reasonable because of the small number of hosts in the network and also because $CW_{\min}$, the minimum backoff window, is quite large and hence minimizes collisions.

As in most models, e.g., [5], we assume that a transmission attempt results in a collision independent of the other attempts. Let $\gamma(n)$ be the probability of a collision, given a node transmits. It can be shown that

$$\gamma(n) \approx n\zeta\delta, \tag{9}$$

where ζ is the transmission attempt rate and δ is the slot length. Of course, ζ depends on $\gamma(n)$ and we have a fixed point equation. (See Chapter 7, [12]). Using the approach of [5, 12], we can obtain the following first-order approximation for $\gamma(n)$.

$$\begin{aligned}\gamma(n) &= \frac{1}{2(CW_{min}-2)}\,(4n + CW_{min} - 5 \\ &\quad - \sqrt{CW_{min}(CW_{min}-2) + (4n-3)^2}\Big)\end{aligned} \tag{10}$$

In Fig. 5, we compare the value of $\gamma(n)$ as obtained from (10), from an exact computation of (9). The goodness of the approximation is evident. We also show $\gamma(n)$ obtained from experimental data for $2 \le n \le 4$. Once again, (10) seems to fit experimental data reasonably accurately.

On average, there are $1/(1-\gamma(n)) - 1 = \gamma(n)/(1-\gamma(n))$ collisions and $1/(1-\gamma(n))$ idle slots between successful transmissions for host j. An average idle time between two transmission attempts (successful or not) is $\delta/\gamma(n)$ Therefore, we have

$$T_{\mathcal{C}} = \frac{\delta}{(1-\gamma(n))\gamma(n)} + \frac{\gamma(n)}{1-\gamma(n)} T_{coll}, \tag{11}$$

where T_{coll} is the average duration of a collision and it depends on the usage or otherwise of RTS/CTS. If RTS/CTS is used, then

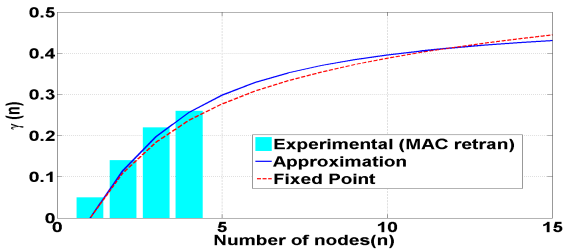

Figure 5: Probability of collision $\gamma(n)$ as a function of the number of hosts n. For $n = 1$ there are no collision errors, only channel losses, which explains the deviation from the model (of about 5%). For $n > 1$ the collisions dominate and the model fits well.

$T_{coll} = T_{RTS} + T_{SIFS} + T_{DIFS} + T_{ACK}$. When RTS/CTS is not used, the exact expression is tedious (since it involves accounting for the duration of the longest of the colliding packets), and we use the following conservative approximation

$$T_{coll} = \max_{ij} \frac{b_{\text{MAC}} + L_{ij}}{r_{ij}}.$$

Another source of a non negligible loss of capacity is the PHY errors. It is known [16] that these errors can occasionally be substantial. We believe it is rather hard to capture PHY errors in a simple model. Further, the hosts react to PHY losses in a rather complex manner which also defies simple characterization. Nevertheless, (8) is reasonably accurate and the approximate linear relationship is very useful in defining control algorithms and developing optimization techniques.

3.5 The rate control algorithm

Summarizing the discussion over the whole section, we provide here a sequence of steps for resource allocation in wired-wireless networks. This sequence of steps is the exact control loop implemented in our system at the end hosts (discussed in Section 4), and can be directly used as a guide for a network practitioner.

1. Determine the weights w_i per application, flow or host.
2. Estimate the current operating point of the WLAN (See Section 3.4).
3. Estimate the current utilizations $\hat{f}_i(t)$ of the flows (or applications) under consideration, the utilization ρ_A of each resource A by aggregating the relevant $\hat{f}_i(t)$, and hence the marking probability p from (4) (See Section 3.2.3).
4. Update the rate caps per application by following (3) (See Section 3.2.2). Then, go to step 2 after M seconds, with $M \approx 1s - 2s$.

