DYNAMIC CHANNEL-AWARE BANDWIDTH MANAGEMENT IN IEEE 802.11 NETWORKS

BY

SAMARTH HARISH SHAH

B.E., University of Madras, India, 1998

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

Abstract

One of the major requirements to support multimedia data in a dynamic environment such as a IEEE 802.11 wireless network is to provide the application minimum throughput guarantees. In this work, we present a cross-layer architecture that enables the provision of rate guarantees to multimedia applications, while still catering to best-effort traffic. We perform centralized arbitration of the shared bandwidth resource, and attempt to understand how this centralized architecture scales to different 802.11 wireless topologies. The major components of our architecture are a channel quality monitor that continuously estimates the channel capacity, a central bandwidth manager (BM) that implements a channel allocation policy to distribute channel time to competing flows, and a rate-control mechanism that adjusts the traffic being injected into the network by a flow in accordance with its allocated fraction of the channel. The channel quality monitor is co-located with the IEEE 802.11 MAC protocol, but does not alter it in any way. It detects changes in medium contention and in channel fading errors as perceived by each network flow, since these phenomena are both time- and location-dependent. We assume adaptive multimedia applications that can function at multiple qualities and can hence work within a range of allocated bandwidths, above the minimum requirement. The bandwidth requirements of the application are translated into channel time requirements, taking into account the quality of the channel. The channel time requirement indicates the fraction of unit time the application needs to actively transmit its data in order to satisfy its throughput requirements. We present several policies for channel allocation by the central BM, in this work: fair, price-based, and utility-based. All our channel allocation policies endeavor to provide minimum throughput guarantees to the network flows. The price-based policy mimics a distributed auction of channel time centrally at the BM. In the absence of wireless fair scheduling at the MAC-layer in current IEEE 802.11 products, we propose rate-control at the higher layers of the OSI protocol stack to control the quantity of traffic entering the network. We implement a leaky-bucket rate-controller at the network interface queue that is continuously updated with the flow's allotted rate. We perform simulation and testbed experiments to evaluate the performance and demonstrate

the feasibility of our overall architecture. Our experiments show that we incur low protocol overhead in providing statistical throughput guarantees to all network flows. Moreover, we find that our basic centralized architecture is flexible enough to work more-or-less unchanged with different IEEE 802.11 wireless network topologies.

Table of Contents

Chapter 1

Introduction

1.1 IEEE 802.11 Network Topologies

Wireless networks employing the IEEE 802.11 protocol are becoming increasingly popular today. Currently, the mostly widely deployed IEEE 802.11-based wireless network is the "hot-spot" network, which is a single-hop wireless local area network (LAN) with an *access point* (AP) connected to the wired Internet. However, the IEEE 802.11 protocol is being considered for medium access control (MAC) and physical layer functions in other types of wireless topologies also. A lot of research is concentrated on mobile, multi-hop, infrastructure-less, wireless networks called *mobile ad-hoc networks* (MANETs) for applications in defense and disaster-relief. Static multi-hop network topologies utilizing the IEEE 802.11 protocol are useful in sensor networks. In addition, single-hop, infrastructure-less, peer-to-peer wireless networks are being studied for use in smart homes.

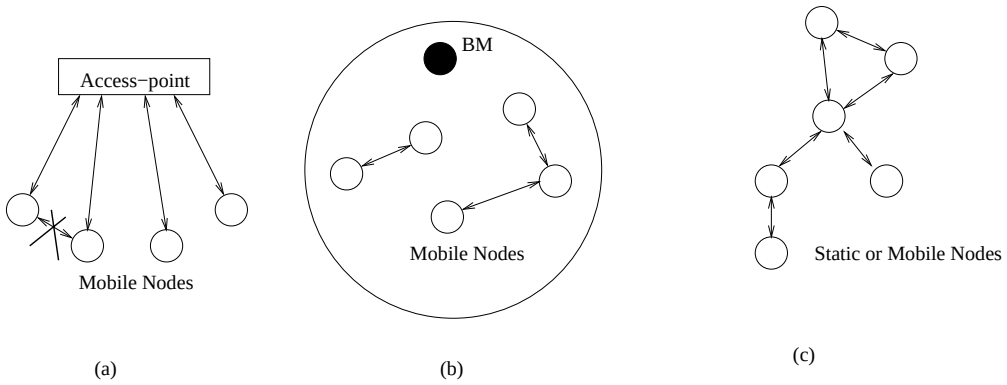


Figure 1.1: Types of IEEE 802.11-based wireless networks.

Figure 1.1 illustrates the three types of networks described above: (a) AP-based wireless LANs, (b)

single-hop peer-to-peer wireless LANs, and (c) multi-hop networks. In the two single-hop topologies, all nodes lie within each other's transmission range so that only a single host can be transmitting at a time in the network. No routing is necessary. In MANETs and static multi-hop networks, spatial reuse of the channel is possible and routing becomes important. The difference between the two single-hop network topologies in Figure 1.1(a) and Figure 1.1(b) is that peer-to-peer transmissions are not allowed in the former whereas they are possible in the latter. All communication in AP-based wireless LANs is "uplink" and "downlink" between the individual mobile hosts and the AP. Typically, AP-based wireless LANs (i.e., hot-spot networks) can be "multi-cell," i.e., consisting of multiple wireless subnets, each connected to the wireline backbone through a different AP. Such a topology is used to extend the coverage area of the wireless hot-spot. A mobile user can roam from one cell or subnet to another, using the AP of its current cell for connectivity to the Internet. Cells or subnets may overlap, i.e., transmissions within one cell may interfere with those in another [1]. A mobile host within range of more than one AP picks the one whose beacons have the best signal strength. The AP-based wireless LAN topology is a special case of the peer-to-peer wireless LAN topology, in which peer-to-peer transmissions are prohibited.

In this work, we address bandwidth management issues in both single-hop topologies, in multi-cell topologies, and in static multi-hop topologies. We illustrate how channel effects vary in these topologies, how these variations affect traffic, and how the problems can be alleviated.

1.2 Multimedia Data and the IEEE 802.11 Protocol: Challenges

In recent times, much effort has gone into solving the problem of transmitting multimedia data over IEEE 802.11 wireless networks, mostly in the area of rate-control of flows using MAC-layer fair scheduling techniques (e.g. [1]). Three mutually orthogonal factors make the problem challenging: (a) QoS requirements of multimedia applications, (b) bursty nature of some multimedia traffic due to compression, and (c) unreliable and dynamic nature of the wireless network. In this section, we describe in detail the challenges faced in bandwidth management in wireless networks.

1. **Application challenges:** Different users sharing a wireless network can have different minimum throughput requirements based on the applications they are running. In the absence of a bandwidth

[1] Network administrators try to have adjacent cells operate on different channels, but due to the limited number of channels and complex network topologies, this is not always possible.

arbitration mechanism, each user/application will pump data into the network as fast as it is generated. If the sum of the rates exceeds the channel capacity, then there will be severe performance degradation for all users/applications. One of the challenges is thus the design of bandwidth allocation policies that distribute the channel bandwidth among users/applications based on their requirements. The bandwidth arbitration policy must ensure that the available bandwidth is fairly shared, even while users/applications are guaranteed their minimum throughput. Alternatively, a price-based incentive mechanism can be used to allocate a larger share of the bandwidth to users who pay more, while still guaranteeing minimum throughput to all admitted flows. Providing some throughput guarantee is essential for multimedia flows because audio and video applications have stringent QoS requirements. The aim of the bandwidth allocation policy is to simultaneously increase user satisfaction, increase channel utilization and, in the case of a price-based policy, to increase revenue for the network provider.

2. **Network challenges:** A wireless network is a dynamic environment. A mobile host waiting to transmit a packet on a shared channel may find it busy a lot of the time, if there are many other hosts also contending for the channel. If the amount of contending traffic is small, then the channel busy delays are reduced. Furthermore, mobility and physical objects such as walls, doors, etc. introduce signal fading that can adversely affect a mobile host's packet transmissions. The wireless channel phenomena of contention and fading experienced by a single source-destination flow may vary with time. In addition, they may also be different for different flows at the same time, depending on the location of the respective sources and destinations.

   A high degree of fading effects and contention on a wireless channel represents poor channel quality and causes reduction in channel capacity. It impedes successful packet transmission, and consequently causes deterioration in throughput. A multimedia flow with strict minimum throughput requirements is thus adversely affected. A flow that experiences poor channel quality requires a larger share of the channel in order to maintain its minimum throughput, since its individual packet transmissions now take longer to be completed. Since channel quality is time-varying and also varies from flow to flow, each flow must independently and continuously monitor its channel quality and estimate available channel capacity. It must re-negotiate its throughput requirements with the bandwidth arbitration mechanism as and when it perceives the channel quality to have changed. Thus one-time bandwidth

allocation at the time of flow startup is not sufficient and dynamic bandwidth adaptation in response to variation in channel characteristics is an integral function in providing throughput guarantees in a wireless network.