4. SYSTEM IMPLEMENTATION

Over the previous sections, we have presented a set of algorithms to enable rate control in small networks comprising wireless and wired hosts. We now discuss how we implemented this framework.

Our framework comprises two basic components: i) the inference of the wireless capacity across all wireless hosts in $\mathcal{N}_W$, and ii) the estimation of the prices and the enforcement of rate limits across all hosts. For both of these components, local host information such as the wireless association rates r_{ij} and the instantaneous rates of the controlled applications x_f, is required. This gives us two possible design choices. The first choice is of a centralized approach where all hosts transmit the relevant information to a dedicated host or device that implements the algorithms, performs all estimations, computes the rate limits to be imposed, and announces the limits to the hosts (which will implement them). Second, a decentralized approach, where hosts share the necessary information and independently perform all computations.

We believe that a decentralized architecture better meets the design goals outlined in Section 3. First, a decentralized approach is incrementally deployable through a light-weight software agent at the hosts; no changes are required to applications, protocols (e.g., MAC, TCP) or devices such as switches and routers which are generally low-cost in small networks. Second, sharing of information only bears a minimal overhead due to the small number of hosts. Third, as pointed out throughout the paper, hosts are better suited to associate traffic information to applications and users.

We implemented the entire functionality of coordination and control at the hosts. Each host periodically (every second in our implementation) polls the operating system for application specific performance information like connection, and network interface statistics. Then, the necessary information for our algorithms, such as application rates x_f, weights w_f, average packet size L_{ij}, and the association rates r_{ij}, is broadcasted to all other nodes in the network. This broadcast communication requires an efficient and reliable communication channel, with modest capacity for control traffic, for timely delivery of the information. We have used reliable multicasting (PGM [1]) to broadcast this information, with an overhead of roughly 250 bytes per second per host for this control traffic. Other overheads such as memory consumption (both code and data), and CPU utilization, are typically less than 30MB, and 2% respectively, on our low-end 2GHz/1.5GB RAM laptop.

We have used a token-bucket approach for rate limiting and shaping. We set the token rate for each connection to be equal to the rate determined by the simulation of the virtual queue. The implementation of rate limiting for outgoing traffic is based on the Windows OS traffic control interface [3]. For incoming traffic from the Internet, where we cannot assume control over the sender, we rate limit at the receiver, and expect that higher-level congestion controllers (such as TCP) will react by rate adaptation. We have used a custom made rate limiter for such incoming traffic.

Observe that our approach assumes that users trust each other. Indeed, we assume that all hosts connected to the network implement our system, that they broadcast their rates and weights truthfully, and that they rate limit their local connections based on the output of the algorithm described in Section 3. Moreover, we assume that the users have agreed on how to set the weights of their traffic. Dealing with malicious or misbehaving hosts is outside the scope of this work.

We have implemented the prototype for the Windows Vista operating system, using interfaces that are available to user processes (with administrative privileges). In principle, the implementation does not require any changes to the operating system, and does not involve installing specialized device drivers.[3] Similar interfaces are available to other modern operating systems, and are typically enabled by default. Hence, our implementation can be easily ported to other operating systems, and requires zero or very little changes to the core of the operating system.

[3]This is true for the various Linux variants. In Windows, the traffic control interface does not implement rate limiting of incoming traffic. We control incoming traffic using our own rate controller, which was implemented as a network driver. We still use the standard traffic control interfaces to rate limit outgoing traffic.

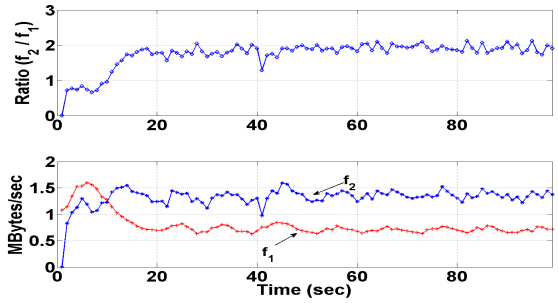

Figure 6: Prioritizing wireless flows in scenario 1. The top figure presents the ratio achieved between the two flows, while the bottom shows the individual flow rates.

5. EVALUATION

We now present an evaluation of the proposed rate allocation framework implemented as described in the previous section. Our evaluation will cover our motivation scenarios outlined in Section 2.2. The experiments were performed in our testbed which is similar to the simple network presented in Fig. 2. In particular, two wireless hosts connect through 802.11g to a Belkin access point. A DSL modem/router provides connectivity to the Internet and DHCP service. The nominal DSL access speeds are 8Mbps downstream and 512 upstream. Finally, two wired hosts are connected directly to the Ethernet ports of the local router. Unless otherwise specified, the experiments refer to TCP flows generated with *iperf*.[4] TCP is in general harder to control due to the ACK feedback loop. UDP results look similar (except the higher observed total capacity for wireless) and have been omitted due to space limitations.[5]

5.1 Scenario 1: Prioritizing wireless flows.

We first examine the scenario described by Fig. 1(left), where two flows (f_1 and f_2) compete in the WLAN; f_2 is downstream towards the wireless host, while f_1 upstream sourced at the second wireless host. The goal here is to provide higher capacity to one of the wireless flows. As discussed previously, this is currently not feasible without modifying the MAC.

We configure the weights per flow to be $w_1 = 1$, and $w_2 = 2$, implying that our desired policy specifies that f_2 should experience twice the rate of f_1. Fig. 6 presents the result of this simple exercise when the controller is enabled and f_2 starts one second after f_1. The top figure highlights the achieved ratio between the two flows, while the bottom the individual flow rates. Our controller succeeds in allocating rates according to the desired weights, with the average flow ratio over time equal to 1.9. The controller takes roughly 10 seconds to converge, and appears stable despite the variability individual flows may experience due to TCP or wireless channel effects.

5.2 Scenario 2: Improving network performance

Scenario 2 examines the case where channel quality is different among hosts which is not uncommon in residential networks [16]. In such cases, the overall network performance may be degraded (cf. [10]) as shown in Fig. 7. The figure highlights the setup presented in Fig. 1(middle), where two wireless flows (f_1 and f_2) are sourced at hosts with different association rates. When f_2 becomes

[4] `http://sourceforge.net/projects/iperf/`

[5] Note that controlling UDP traffic can only be achieved at the sender side.

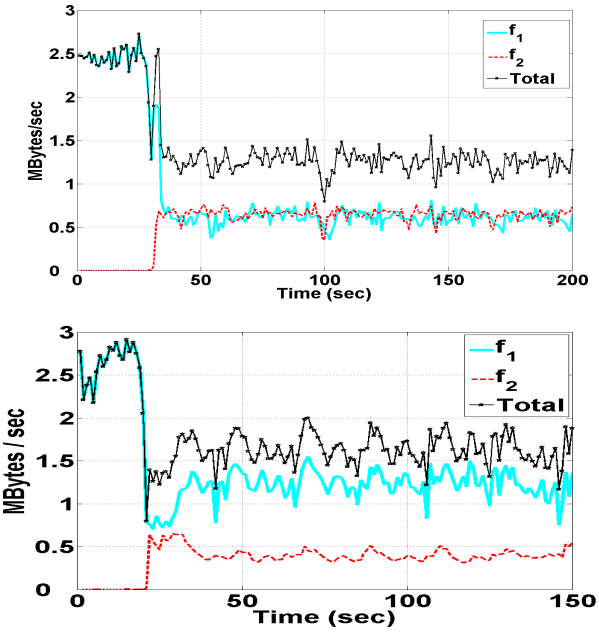

Figure 7: Prioritizing wireless flows in scenario 2. Top: Uncontrolled case. Bottom: Controlled case.

active roughly at 30 seconds (Fig. 7, top), the total capacity of the WLAN experiences a drop of roughly 55% due to the small association rate of the wireless host (12 Mbps).