The IEEE 802.11 protocol [2] can operate in one of two modes: Point Co-ordination Function (PCF) mode or Distributed Co-ordination Function (DCF) mode. In the PCF mode, a central co-ordinator determines the transmission schedule and polls the other mobile hosts at appropriate intervals of time for data. There is no contention for the medium. In the DCF mode, the mobile hosts contend for the medium using a distributed *carrier sense multiple access with collision avoidance* (CSMA/CA) protocol. Most current vendors of IEEE 802.11 products do not implement the PCF mode, thus making the DCF mode the default standard of IEEE 802.11 operation, even for the AP-based wireless LAN topology, where a central co-ordinator is readily available. A new wireless multimedia architecture must be designed in such a way that it is easily interoperable with off-the-shelf IEEE 802.11 DCF implementations.

3. **Traffic challenges:** Once a user/application has been allocated a share of the channel bandwidth by the bandwidth arbiter, it must ensure it conforms to this allocated share by restricting its transmission rate. Flows must co-operatively control their packet transmission rates so that no one flow eats into the guaranteed channel share of another flow elsewhere in the network, causing its throughput requirements to be violated. One possible way to perform rate-control is through fair scheduling at the MAC-layer. However, this involves modification of the basic IEEE 802.11 protocol. Such a bandwidth management architecture would hence not be feasible with current off-the-shelf IEEE 802.11 cards. There is thus a challenge to design distributed traffic-control schemes at the higher layers of the OSI protocol stack, that can be readily deployed in existing IEEE 802.11 networks. Fair scheduling at the MAC-layer ensures very precise conformation to the allocated rate, even at very small time-scales. The higher-layer traffic-control schemes may be permitted to trade-off some of this precision for interoperability with current off-the-shelf products, while still providing fair channel usage at the larger time-scales.

   A bandwidth management solution must also address different types of traffic, namely real-time and best-effort. Minimum bandwidth, delay, and throughput-stability requirements of real-time traffic must not be violated, and best-effort traffic must not be starved. We have found that traffic-control

ensures short queues, thus reducing delay experienced by real-time traffic. The transmission delay over small-scale wireless networks is small compared to the queuing delay.

1.3 Overview of Dynamic Bandwidth Management Architecture: Contributions

The center-piece of our bandwidth management architecture is a *Bandwidth Manager* (BM) which is unique to a wireless subnet. The BM runs a policy-based channel time allocation algorithm. The policy could be a fair policy, a price-based policy, or a utility maximization policy. When a flow starts up, it specifies its minimum and maximum bandwidth requirements to a middleware agent. Depending on the instantaneous channel quality perceived by the flow, these minimum and maximum bandwidth requirements are converted into *channel time proportion* (CTP) requirements. The CTP represents the fraction of unit time a flow must have access to the channel in order for it to transmit a sufficient amount of data to satisfy its throughput requirements, given a certain channel quality. The minimum and maximum CTP requirements of the flow are obtained by the BM. In addition to the CTP requirements, for the price-based allocation scheme, the flow's budget is also sent to the BM. The BM computes and returns CTP allocations for all the flows based on their CTP requirements using the appropriate channel time allocation algorithm. A rate-control scheme used controls the packet transmission rate of the flow so that it conforms to its allocated channel share. The flows in our system are *managed*, i.e., the rate-control for the individual flows co-operatively ensures that the individual flows refrain from exceeding their allotted channel share and eating into other flows' share. There is a MAC-layer channel quality monitor that is polled periodically for a channel capacity estimate. If the channel capacity estimate changes significantly, re-negotiation for channel time occurs, because a flow's channel share must depend on the channel quality it perceives. The MAC-layer channel quality monitor does not modify the IEEE 802.11 protocol in any manner. The flows are also co-operative in the sense that the middleware agent contacts the BM and releases any excess channel share allocated to them, either when their traffic pattern changes or the channel quality changes. A policing mechanism (see Chapter 4) can be used to detect the unmanaged "rogue" flows and eliminate them from the system.

The QoS guarantee provided by our bandwidth management scheme is that each flow will obtain at least its minimum requested CTP for almost 100% of its active duration. If at any time the system cannot

guarantee the flow at least this level of QoS, it will reject it altogether. The guarantee in terms of bandwidth is that the allocated bandwidth to a flow never falls more than some pre-determined factor below the minimum requested bandwidth.

A key feature of our architecture is the use of *cross-layer design* in tackling the wireless bandwidth arbitration problem. The flow's agent that communicates its parameters with the BM, the MAC-layer channel quality monitor, and the rate-control module may all be at different layers of the OSI protocol stack in a given mobile host. However, they co-operate with each other in providing the application its required QoS. The co-operation is accomplished via the exchange of vital state information pertaining to the flow's operation, such as the channel quality observed and the CTP allocated by the BM.

There exist several cross-layer bandwidth management architectures in literature [3, 4, 5, 6, 7] that specifically address one or more of the individual challenges outlined in the previous section, for various types of network topologies. For example, dRSVP [6] continuously measures the available bandwidth out of a IEEE 802.11 wireless interface in a MANET environment. It distributes the available bandwidth on an interface among different end-to-end flows that traverse that interface, using an enhanced version of the Resource reSerVation Protocol (RSVP) for reservation. This scheme does not however consider the location-dependent nature of the available bandwidth. More importantly, it also does not address the problem of bandwidth arbitration between flows traversing different interfaces that contend for the same shared channel. Ours is the first architecture that simultaneously addresses all the concerns listed in the previous section in a co-ordinated manner, for various IEEE 802.11 topologies.

Using our dynamic bandwidth management architecture, we can provide a high degree of QoS guarantee to multimedia applications in the presence of wireless network dynamics, while not starving best-effort traffic. **We show that centralized bandwidth arbitration can be used to provide throughput guarantees over different IEEE 802.11 topologies.** Our solution works with off-the-shelf IEEE 802.11 products and with common wireless network topologies. One major contribution of our work is a continuous channel capacity estimation mechanism that accurately captures both the spatial and temporal variations in channel quality. The bandwidth allocation to an individual user/application varies with variation in the channel quality it experiences. The concept of *channel time proportion* (CTP) used in our work is a unique way of bringing users/applications with different channel qualities on a level playing field, at the time of bandwidth allocation. In this work, we also present several novel bandwidth allocation policies: fair, price-based, and

utility-maximization-based. Furthermore, we explore new methods of rate-control apart from MAC-layer scheduling.

1.4 Outline of this Work

The following are the tasks involved in bandwidth management for IEEE 802.11 networks:

1. Continuous channel capacity estimation which takes into account location-dependent fading and interference effects.
2. Normalization of bandwidth requested with respect to channel quality observed.
3. Communication between mobile hosts and bandwidth arbiter of bandwidth requests and allocations.
4. Bandwidth allocation policy at the arbiter.
5. Co-operative traffic control by users/applications to conform to bandwidth fraction allocated.
6. Adaptation in response to changing network and traffic conditions.

In the following chapters we will describe in detail the mechanisms we employ in our bandwidth management solution to perform each of the tasks listed above. In the next chapter, we present a network model to formalize the problem and define the notation used in this work. In chapter 3, we discuss in detail our channel capacity estimation scheme and the normalization of a user/application's requested bandwidth with respect to channel quality it observes to its destination. Chapter 4 contains a description of the overall architecture of our bandwidth management scheme and the communication protocol used. Mechanisms for co-operative traffic control and for adaptation to changing network conditions are also addressed in this chapter. Chapter 5 describes the two bandwidth allocation policies that we experimented with. Chapter 6 contains results of experiments conducted with the ns-2 [8] network simulator as well as those conducted in a wireless testbed. Chapters 7 and 8 extend the basic architecture to more complex IEEE 802.11 network topologies. Chapter 9 summarizes related work in this area. Chapter 10 outlines areas of future research and Chapter 11 concludes this work.

Chapter 2

Network Model, Notation, and Framework

In this chapter, we describe the notation and formalize the model of the network used in this work. We first describe in general terms the parameters involved in our bandwidth management architecture. The AP-based wireless LAN model simplifies the use of some of these parameters. In addition to the general parameters, a few more parameters are required for the price-based bandwidth allocation scheme. Table 2.1 encapsulates the notation used in this work.

2.1 General Network Model

In the previous section, we briefly surveyed three IEEE 802.11 network topologies. In our work, we address the issue of bandwidth management in the two so-called *single-hop* topologies. In these topologies, each node in a wireless subnet is within the transmission range of every other node in the same subnet. Hence, only one node in the subnet can transmit at a time over the channel. Since every node is within the transmission radius of every other node, routing within a wireless subnet is single-hop.

Each wireless subnet has one elected system that hosts the bandwidth arbiter called the *Bandwidth Manager* (BM). In our prototype implementation over a single-hop, peer-to-peer wireless LAN, we choose one of the more resource-rich nodes in the subnet, i.e. one of the laptops, as the host system for the BM. We assume that the BM program resides on a well-known port in a system whose IP address is well-known in the wireless network. A service discovery mechanism such as the ones described in [9, 10] can be used to obtain the IP address and port number of the BM service. A small-scale, single-hop subnet lends itself naturally to the centralized bandwidth arbiter architecture. However, in later chapters we show that the centralized scheme, contrary to intuition, scales well to larger-scale multi-cell and multi-hop wireless

networks consisting of tens or hundreds of nodes also, but not to very large-scale networks comprising thousands of nodes. We explain the reasons behind this in later chapters. The BM has to register with the service discovery system upon startup. If the BM suddenly becomes unavailable, due to a crash or due to mobility, an *election algorithm* can be run to elect a new one after a time-out.