Applying rate control can increase overall network performance by giving higher priority to hosts that experience good channel quality. For example, Fig. 7 (bottom), presents a scenario where we configure the weights per flow to be $w_1 = 3$, and $w_2 = 1$, effectively allocating the medium with a bias towards the host associated at 54 Mbps. This results in increasing the total WLAN capacity by roughly 20% compared to the uncontrolled case, and the rate limited host achieving a rate of 410KBps (600KBps in the uncontrolled case). Depending on the scenario and the extent to which rate-limiting the slow host is desirable, the total capacity of the network will increase as the bias towards the faster hosts increases. Overall, the gain in capacity after applying the controller can be estimated by the analysis in Section 3.4, and will be roughly equal to

$$1 - \frac{(1+l)(w_1+w_2)}{2(w_1+w_2 l)},$$

where $\frac{1}{l}$ reflects the percentage of capacity lost when slow hosts are introduced in the WLAN. In the particular example, where $\frac{1}{l} \approx 0.55$, the estimated theoretical gain is 0.23 or 23%. Consequently, we are able to mitigate the network performance loss due to the presence of slow nodes without modifying the MAC. It is also possible to increase the rate of the slow node and, consequently, reduce overall network performance by increasing the weight of the slow node. Hence, it is possible to implement other wireless resource allocation objectives without changing the MAC.

5.3 Improving network responsiveness

When a network resource is highly utilized, then its associated queue grows in size. Applications using the resource will then experience high latencies. Here, we reproduce such a scenario similar to the one depicted in Fig. 1(right). The two flows, f_1 and f_2, represent traffic from high-quality video streaming from YouTube and

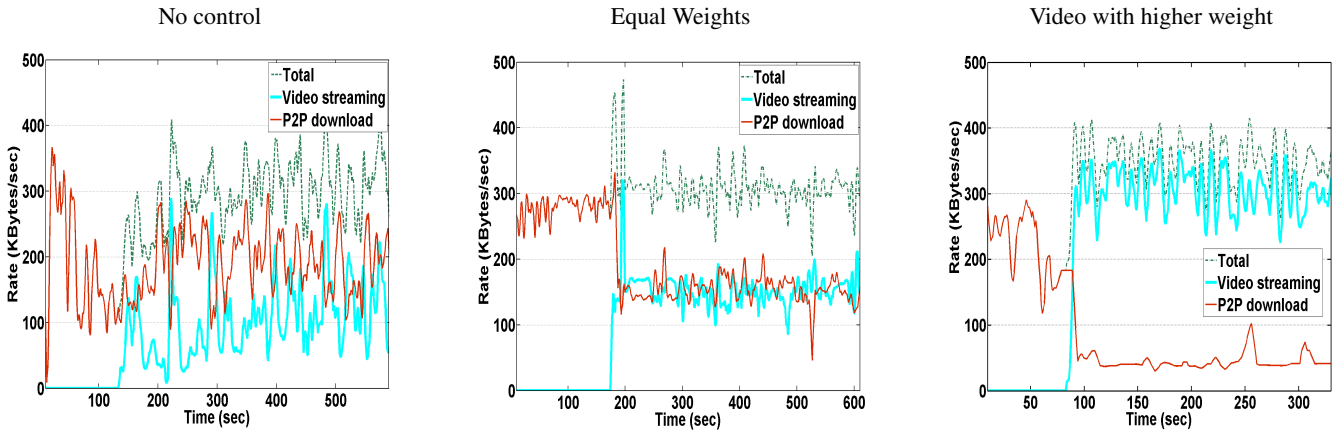

Figure 8: Experiments with both P2P downloading and video streaming. Left: Uncontrolled traffic, video streaming rate is lower than p2p downloading and the video frequently pauses. Center: P2P and streaming get the same rate, however there is still not enough rate for video streaming and the video pauses frequently. Right: Streaming is configured with a weight 6x of the p2p weight; rates are allocated proportionally and the streaming rate is high enough to achieve smooth playback without pauses. (Rates smoothed for 5sec to improve readability.)

a peer-to-peer (p2p) application respectively. We will introduce f_3 later in this section. To estimate the queuing delay, we were continuously pinging the first upstream broadband router.