We assume a wireless subnet has a set of flows F. Each flow $g \in F$ is uniquely identified by its source IP address, source port number, destination IP address and destination port number. We call this unique identifier the *flow-id* of the flow. A new flow f registers with the BM before beginning its transmission. The application initiating flow f has a minimum bandwidth requirement $B_{min}(f)$ and a maximum bandwidth requirement $B_{max}(f)$. A best-effort flow will have $B_{min}(f) = 0$. Figure 2.1 shows the shape of the utility curve of the application. Such a utility curve has also been used previously in literature [6]. The advantage of a linear utility function is that it is easier to obtain and communicate. Minimum and maximum bandwidth requirements for an application/user are more intuitively understood, and easier to derive, than more complex functions. Specifying the function to the bandwidth arbiter is also easier, since there are only two parameters. We later extend our scheme to stepwise utility functions also, which have also previously been used in literature [4]. Continuous, convex functions used in [11, 12] can be approximated as linear utility functions also, by substituting the asymptotic portions with straight lines and connecting them with a linear slope.

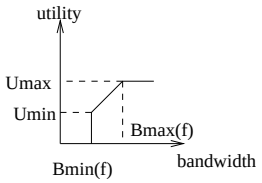
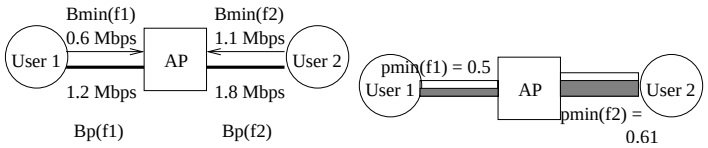


Figure 2.1: Utility curve of flows.

A flow f also has an estimate of the channel quality, including channel capacity $B_p(f)$ and loss rate $L(f)$, over the shared channel from its source to its destination host, which it obtains from the MAC-layer channel quality estimation mechanism located at its source. This MAC-layer channel quality estimation mechanism is described in the following chapter. Note that the channel capacity estimate can vary from source-destination pair to source-destination pair in a wireless subnet, due to location-dependent fading and interference. Different flows emanating from the same source to different destinations can thus have different values of $B_p(f)$ and $L(f)$.

One major contribution of our scheme is that we convert bandwidth requirements into *channel time proportion (CTP)* requirements. The channel time proportion is the fraction of unit time a flow utilizes the channel for data transmission. This is directly related to the bandwidth because, the larger this fraction is, the more data in unit time a flow can transmit, and thus the larger the flow's throughput is. The details of the conversion of bandwidth requirements into CTP requirements are discussed in the next chapter. The minimum bandwidth requirement of flow f, $B_{min}(f)$ is converted into a minimum CTP requirement $p_{min}(f)$ and the maximum bandwidth requirement $B_{max}(f)$ is converted into a maximum CTP requirement $p_{max}(f)$. The conversion takes into account the channel capacity estimate $B_p(f)$ and loss rate estimate $L(f)$. Note that $0\% \leq p_{min}(f) \leq p_{max}(f) \leq 100\%$.



(a) Using bandwidth in admission control.

(b) Using CTP in admission control.

Figure 2.2: Difference between using bandwidth and CTP for admission control.

Figure 2.2 shows the difference between using CTP requirements for admission control as opposed using pure bandwidth requirements as in [6]. Figure 2.2(a) shows the uplink bandwidth requirements, and estimated uplink channel capacities for two users. Using only bandwidth for admission control as in [6], since the required bandwidth is less than channel capacity for both users, they are both admitted. In Figure 2.2(b), the CTP requirements for the two users are shown. These are computed from the bandwidth requirements and effective channel capacities, as described earlier. It turns out that, when admission control is done using CTP, both users cannot be admitted because, in sum, they require more than 100% of unit time on the channel, which is obviously not possible. If admission control based purely on bandwidth requirements and not CTP (Figure 2.2(a)) is done, then an overflow will occur at the interface queue. Essentially, using bandwidth in admission control is possible for point-to-point wireline networks, but in shared channels, only one host can be active on the air at an instant, and hence, care must be taken to ensure that the total active air time for all hosts does not exceed unity. The CTP requirement represents air time required by a particular host.

Essentially, in a point-to-point wireline network, admission can be done independently and locally per-user using bandwidth alone, since all links are independent and shielded from each other. In a shared wireline network, although the link is shared, the effective capacity of the link is perceived to be same by all users. Thus, admission control must be co-ordinated, but can be on the basis of bandwidth alone. Only in a shared wireless network, is the channel shared, and also differing in quality for different users.

On the other hand, in a wireless network, not only is the air channel shared, but depending on different fading and interference levels for each user, the channel capacity perceived by each user is different. This means admission control must not only be co-ordinated, but must also take into account the different channel quality/capacity perceived by each user. A metric is required which brings users with different channel qualities on the same level plane for admission control. We therefore use a different metric, *channel time proportion*, and perform co-ordinated admission control, to account not only for the shared nature of the resource, but also its location-varying nature.

At the time of joining a wireless subnet, flow f specifies its minimum and maximum CTP requirements, $p_{min}(f)$ and $p_{max}(f)$, to the BM. If the flow f is admitted, the BM adds it to its set of admitted flows F and allocates a certain CTP $p_a(f) \in [p_{min}(f), p_{max}(f)]$ to it. If flow f is rejected, then it is not added to F and its allocated CTP $p_a(f) = 0\%$. Flow f, once admitted, uses this allocated CTP $p_a(f)$ to calculate its transmission rate. It transmits using this transmission rate until either it stops or until a new $p_a(f)$ value is allocated to it. A new $p_a(f)$ could be allocated to it when there is a change in the channel characteristics or in the network traffic characteristics.

2.2 Network Model for AP-based Wireless LANs

In the previous section, we described a general network model for peer-to-peer communication over IEEE 802.11. The AP-based wireless LAN topology is a *subset* of the peer-to-peer topology, because each host now has only two flows: uplink to the AP, and downlink from the AP. Identifying a flow is also thus easier, since instead of the source and destination IPs/ports, only one IP/port number combination is now required, since the other end-point is always the AP. A boolean variable indicating whether the flow is uplink or downlink is also required. The solution for peer-to-peer topologies is also applicable to AP-based topologies.

The AP-based wireless LAN is merely a special case of the single-hop peer-to-peer network, but with no peer-to-peer communication between mobile nodes (see Figure 1.1). All communication is between the

AP and the mobile hosts. Most current AP-based wireless LANs also use IEEE 802.11 DCF, because PCF, although better suited to such a topology, is not supported by many vendors. Since DCF is also used for the AP-based topology, the contention characteristics are similar to those in a single-hop peer-to-peer network. There are some variations, however, since all transmissions in this topology involve the AP, but not in the peer-to-peer toplogy. Uplink and downlink traffic between a particular mobile host and the AP can simply be considered as two separate single-hop flows, and their respective CTP requirements can be allocated accordingly by the BM. The BM is situated in the distribution system of the extended (i.e. multi-cell - see also Chapter 7) wireless LAN, or even at a remote location.

Since uplink and downlink flows are the only types of flows possible, channel capacity estimates are only required for the AP-to-mobile host bi-directional link. All flows pertaining to a mobile host have the same two end-points: the mobile host itself and the AP. This is in contrast to the peer-to-peer model where different flows emanating from a single source mobile host can have different destination mobile hosts. Bandwidth management for peer-to-peer wireless LANs is thus *per-flow*, since each flow can traverse a different source-destination link and thus experience different location-dependent channel conditions. On the other hand, bandwidth management for AP-based wireless LANs is *per-host*, since all flows pertaining to a mobile host traverse the same AP-to-mobile host bi-directional link and thus encounter the same channel conditions. Thus while a mobile host in a peer-to-peer wireless LAN has multiple values of $B_p(f)$ and $L(f)$ for different flows f for which it may be the source or destination, a mobile host in a AP-based wireless LAN has only one $B_p(f)$ and one $L(f)$ estimate, for the uplink to the AP. The AP has multiple $B_p(f)$ and $L(f)$ estimates, one for each downlink. The minimum and maximum CTP requirements are both calculated separately for uplink and downlink flows, and added to give one unified set of minimum/maximum CTP requirements for a particular host in a wireless LAN. In general, the AP-based wireless LAN topology is a subset of the single-hop peer-to-peer network topology. The flow-id of a flow in an AP-based wireless LAN is merely the IP address of the mobile host and an identifier which indicates whether it is an uplink or a downlink flow.

2.3 Network Model for Multi-Cell and Multi-Hop Networks

Figures 2.3 and 2.4 show the network architectures for a typical multi-cell network and a typical multi-hop network respectively. A multi-cell network consists of a number of AP-based wireless LANs, with

overlapping coverage areas. The APs are connected together in a backbone distribution system, which provides wireline access to the Internet. The entire network is operated by a single administrator. The multi-cell or extended set wireless network is used to provide greater geographical coverage to support mobile users. The overlapping cells provide seamless connectivity as users move from one cell to the other. A user switches from one cell to another when the SNR of the beacon of the AP in the second cell is stronger than the SNR seen by the user in the first cell.

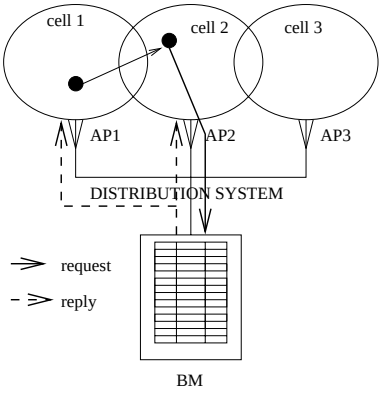


Figure 2.3: Multi-cell network.

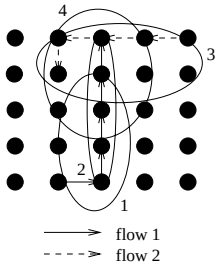


Figure 2.4: Multi-hop scenario.

We assume users are uniquely identified in the multi-cell networks, and communicate only with the AP, not peer-to-peer. Bandwidth allocation is done over the entire extended set network, not per-AP. This is because there may be interference between nodes in adjoining cells of the network. These nodes thus share the medium, and one cannot allocate resources independently for such nodes.

All wireless transmissions in a multi-cell network are still single-hop. In a multi-hop network, we assume that there is a routing protocol and forwarding capability in the network stack of the wireless nodes. We also assume peer-to-peer communications with neighbors, and end-to-end construction of routes. In this work, we assume the nodes in a multi-hop network are static, while in the single-hop and multi-cell

scenarios we assume mobility. This is because our centralized bandwidth management architecture scales to single-hop and multi-cell mobile environments, but only to static multi-hop network topologies of a few tens or hundreds of nodes. Thus in a multi-hop network, a single end-to-end connection comprises multiple wireless hops along the way. Once again, there exist some single-hop transmissions that interfere with each other on the network, and other single-hop transmissions that do not. We use the concept of *maximal cliques* [12, 30] to determine which nodes interfere with each other and which do not.

We represent individual one-hop transmissions in the multi-cell or multi-hop network as points in a graph, and links between two points in the graph represent interference between the respective subflows. A maximal interference clique in such a graph represents a set of one-hop transmissions wherein each one-hop interferes will all the others in the set. A maximal interference clique is a set of one-hop transmissions (belonging to the same or different end-to-end flows) that share unit channel capacity. Only one transmission in the clique can be active on the channel at an instant in time. The maximal interference cliques determine *which* transmissions in the network interfere with each other. A maximal interference clique comprises several one-hop transmissions; a one-hop transmission can be a member of several maximal cliques. Details can be found in Chapter 8.

The bandwidth requirements $B_{min}(f)$ and $B_{max}(f)$, the channel quality estimates $B_p(f)$ and $L(f)$, and the CTP requirements $p_{min}(f)$ and $p_{max}(f)$ are still defined the same way in the multi-cell environments as they are in the single-hop case. In our work on multi-hop networks, we assume a stepwise utility function for flows, so there are discrete number of allocatable bandwidths (one corresponding to each step on the function), and a corresponding CTP requirement for each bandwidth level. For details see Chapter 8.

2.4 Pricing Parameters

To allocate bandwidth and corresponding channel time, we explore pricing policies, especially the bidding approach. We use a centralized arbiter of channel time for a wireless LAN subnet. We call this arbiter the *Bandwidth Manager* (BM). The BM could conceivably use any type of arbitration algorithm to allocate the channel time resource (e.g., fair, priority-based, price-based). We choose a bidding/pricing-based approach in our arbitration to address hot spot environments in airports, hotels, coffee-shops, etc. The main advantage of using a bidding/price-based scheme is that it prevents users from dishonestly requesting more resources than required. If they do so, they will either (a) be less competitive if they choose to keep their bid the same,

or (b) lose financially, if they increase their bid also to keep up with the artificially inflated requirement. Each user submits a bid to the BM, in addition to its CTP requirements. A bid is a maximum sum of money that a user is willing to pay towards the satisfaction of his/her requirements. The BM takes into account the bids and requirements of all users in allocating channel time. In this subsection, we first describe the user's parameters, especially the bid details, and then describe the system parameters.

User View: Each user f has a *maximum* price $\rho_u(f)$ that he/she is willing to pay per minute towards every 1% of the minimum channel time requested [1]. $D(f) = \rho_u(f) \cdot p_{min}(f)$ is thus the *maximum bid* (in cents per minute) that the user f is willing to make towards its minimum CTP requirements. The user may get *more* CTP than just the minimum, for this bid, but if admitted, he/she will get no less than the minimum. If not admitted, he/she will get no CTP, and will be charged nothing. If $p_{min}(f) = 0\%$, i.e., for a best-effort user, $D(f)$ is set differently. It is set directly by the user, based on the importance of the best-effort channel time to the user's activity. It is the maximum bid in cents per minute that the user is willing to make for any CTP that it can manage to obtain. The values of bid $D(f)$ and bandwidth requirements are valid for user f's whole session, unless the user changes them in-between. The values of $p_{min}(f)$ and $p_{max}(f)$ change with a change in both the bandwidth requirements, as well as with a significant change in the channel capacity estimate in either direction. Changes must be conveyed to the BM to initiate re-allocation of resources with the new requirements. A threshold is set so that instantaneous fluctuations in $p_{min}(f)$ and $p_{max}(f)$ are not reported [13], and only variations above this threshold are reported.

We define *price-index* of user f as $mp(f) = \frac{D(f)}{p_{max}(f)}$. A user's price-index is its bid (i.e., its "richness") normalized over its maximum CTP request. We define price-index here because it is an integral metric in our price-based channel time allocation algorithm.

Users may not personally be aware of the concept of channel-time, only that of bandwidth, and may hence only specify their maximum price per unit of bandwidth. In this case, the price per unit bandwidth must be converted into $\rho_u(f)$ which is price per unit CTP. Our algorithm only deals with $mp(f)$, which is obtained from price per unit CTP, not price per unit bandwidth.

Service Provider View: A centralized *Bandwidth Manager* (BM) co-located with the access-point (AP) takes as input the CTP requirements of all active users f and their bids $D(f)$. It runs the algorithm and

[1] We have arbitrarily picked minutes and cents, respectively, as the units of time and money in this work. Similarly, the minimum resolution of CTP used in this work (1%) has been arbitrarily chosen. Network providers can set these parameters to suit their network.

returns to the users the current channel time price ρ_s (in cents per minute for 1% of the channel time). ρ_s is a system-wide price set by the BM for 1% of channel time. It is calculated by the price-based channel allocation algorithm, based on the $mp(f)$ values submitted by the users. In addition to the system-wide price, the BM also returns to each user his/her CTP allocation $p_a(f)$ computed by the algorithm using this price. The algorithm is presented in Section 5. The revenue obtained by the system from a particular admitted user f, in cents per minute, is thus $R(f) = \rho_s \cdot p_a(f)$, where $p_a(f) \in [p_{min}(f), p_{max}(f)]$. In other words, the revenue from a particular user is its allocated CTP multiplied by the system-wide price per unit of CTP.

If this revenue $R(f)$ from a user f ends up being smaller than user f's bid $D(f)$, then the balance $\lambda = D(f) - \rho_s \cdot p_a(f)$ is refunded by the BM to the user. The refund concept is very important because it encourages the user to bid high. If the bid is excessive, no harm is done, since the balance will be refunded. (By "refunded", we mean "not charged", i.e. the amount charged is less than the amount offered.) The bid $D(f)$ is hence a *credit* in the user's account. The system *honestly* draws from this credit as much as is required to satisfy the user's requirement, amidst the competition, while the rest, if any, remains with the user. Upon submitting this maximum bid $D(f)$, the user does not have to keep bidding as network conditions change. The pricing algorithm will determine what the user's bid would have been, had the user been free to bid, and do the bidding on behalf of the user, up to a maximum of $D(f)$. The user may change $D(f)$ at any time. However, we expect such changes to be rare because, the possibility of a refund encourages users to truthfully submit high bids to begin with.

It should be noted that users do not actually pay before obtaining service. Rather, they *offer* a maximum bid $D(f)$ towards their requested service. There is no transfer of money at this point. The system calculates, based on the users' bids, the system-price ρ_s, and applies it to the users' allotted channel shares $p_a(f)$, at each instant in time. At an instant in time, therefore, a user is charged $R(f) = \rho_s \cdot p_a(f)$ for service at that instant, based on the current values of ρ_s and $p_a(f)$. This user charge may be less than the user bid $D(f)$, and the difference is what we refer to as a "refund." Once again, there is no transfer of money involved. The user charge is aggregated over the time the user spends in the system, and billed to the user when he leaves the system. This is when there is an actual money-transfer.