Fig. 8(left) depicts the rates of the p2p and the video streaming traffic, when no control is enforced. The p2p traffic captures most of the broadband capacity, resulting in not enough capacity for video playback, causing frequent pauses. Measured ping times averaged 243ms, and often were in excess of 500ms. Ping times without traffic are in the order of 30ms.

Fig. 8(center) depicts the controlled case when equal weights were assigned to each application. As expected, when both applications were active, their rates were similar (151KB/s and 157KB/s). Unlike the uncontrolled case, the ping times now were significantly reduced, as we have reduced the network queuing; our controller allows just enough traffic so that network queues do not grow but the network is utilized at the same time. The average ping time was roughly 55ms, and less than 3% of the measurements were greater than 150ms. However, even in this case, the video playback freezes frequently not receiving enough throughput.

To provide a better experience for video streaming, we reduced the weight of p2p to be 6 times lower than the weight of video streaming. Fig. 8(right) indicates this case, for which the video playback did not suffer from pauses. The average ping time was also low, around 68ms. This means that other (low rate) network applications, such as web browsing and emailing, would also perceive a responsive network. The choice of 6 as the ratio of the weights was arbitrary in order to guarantee that there is enough capacity for the video. In practice, either the user or some other algorithm should determine an appropriate ratio (and weights) in order to guarantee the desired performance, by taking into account the capacity of the network resources and the demands of the applications. Determining the weight ratios is out of the scope of this work.

Compared to the first two scenarios examined, this is a more challenging scenario as p2p applications open a large number of TCP connections, most of which with a small rate. Indeed, during these experiments, the number of connections included in f_1 was larger than 200. Even in such an extreme scenario, the controller manages to achieve the desired weighted allocation, and provide a stable performance.

Consider now a similar case, with f_1 and f_2 being upstream flows, and introducing the internal flow f_3. This can further have a pronounced impact on the network as Fig. 9 denotes. f_3 creates significant queuing in the WLAN, which delays the ACKs of f_1, and eventually connection f_1 drops. Enabling the controller "reserves" capacity for the ACK stream of flow f_1.

5.4 Virtual queue and utilization

The price for controlling the queuing delay at the constrained resource is a reduction in its utilization. To evaluate this reduction, we have performed the following controlled experiment. We emulate a link of 100kBytes/sec with three long term TCP transfers (from three different hosts). Table 3 gives the *application rate* of each of flow for various configurations of the rate controller (see Section 3.3). Table 3 shows that the effective rate (i.e., *Total*) reduced from 76.83kB/s (uncontrolled) to 66.65kB/s – 75.46kB/s, depending on the controller. Hence, the penalty for rate controlling can be made quite small.

Moreover, rate control brings advantages with respect to fairness and efficient use of resources. Examining the ratio of bytes sent (goodput) over the total bytes (G/T), we observe that rate control reduces the overhead by 10% (see goodput ratio G/T). In other words, rate control reduced the amount of capacity spent in retransmissions. This was the result of preventing saturation at the bottleneck link. By queuing packets at the end-hosts (and before the TCP layer), we implicitly signal congestion to the TCP endpoints before congestion builds up in the network and packets are dropped. Recall also that controlling the queues at the hosts reduces queuing delays, and that, in general, benefits congestion control algorithm, like TCP. Observe in Table 3 that in the uncontrolled case, connection A received a higher rate than the other two connections. This unfairness was common in this particular scenario (though not universal, i.e., in some cases TCP allocated the capacity fairly). We never observed such unfairness with rate control.