The instantaneous revenue of the network provider (i.e., the BM), in cents/min, is $R = \sum_{f \in F} R(f) = \rho_s \cdot \sum_{f \in F} p_a(f)$, where F is the set of admitted users. The total revenue is the value of R aggregated over the

entire time of operation of the network. If the set F changes (users arrive or leave), or if a user changes its parameters, then the price ρ_s and the allocated CTPs of all the users f have to be recomputed, and the value of R in cents/min also changes. The algorithm also has a fixed *reserve price* ρ_r, computed from the cost of installation and maintenance of the network, that a user is minimally charged for service in the absence of any competition. The system price ρ_s determined by the algorithm is no less than the reserve price at any point in time ($\rho_s \geq \rho_r$), thereby ensuring that the benefit to the network provider is non-negative. The extent of the benefit depends on the price set by the algorithm, which depends on the demand for channel time.

Notation	Meaning
F	Set of flows admitted by the BM.
$g \in F$	All individual flows previously admitted by the BM.
f	New flow requesting admission.
$B_{min}(f)$	Minimum bandwidth requirement of flow f.
$B_{max}(f)$	Maximum bandwidth requirement of flow f.
$B_p(f)$	Channel capacity perceived by flow f.
$p_{min}(f)$	Minimum channel time proportion required by flow f.
$p_{max}(f)$	Maximum channel time proportion required by flow f.
$p_a(f)$	Total channel time proportion allotted to flow f.
$\rho_u(f)$	Maximum price flow f is willing to pay in cents per minute for 1% of CTP.
$D(f)$	Flow f's maximum bid in cents/min.
$mp(f)$	Price-index of flow f.
ρ_s	Current channel time price computed by BM in cents per minute for 1% of CTP.
ρ_r	System reserve price.
R	Instantaneous revenue for the network provider.

Table 2.1: Explanation of notation used in this work.

2.5 Framework

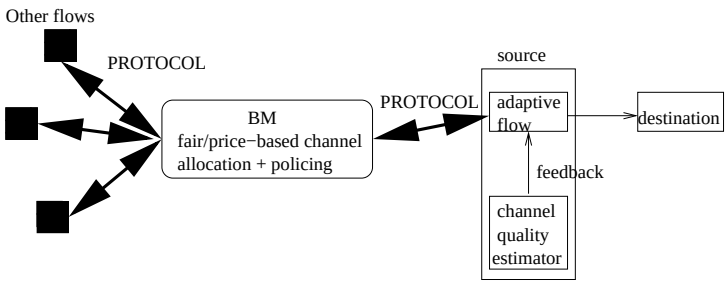


Figure 2.5: Bandwidth management framework.

Our bandwidth management solution employs a *feedback*-based framework. At the time of flow es-

tablishment, each flow negotiates for some fraction of the channel. The bandwidth manager (BM) uses its bandwidth allocation policy to determine what fraction of the channel this flow can be granted. It may need to revoke some channel time from other previously admitted flows, while ensuring that their minimum operation quality level is not violated, in order to be able to offer channel time to the new flow. The bandwidth is revoked and reallocated based on some policy - fair or price-based - hardwired at the BM. Once the flow is established, it must ensure that it conforms to the fraction of the channel allocated to it. Periodically, it receives *feedback* from the channel about the channel quality it is experiencing. Based on this channel quality feedback, the flow must adapt itself. If the feedback indicates that the channel quality is poor, then the flow must attempt to secure a larger fraction of the channel so that it can still send over the error- and contention-prone channel the minimum quantity of data it needs to send in order to maintain a certain quality level. In attempting to secure a larger fraction of the channel, the flow must once again contact the BM so that the BM can use its policy to adjust the channel fractions of all the flows, to account for the changed channel conditions being experienced at some locations. In addition to channel allocation, the BM must also perform *policing* in order to catch "rogue" flows that refuse to conform to their allocated channel share. The rogue flows must then be dealt with offline, since it is not possible automatically to *prevent* anyone from transmitting on the air at 802.11 frequencies.

Our methodology thus involves an adaptive flow, a central channel arbiter configured with a channel allocation scheme, and a communication protocol between them. In addition to this, our scheme also involves a channel quality estimator that periodically reports to the flow the channel quality it is experiencing. The estimator provides feedback to the flow regarding its performance, and triggers flow adaptation in the case of variation in channel quality. In the next chapter, we describe the working of the channel quality estimator. In Chapter 4, we describe the interaction of the flow with the BM and elaborate on the flow adaptation. In Chapter 5, we discuss in detail two possible bandwidth allocation policies that can be used at the BM. Figure 2.5 illustrates the framework of our bandwidth management solution.

Chapter 3

Channel Quality Estimation

In this section we describe both channel capacity estimation and channel loss rate estimation. First we describe channel capacity estimation, and then motivate and describe channel loss rate estimation.

Channel capacity estimation is vital to guarantee minimum throughput to applications in wireless networks. A channel arbiter must know the channel capacity to be able to distribute it between the contending flows so as to satisfy the flows' individual requirements. The channel capacity is inversely related to the quality of the channel in a wireless network. The higher the levels of contention [14] and of fading errors are, the lower the available channel bandwidth for the flows is. Channel capacity estimation is thus equivalent to channel quality monitoring.

Channel capacity estimation is complex in wireless networks because fading and channel contention phenomena vary both with time as well as location in the wireless network, thus necessitating continuous, distributed channel capacity estimation. Our channel capacity estimation scheme does not modify the IEEE 802.11 DCF CSMA/CA MAC protocol in any manner, but gauges the effect of phenomena such as contention and fading, which influence the available channel capacity, on it. Based on the effect of the phenomena on the working of the medium-access scheme, each mobile host estimates the total channel capacity to each of its neighbors [1].

A Channel capacity estimation module is co-located with the IEEE 802.11 protocol at the MAC layer of every mobile host in the network. In a single-hop peer-to-peer wireless LAN, it obtains a channel capacity estimate $B_p(f)$ per flow f sourced at the host it resides on. A bi-directional flow with higher-layer data and higher-layer acknowledgements is considered at the MAC layer as two uni-directional flows. Channel capacity estimation is done for both flows. The channel capacity estimated for the higher-layer acknowl-

[1] A neighbor of a mobile host is defined as any other mobile host within its transmission range.

edgement flow at the destination is transmitted in-band to the source. CTP computation and re-negotiation for the flow as a whole is done at the source taking into account the CTP requirement of both the higher-layer data and acknowledgement packets. Alternatively, since the acknowledgement flow requires only fraction of the channel compared to the data flow, a small constant overhead can be added to the CTP computed for the data flow, instead of precisely computing the acknowledgement flow's CTP. This introduces a very slight inaccuracy but simplifies CTP computation.

In a single-hop peer-to-peer wireless network, we perform bandwidth management per-flow, since each flow can have a different destination. In an AP-based wireless LAN, we can perform bandwidth management per-host since every host only communicates with one other host, i.e. the AP. There are thus only two flows per mobile host - one uplink to the AP and the other downlink from the AP. Both traverse the same AP-mobile host link. In this scenario, each host, rather than application, specifies its bandwidth requirements which must be converted to CTP requirements. Channel capacity estimation is done only for links between mobile hosts and the AP. The mobile host does the estimation for its uplink while the AP does it for each downlink. CTP computation is done separately for uplink and downlink flows at the AP.

3.1 Underlying Theory behind Estimation Mechanism

IEEE 802.11 relies on the DCF method to coordinate the transmission of packets based on CSMA/CA without any central co-ordinator. The packet transmission sequence is illustrated in Figure 3.1. Before transmitting a packet, a node senses the channel to make sure that the channel is idle; otherwise it backs off by a random interval and senses the channel again. If the channel is idle, it transmits a RTS (Request-to-Send) packet to signal its intention to send a packet[2]. On receiving the RTS packet, the destination node replies with a CTS (Clear-to-Send) packet to give the sender a go-ahead signal, and to silence the destination node's neighboring nodes. After receiving the CTS packet, the sender sends the DATA packet, and it is then acknowledged by an ACK packet from the receiver.

Figure 3.1 shows the stages in the transmission of a single packet using the IEEE 802.11 DCF MAC protocol. Details of the individual messages and gaps can be found in [2]. Similar to [15], we measure the throughput of transmitting a packet as $TP = \frac{S}{t_r - t_s}$, where S is the size of the packet, t_r is the time the ACK is received and t_s is the time that the packet is ready at the MAC layer. The time interval $t_r - t_s$ includes

[2]For very small packets, the sender may skip the RTS packet and directly send out the DATA packet.

the channel busy and contention time. The value of TP is normalized (see below) and the normalized value is averaged over all successful packet transmissions to a particular neighbor in a time interval. This gives the channel capacity estimate for that neighbor. All flows f to this neighbor use this value as their $B_p(f)$. We keep separate throughput estimates to different neighbors because the channel conditions may be very different to each one. We only keep an estimate for *active* links, since we do not have any packets to measure $t_r - t_s$ over inactive ones.

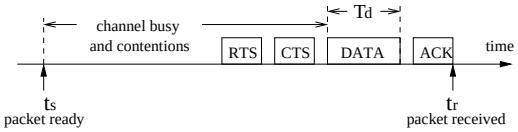


Figure 3.1: IEEE 802.11 unicast packet transmission sequence.