The ratio T/C is the network utilization ρ of our virtual queue. The measured ρ is close to the theoretical values predicted by (5). For example, optimal $\rho = 0.881$ for $B = 5$ and $\alpha = 1$; the observed utilization was 0.867. A larger B increases utilization at the cost of responsiveness. Overall, the particular configuration choices of the virtual queue size B were not very important (with the exception of $B = 3$ that is a bad choice even theoretically). Such robustness is attractive from an engineering point of view, since it negates the need for fine-tuning.

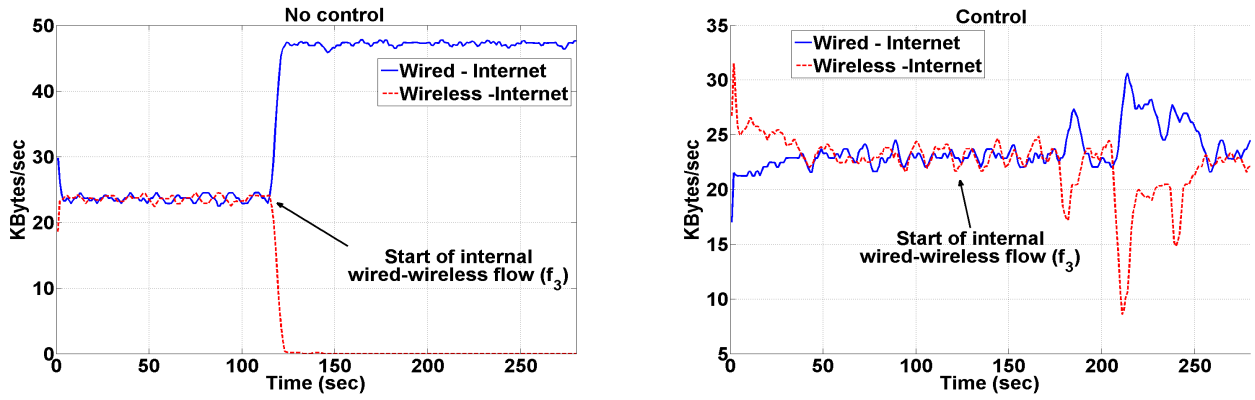

Figure 9: Uncontrolled and controlled cases for the scenario in Fig. 1(right), with two upstream flows to the Internet (f_1 and f_2), and an internal wired-to-wireless flow (f_3) If the f_3 is not rate-limited, f_1 drops because the wireless is saturated.

Table 3: Performance of three web transfers (of equal priority) [in kBytes/sec] and network utilization

Controller	Connection 1	2	3	Total	G/T	Gdput	T/C
Uncontrolled	43.6	15.7	17.4	76.83	0.78	59.9	0.97
$B=3\,\alpha=1$	26.6	22.3	21.6	66.65	0.84	56	0.79
$B=3\,\alpha=2$	24.	23.1	22.8	70.02	0.84	58.8	0.83
$B=5\,\alpha=1$	25.1	23.7	26.6	75.46	0.84	63.4	0.87
$B=5\,\alpha=2$	25.6	24.1	24.2	74.01	0.84	62.2	0.88
$B=10\,\alpha=1$	25.8	24.8	24.4	75.08	0.83	62.3	0.90
$B=10\,\alpha=2$	26.2	24.4	24.9	75.64	0.83	62.7	0.91

G/T is the ratio of unique bytes transferred (goodput) over total transfers (including retransmissions). T/C is the ratio of total transfers over (our estimate of) the capacity of the uplink (100kBytes/sec).

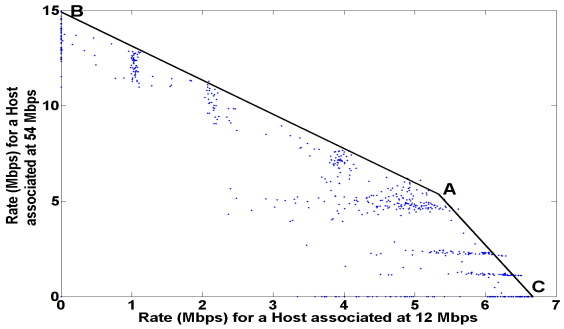

Figure 10: Prioritizing wireless flows in scenario 2. There are two controlled flows, flow f_1 with association rate 54 Mbps and flow f_2 with 12 Mbps. There is an additional uncontrolled (interfering) flow f_3 whose rate is not plotted.

6. DISCUSSION

We have presented a framework for traffic management and resource allocation in small networks. However, our approach leaves some issues for future work; we outline here the most important of these.

Non-compliant devices. So far, we have operated under the assumption that all hosts participate in the inference and control phases of our algorithms. If this is not the case, we can still infer the equivalent capacity, both for wired and wireless resources; yet, to control non-compliant or partially compliant devices we need to partner with an intermediate device, such as a middle box, AP or switch. For the wired case, our algorithm produces estimates of the controlled capacity $\hat{C}$ (from compliant devices), while standard bandwidth estimation techniques, such packet-pairs or probe-based, can estimate the total capacity in certain scenarios. For the wireless, our estimated capacity will be less that the nominal capacity of the channel: from Section 3.4, we will estimate a reduced capacity for the WLAN, and the difference between this and the nominal capacity is approximately equal to the weighted sum of the rates from the non-compliant devices.

Wireless characteristics. Our model currently does not account for interference, but it can be extended to do so. Consider the example from Fig. 10, which is similar to Fig. 4 in Section 3.4. The difference here is that we have three flows, one of which is an interfering host. Point A corresponds to the case when all three flows behave as unconstrained. Point B corresponds to the case when f_2 and f_3 are unconstrained and $f_1 = 0$. It is easy to verify analytically that the region between A and B is linear, as shown by the experiment (similarly between A and C). Hence, in presence of the interference the rate region is piece-wise linear, and the parametrization is only slightly more complex. We leave the full analysis here for future work.

In practice, interference will manifest itself as an increase in lost packets. We can measure this as a drop in capacity, and hence perform our allocation on the revised capacity automatically. Additionally, we have not discussed association rate adaptation. Our system continuously monitors the association rates and adapts the optimization problem and the associated rate limits. In the experiments, the controller had time to adapt between the changes in the association rates.

Network size. Our solution targets small networks, with at most 10s of devices (and could also extend to networks with a few APs). This is already a large market segment, typified by low-end hardware with limited functionality. For a more complex wireless network, interaction between different subnets/APs may make our conservation laws less appropriate. Examining the efficiency of the proposed algorithm in more complex scenarios is out of the scope of this work.

Setting the flow weights. Setting the weights w_i is a challenging and an "orthogonal" problem that should take into account user priorities and intentions that are very difficult to infer even at the host-level, since the weights of a given application can be context dependent. One way to set initial weights is to set $w_f = Mx_f^{target}$, where x_f^{target} is a nominal target rate for the connection, and $M = E[p/(1-p)]$, related to the target utilization via (5). We do not elaborate the problem further here.

7. RELATED WORK

Traffic management and resource allocation has largely been ignored for small networks. Several management solutions have been proposed for large-scale enterprise networks [6, 22, 17]. These studies attempt to impose pre-defined detailed policies on the network; policies are enforced by network equipment through a centralized control. Our work targets much simpler networks without dedicated administrative entities, and our goal is to improve performance rather than minimize configuration or security errors.

A small number of studies [16, 7] gave some insight into the characteristics of residential and small office networks. Specifically, Papagiannaki et al. [16] examine wireless characteristics of home networks and show how simple changes in the configuration of the network could result in a big impact on performance. Similarly, Dischinger et al. [7] show significant variation in the capacity of the broadband access link offered to residential users. Our rate controller takes into account these variations by adapting to the offered capacity and traffic demands.