This link layer measurement mechanism captures the effect of contention on available bandwidth. If contention is high, $t_r - t_s$ will increase and the throughput TP will decrease. This mechanism also captures the effect of fading errors because if these errors affect the RTS or DATA packets, they have to be re-transmitted. This increases $t_r - t_s$ and correspondingly decreases available bandwidth. Our available channel capacity measurement mechanism thus takes into account the phenomena causing it to decrease from the theoretical maximum channel capacity, which can be 1, 2, 5.5 or 11 Mbps for current IEEE 802.11 networks.

It should be noted that the available capacity is measured using only *successful* MAC layer transmissions to a neighbor. In Section 3.6, we explain when there could be a large fraction of unsuccessful MAC transmission attempts, and how channel quality estimation is done in such a case. This channel capacity estimation method has also been used in [16].

3.2 Normalization

It is clear that the throughput measured using the scheme illustrated in Figure 3.1 depends on the size of the packet used. Larger packet has higher measured throughput because the protocol sends more data once it grabs the channel. To make the throughput measurement *independent* of the packet size, we normalize the throughput of a packet to a pre-defined packet size. In Figure 3.1, $T_d = S/BW_{ch}$ is the actual time for the channel to transmit the data packet, where BW_{ch} is the channel's bit-rate. Here we assume the channel's bit-rate is a pre-defined value: 1, 2, 5.5 or 11 Mbps for current IEEE 802.11 networks. The transmission

times of two packets should differ only in their times to transmit the DATA packets. Therefore, we have:

$$(t_{r1} - t_{s1}) - \frac{S_1}{BW_{ch}} = (t_{r2} - t_{s2}) - \frac{S_2}{BW_{ch}} \quad (3.1)$$

$$= \frac{S_2}{TP_2} - \frac{S_2}{BW_{ch}} \quad (3.2)$$

where S_1 is the actual data packet size, and S_2 is a pre-defined standard packet size. By Equation (3.2), we can calculate the normalized throughput TP_2 for the standard size packet. (The value of TP_2 is used by flows f traversing this wireless link as the perceived channel capacity $B_p(f)$ for this link.) The normalized throughput estimate must be scaled appropriately to a flow's mean packet size, using the inverse of Equation (3.2), before being used in the computation of the flow's CTP requirements. The normalization procedure is useful because it means we can keep a single channel capacity estimate per active outgoing link instead of multiple estimates pertaining to flows with different mean packet sizes traversing the same outgoing link.

It must be kept in mind that the channel capacity measurement is done using MAC layer frames. These MAC layer frames include protocol headers from the intermediate layers of the protocol stack between the application and the link layers. The scaling operation at the time of CTP computation must take into account the fact that the lower layers of the protocol stack will add their respective headers to each packet, and thus consume some of the channel capacity. The size of the lower-layer headers must be added to the application packet size in the scaling operation. Thus the size of the frame S at the MAC layer is calculated as $S = S_{app} + \sum_{i=1}^{n} h_i$, where S_{app} is the size of the data at the application layer, n is the number of layers of the OSI protocol stack between the application and physical layers and h_i is the size of the header added by layer i of the stack.

To verify the validity of Equation (3.2), we simulated a group of mobile nodes within a single-hop ad hoc network using the *ns-2* network simulator [8]. We sent CBR traffic from one node to another, and varied the packet size from small (64 bytes) to large (640 bytes) during the course of the simulation. The measured raw throughput is normalized against a standard size (picked as 512 bytes). Figure 3.2 shows the result of the measured raw throughput and its corresponding normalized throughput. Obviously, the raw throughput depends on the packet size; larger packet size leads to higher measured throughput. The normalized throughput, on the other hand, does not depend on the data packet size. Hence, we use the

normalized throughput to represent the bandwidth of a wireless link, to filter out the noise introduced by the measured raw throughput from packets of different sizes.

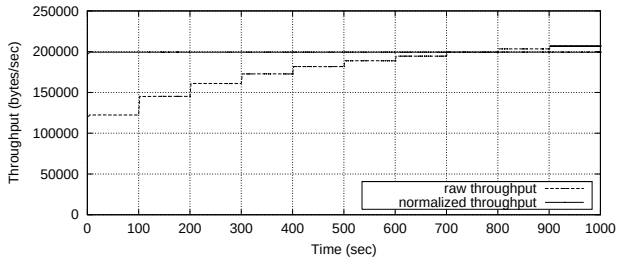


Figure 3.2: Raw throughput and normalized throughput at MAC layer.

3.3 Robustness of Channel Capacity Estimation

Another important issue is the *robustness* of the MAC layer bandwidth measurement. We measure the bandwidth of a link in discrete time intervals by averaging the throughputs of the recent packets in the past time window, and use it to estimate the bandwidth in the current time window. Obviously, this estimation may not be accurate because the channel condition may have changed. To evaluate the estimation error, we run a CBR flow over UDP with data rate 160 Kbps from a node to another in a 10 node one-hop environment. Background traffic consists of 1 greedy TCP flow in the light channel contention case, and 7 TCP flows in the heavy contention case. Here we use TCP only to generate bursty cross-traffic to the UDP flow. We measure and normalize the throughput of the CBR flow every 2 seconds using the average of packet throughputs in the past time window. Our results show that under light channel contention, over 97% of the estimates are within 20% of error; under heavy contention, still over 80% of the estimates are within 20% of error. We thus conclude that using average throughput of past packets to estimate current bandwidth is feasible and robust.

3.4 Discussion

It should be noted that the channel capacity estimation mechanism in no way alters the IEEE 802.11 protocol. Our mechanism, with the normalization extension, was satisfactorily accurate for the scenarios in our simulation and testbed experiments. However, the theory behind the normalization may not be appli-

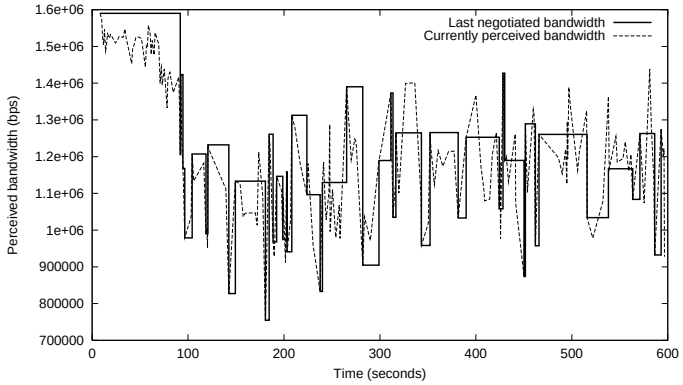
cable to arbitrarily large packet sizes and arbitrarily high bit-error rates. This is because, at high bit-error rates, the measured throughput only increases with increase in packet size upto a threshold. Beyond this threshold, the high bit-error rate means that larger packets have a greater probability of being corrupted and hence require re-transmission with a greater probability. The channel capacity estimate thus *decreases* with increase in packet size beyond this threshold. Basically, we assume that the bit-error rate in the network and the range of possible packet sizes is such that the channel capacity only increases with increase in packet size, thus permitting normalization. We believe this is a realistic assumption for most existing IEEE 802.11 networks. In our experiments using the ns-2 simulator, the range of possible packet sizes is 64 to 640 bytes, corresponding only to the increasing portion of the channel capacity estimate versus packet size curve.

In such cases of high bit-error rates and large packet sizes, the capacity estimation scheme could keep an indexed table of separate estimates for different packet size ranges per active link, rather than maintaining a single normalized estimate per active link and scaling it to various packet sizes at the time of flow establishment/re-negotiation. If the indexed table method is used, the source and destination must both perform total bandwidth estimation, for data and acknowledgements respectively. The destination must periodically communicate its bandwidth estimate for acknowledgement packets with the source using an in-band signaling mechanism. (The signaling itself consumes negligible bandwidth.) In the single normalized estimate method, the source alone does the estimation and appropriately scales the normalized estimate for both data and acknowledgement packet sizes. Thus, although the indexed table estimation method improves accuracy of the estimate in certain special cases, it also incurs a small storage space and in-band signaling overhead. We only implement the normalized estimate method, not the indexed table method.

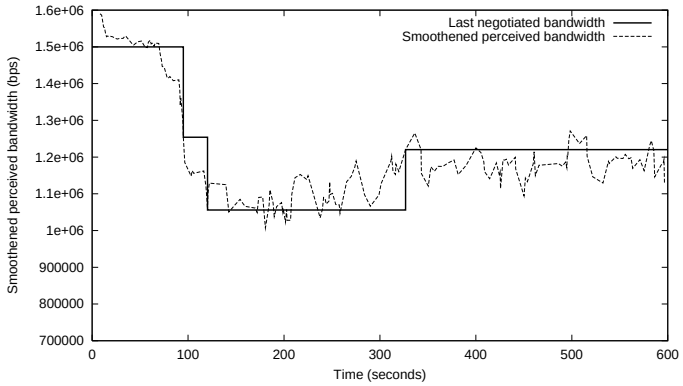
3.5 Channel Capacity Estimation in a Single Subnet

We experimented with the channel capacity scheme described above using the ns-2 network simulator. In one of the experiments, we used a 170m $\times$ 170m network area and the transmission range of each node was 250m. Hence, the entire network area falls within every node's transmission range, forming a single subnet IEEE 802.11 environment. The maximum theoretical channel capacity was 2 Mbps. We used the random waypoint mobility model with moderate node speeds in our simulations. Our simulation scenario consisted of a 20-node network with 10 flows from a randomly picked source to a randomly picked destination. Each flow had a transmission rate between 100 kbps and 200 kbps, which are typical rates for an audio streaming

application. All 10 flows used 512 byte packets. The simulation ran for 600 seconds. The 10 flows arrived one by one after the simulation started. Then, two flows were ended at time 149 seconds and time 264 seconds into the simulation respectively.