Our approach resonates with the Congestion Manager (CM) approach of Balakrishnan et al. [2], particularly in terms of philosophy which puts the application in control, accommodates traffic and application heterogeneity and adapts to traffic dynamics. However, unlike the CM approach, our system does not need to probe periodically to infer network characteristics, neither do we use a congestion window-based control. Additionally, our controller is especially designed to take account of wireless networks and their idiosyncrasies. Our approach also shares some of the goals of the wireless distributed fair scheduling by Vaidya et al. [20], which implements a distributed fair queuing that achieves proportionally fair allocations. However, the authors there describe a MAC implementation, and assume that all association rates are the same. In contrast, we provide a general solution that requires no modification to underlying protocols and is implemented at the application layer.

Our model of the wireless resource is based on existing analytical models (see [5] and follow-up works). However, to the best of our knowledge no other wireless model gives a simple parametrization of the rate region of an unsaturated wireless network.

8. CONCLUDING REMARKS

We have presented a practical resource allocation framework for small networks that contain both wired and wireless hosts. The controller jointly optimizes over multiple wired and wired resources. To enable allocations in wireless networks, we have derived a conservation law for wireless networks, which is both simple and effective, and a good approximation to experimental data. We have further demonstrated the feasibility of application layer control by building and experimentally validating the framework in user space. We believe that the proposed framework can significantly augment the available traffic management functionality and as a result the user experience. This was demonstrated through sample scenarios that cannot be supported in today's networks due to protocol or device restrictions.

9. REFERENCES

[1] RFC 3208 - PGM Reliable Transport Protocol Specification. `http://www.faqs.org/rfcs/rfc3208.html`.

[2] H. Balakrishnan, H. S. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *SIGCOMM*, 1999.

[3] Y. Bernet. *Networking Quality of Service and Windows Operating System*. New Riders, 2001.

[4] D. Bertsekas and G. R. *Data Networks*. Prentice-Hall, 1987.

[5] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. In *IEEE J. Sel. Areas Commun.*, 2000.

[6] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *ACM SIGCOMM*, 2007.

[7] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *IMC, AM SIGCOMM Internet Measurement Conference*, 2007.

[8] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and capacity estimation. *IEEE/ACM Transactions on Networking*, 12(6):963–977, December 2004.

[9] R. J. Gibbens and F. P. Kelly. Resource Pricing and the Evolution of Congestion Control. *Automatica*, 35:1969–1985, 1999.

[10] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *INFOCOM*, 2003.

[11] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 1998.

[12] A. Kumar, D. Manjunath, and J. Kuri. *Wireless Networking*. Morgan-Kaufmann (an imprint of Elsevier), USA, 2008.

[13] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *ACM SIGCOMM*, 2001.

[14] S. Machiraju, D. Veitch, F. Baccelli, and J. Bolot. Adding definition to active probing. *ACM SIGCOMM Computer Communication Review*, 37(2):17–28, April 2007.

[15] Office of Communications UK (OfCom). Consumer experience of broadband performance: initial findings, 2009. `http://www.ofcom.org.uk/research/telecoms/reports/bbspeed_jan09/`.

[16] K. Papagiannaki, M. Yarvis, and W. S. Conner. Experimental characterisation of home wireless networks and design implications. In *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[17] J. Rexford, A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang. Network-wide Decision Making: Toward a Wafer-thin Control Plane. In *HotNets–III*, 2004.

[18] S. L. Song Chong and S. Kang. A Simple, Scalable and Stable Explicit Rate Allocation Algorithm for MAX-MIN Flow Control with Minimum Rate Guarantee. *IEEE/ACM Trans. on Networking*, 9(3):322–335, 2001.

[19] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkausser, 2004.

[20] N. Vaidya, A. Dugar, S. Gupta, and P. Bahl. Distributed fair scheduling in a wireless LAN. *IEEE Trans. Mobile Computing*, 4(6):616–629, 2005.

[21] G. Vinnicombe. On the stability of networks operating TCP-like congestion control. In *Proc. IFAC World Congress*, Spain, 2002.

[22] H. Yan, D. A. Maltz, T. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4D network control plane. In *NSDI*, 2007.