(a) Without smoothing.



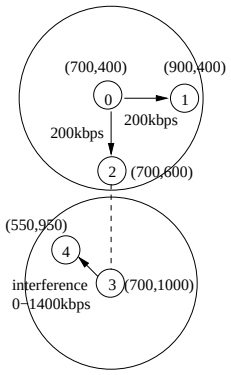
(b) With smoothing.

Figure 3.3: Perceived channel capacity and re-negotiations corresponding to its variation.

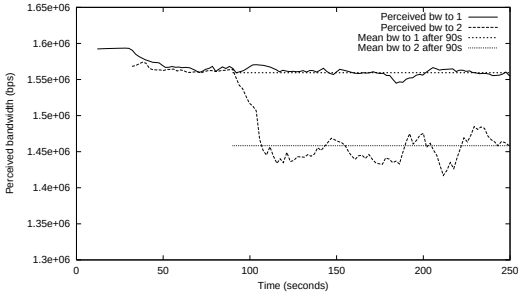
Figure 3.3(a) shows the variation of perceived channel capacity for one of the flows in the above experiment as measured by its channel capacity estimator at the MAC layer. When the perceived channel capacity varies by a factor $\delta \geq 15\%$, the flow has to re-negotiate its CTP requirements with the BM. The superimposed stepwise curve shows the bandwidth last used for re-negotiation. We have also experimented with smoothed perceived channel capacity estimates, which reduced the overhead of re-negotiation frequency. Figure 3.3(b) is a plot of a running average of the measured perceived channel capacity with exponential

decay, which is used for smoothing of the estimate. The smoothed estimate falls as contention increases, when the 10 flows arrive at the start of the simulation, and rises when the two flows end and contention decreases. This experiment shows how the channel capacity perceived by a flow varies with time.

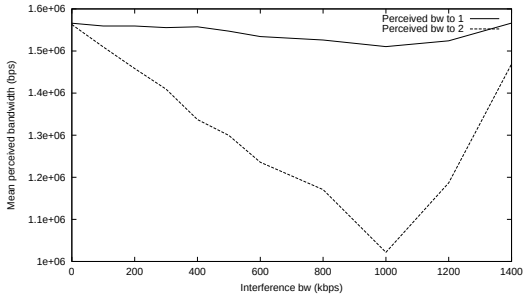
3.6 Channel Capacity Estimation in Multi-cell/Multi-hop Environments



(a) Scenario.



(b) Different throughput to different neighbors on the same interface.



(c) Accuracy of basic channel capacity estimation.

Figure 3.4: Channel capacity estimation in multi-cell environment.

In this ns-2 experiment, we show how channel capacity perceived by different flows varies with their location. We show the effect of interference from neighboring cells on capacity estimation in a particular cell. Transmission range is again 250m and interference range is 550m. The maximum theoretical channel capacity is 2 Mbps and all flows use packet size 512 bytes. The cartesian co-ordinates of the mobile hosts are indicated next to the hosts in Figure 3.4(a). The simulation lasts 250 seconds and the interference flow

starts 90 seconds into the run.

Figure 3.4(b) shows the performance of the channel capacity estimation scheme in the multi-cell scenario of Figure 3.4(a). Hosts 0, 1, and 2 are in one cell while hosts 3 and 4 are in another. Hosts 0 and 3 are the APs and their respective coverage areas, i.e. the transmission ranges, are shown in the figure. The dotted line in the figure shows that the interfering packet transmissions from host 3 can be sensed at host 2, since the interference range is greater than the transmission range[3]. The plot shows that the throughput observed out of the same interface (host 0) to different neighbors can be different based on different levels of contention. The interference flow from host 3 to host 4 is 200 kbps and starts 90 seconds into the simulation. It lies within the interference range of host 2 and hence contends with the flow from host 0 to host 2, but is out of range of host 0 and host 1 and hence does not affect the available bandwidth from 0 to 1. The channel capacity estimate thus varies with the location of the destination, even for flows emanating from the same source. A smoothing factor (running average with decay) has been applied to the estimated available bandwidth plots, as in the previous section.

In the second part of this experiment (Figure 3.4(c)), we increased the rate of the interference flow. As the bit-rate of the flow from 3 to 4 increases, the channel capacity estimate of the flow from 0 to 2 decreases until it reaches a knee. After this, the available bandwidth appears to increase with an increase in contention! This is obviously an anomaly. When the bit-rate of the flow from 3 to 4 increases, this flow practically captures the channel. RTS transmissions from node 0 to 2 are not acknowledged with a corresponding CTS, because the host 2 can sense the continuous transmission from 3 to 4 within its interference range. After several RTS re-transmissions, node 0 gives up and drops the packet[4].

Thus, depending on the rate of the interference flow, packets from host 0 to 2 are dropped after 90 seconds. The anomalous channel capacity estimate is a result of the way the historical decay in the running average works. Older samples decay only when newer ones become available. If packets from 0 to 2 are dropped after 90 seconds, few new samples (when interference flow is 1200 kbps) or no new samples (when interference flow is 1400 kbps) are available. Recall that $t_r - t_s$ is only computed for successfully transmitted packets, not dropped ones. Thus the capacity estimate continues to be incorrectly computed using samples

[3]Actually, hosts 3 and 4 can sense but not decipher the CTS transmissions of host 2, but the duration of these CTS transmissions is very short, and hence they do not produce much interference. Similarly, host 2 can also sense but not decipher CTS transmissions of host 4.

[4]This problem does not occur in a single-hop scenario because, in such a case, node 0 will also be able to sense the continuous transmission from 3 to 4 and will not send an RTS to begin with, during this time.

that were successfully transmitted before time 90 seconds. Essentially loss rate is high after 90 seconds, but the channel capacity continues to be computed using successful transmissions from before this time. Hence this channel capacity is inaccurate. We must find a way to *incorporate* the high loss rate somehow in the channel capacity estimate, to indicate that in fact, the flow from 0 to 2 is perceiving a near-zero channel capacity, after 90 seconds.

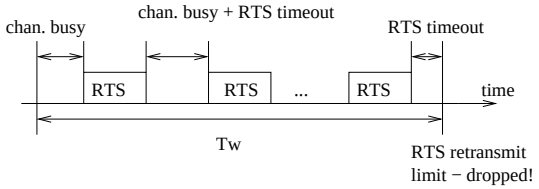


Figure 3.5: Raw throughput and normalized throughput at MAC layer.

The experiment above shows that the basic channel capacity estimation theory of Section 3.1 ceases to work at a particular source, when there is interference at the source from a bulk transmission from a neighboring cell. That is, when there is a bulk transmission within interference range of the source, but not transmission range. The bulk transmission in our experiment was $\geq$ 1200 kbps. As shown in the experiment, the perceived bandwidth readings from node 0 to node 2 become anomalous as the interference flow becomes larger than 1200 kbps. As mentioned earlier, this is because the estimation theory of Section 3.1 uses only successful packet transmissions in its channel capacity measurement, and even when most of the packets are lost (due to RTS retransmission limit being exceeding consequent to repeated CTS suppression), an incorrect channel capacity estimate is calculated using the few successful packets. In reality, we must take into account time lost due to unsuccessful packet transmissions[5] which affects channel capacity estimate. We must also estimate packet loss rate, since the lost packets must be compensated by the source, and extra bandwidth must be reserved for these compensation packets.

We augment the channel capacity estimation theory of Section 3.1 to estimate loss rate, and also take into account the effect of lost packets on channel capacity. We thus measure channel *quality* including both capacity and loss rate. In addition to measuring t_r and t_s for successful transmissions, we also measure T_w for unsuccessful packets, as shown in Figure 3.5. The figure shows the packet transmission sequence for an unsuccessful packet.

[5]Packet loss due to RTS retransmission limit being exceeded can also occur if bit-error rate on the channel is high as a result of fading effects. RTS/CTS/DATA/ACK frames may be repeatedly affected by bit errors causing repeated renewal of the RTS-ACK cycle via repeated RTS retransmission.

If the retransmit limit is reached for a flow's packet, the packet is dropped, resulting in a wastage of the flow's channel time, and an increase in its loss rate. This time wasted is represented by T_w in Figure 3.5.

The phenomenon of interference from neighboring cells is unique to the multi-cell 802.11 environment (i.e. multiple APs in the same network for extended coverage area) and causes packet loss due to CTS suppression. It occurs because the interference range for typical IEEE 802.11 transmitters is much larger than the transmission range. This results in a hidden node problem where bulk transmissions in one cell cause a carrier sense busy at a receiver in a neighboring cell, while the sender in that neighboring cell is out of range. Hence, the sender keeps transmitting RTSs repeatedly while the receiver keeps silent because it senses its channel to be busy. When a CTS is not received after a fixed number of RTS retries, the frame is dropped by the sender. This phenomenon of CTS suppression is not present in single-cell environments because, in such environments, every host can hear every other host. When a host finds the medium busy, it cannot send an RTS to begin with and the question of waiting for CTS does not arise. CTS suppression causes the RTS retransmit limit to be exceeded. This effect is shown in Figure 3.5.

Our channel quality estimation, which takes into account both channel capacity and loss rate, works as follows. Assume a flow attempts to transmit k packets in a time interval T. Assume that j of these attempts result in successful packet transmissions while $k-j$ packets are dropped in the MAC protocol, i.e., the sending MAC protocol, for whatever reason, does not receive the ACK frame that indicates sucessful completion of frame transmission. Assume that the flow has packet-size s at the MAC layer. The channel capacity perceived by flow f is calculated as $TP = \frac{j \cdot S}{\sum_{j}(t_r - t_s) + \sum_{k-j} T_w}$. The loss rate estimate at the MAC for this link is $L = 1 - \frac{j}{k}$. All flows f traversing this link use this value of L (one value per wireless link) as their loss rate estimate $L(f)$ for this link. Both channel capacity and loss rate estimates are computed every time interval T and running averages with history decay are kept. The normalization considerations remain the same with TP computed in this manner, i.e., TP_2 and $B_p(f)$ for flows f are the same.

3.7 Converting Bandwidth Requirements to CTP Requirements

We use the channel capacity $B_p(f)$ and loss rate $L(f)$ perceived by flow f, determined in the previous subsection, to convert its bandwidth requirements to its CTP requirements.

First, we incorporate loss rate to augment the bandwidth requirement. In general, to deal with packet losses, some redundancy needs to be built into the multimedia streaming protocol. In the worst case, this

redundancy is simply the re-transmission of the frame, or it could be something more sophisticated in the network stack such as forward error-correction (FEC). The amount of redundant data that needs to be sent is a function of the loss rate. From the point of view of resource reservation, the response to loss rate should be over-reservation of bandwidth resources so as to accommodate the redundancy. Thus, if a flow f needs minimum and maximum bandwidth $B_{min}(f)$ and $B_{max}(f)$ respectively, and it experiences a loss rate $L(f)$, then to accommodate redudancy, we must augment these requirements to $B'_{min}(f) = \zeta(B_{min}(f), L(f))$ and $B'_{max}(f) = \zeta(B_{max}(f), L(f))$. The ζ function represents the redundancy added to the protocol, for a particular bandwidth and loss rate. Thus, $B'_{min}(f) \geq B_{min}(f)$ and $B'_{max}(f) \geq B_{max}(f)$. In this work, we we simply assume that packet loss is compensated by re-transmission, i.e., $B'_{min}(f) = \zeta(B_{min}(f), L(f)) = \frac{B_{min}(f)}{1-L(f)}$. Similarly for $B'_{max}(f)$. Note that for multi-hop flows, the augmented bandwidth requirement for a particular hop must take into account *cumulative* loss rate over all subsequent hops, i.e., $L(f)$ for a particular hop is the product of loss rate for each hop starting from this one to the last one. If $L(f) = 1$ at any point, then the flow is obviously rejected since it cannot manage any throughput at all.

The next step is to convert these augmented bandwidth requirements $B'_{min}(f)$ and $B'_{max}(f)$ into *channel time proportion* (CTP) requirements.

We can illustrate the need for doing this conversion prior to channel bandwidth allocation. Assume that the delay $t_r - t_s$ from Figure 3.1 in transmitting a single MAC frame is such that 10 frames of a particular size S can be transmitted in a second over the user-AP link for user f. Assume that user f's minimum bandwidth requirement is 3 frames of size S per second. Then, user f requires at least 30% of the channel capacity. It needs to be active on the channel for 30% of unit time to meet its minimum bandwidth requirement. This leaves only 70% of unit time available to other users, which directly affects their admission. Although our example is in terms of frames per second of frame size S, we can extend this logic to bits per second also. If $B_p(f)$ for a flow of a certain frame size is h bps, and flow f's minimum throughput requirement is l bps, then in effect the user requires a fraction $\frac{l}{h}$ of unit time on the channel.

The CTP requirements of a flow f, $p_{min}(f)$ and $p_{max}(f)$, can thus be obtained by simply dividing its respective augmented bandwidth requirements $B'_{min}(f)$ and $B'_{max}(f)$ by its channel capacity estimate $B_p(f)$, after the estimate has been scaled appropriately for flow f's mean packet size. Thus, assuming $B_p(f)$ is the appropriately scaled value, $p_{min}(f) = \frac{B'_{min}(f)}{B_p(f)}$ and $p_{max}(f) = \frac{B'_{max}(f)}{B_p(f)}$.

In the case of a multi-cell environment, it may appear erroneous that 100% of the CTP in a particular

wireless cell is allocated to flows in that cell, when in fact, part of the channel is taken by interference flows from other cells. Upon examining the channel capacity estimation scheme closely, however, we note that the channel time consumed by the interference flows from other cells appears as part of the channel busy time sensed by flows in this particular cell. (See Figure 3.1.) This time is thus accounted for, in the CTP requests of the flows in this particular cell. When a flow makes a CTP request to the BM in its cell, it considers that it is going to be impeded by flows in other cells and takes this impedance into account when determining its request. We can thus allocate all 100% of the CTP in a particular wireless cell to flows in that cell itself, since the flows' respective CTP requests themselves take into account the fact that they are going to be impeded from transmitting by flows from neighboring cells.

3.8 Alternative Channel Capacity Estimation Techniques

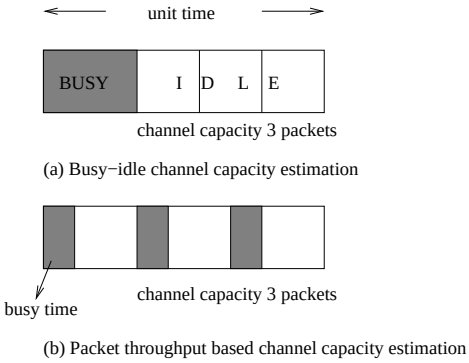


Figure 3.6: Alternate channel capacity estimation technique.

There exists related work in channel capacity estimation [17] in which a wireless network interface's average busy fraction of unit time is calculated and the rest (average idle time) is available for use. In contrast, we measure how portions of this compulsory busy fraction affect each transmission out of the interface. The network interface can be unavailable for several reasons: the IEEE 802.11 protocol's Network Allocation Vector (NAV) could be set, the protocol could be counting down a backoff, or time could be lost to a corrupt transmission. Measuring busy time is tedious because all these phenomena have to be tracked. We believe measuring how portions of the busy time affect individual transmissions is simpler. Figure 3.6 shows the conceptual equivalence of the two approaches. The available channel capacity in terms of packets per second of a certain packet size is the same with both approaches. In the busy-idle channel capacity

estimation scheme also, smaller packet size implies lower throughput since a greater percentage of the idle time is wasted in sending RTS, CTS, and ACK for the individual packets than in sending the actual payload.

In [18], a channel capacity estimation scheme is proposed which takes into account the traffic load on the network. Admission control using available bandwidth estimation is different when the network is unsaturated, semi-saturated, and saturated. A prediction mechanism is used to determine what would be the bandwidth perceived at the MAC layer of a node, in case a particular flow is admitted. This mitigates one of the problems of our scheme, which is that the bandwidth estimation is *reactive*. The estimate is derived from packets already transmitted, and is not predictive. Hence, the effect of a new flow cannot be known before the flow begins. However, as described in future chapters, our scheme provides for *re-negotiation*, so at any instant, if a node finds that its previous channel capacity estimate is outdated, it can re-negotiate using the new, more accurate, estimate. Thus if the arrival of a newly admitted flow dramatically alters the old channel capacity estimate, the channel allocations in the network can be updated to take this into account, via the dynamic re-negotiation.

The experimental results in [18] do not take into account the re-negotiation feature of our scheme, whereby incorrect channel capacity estimates are continuously corrected. Furthermore, the results show that our scheme results in a severe bandwidth shortfall, in comparison to the requirements for real-time flows. In fact, this is not a problem with the channel capacity estimation. We assume real-time flows also to have multiple operating qualities, and hence to be flexible up to a point. Our bandwidth allocation scheme (described in later chapters) takes advantage of this flexibility to revoke some bandwidth from real-time flows, degrading their quality upto a point, so that newer flows can be admitted and not starved. On the other hand, the authors of [18] assume that the real-time flows are not flexible at all, and are prepared to let best effort flows starve in order to meet the inflexible real-time requirements. The shortfall seen in the experiments in [18] is thus a result of the bandwidth allocation policy used, rather than a product of inaccurate channel capacity estimation.

Our channel capacity estimation coupled with dynamic re-negotiation, is accurate enough to enable us to implement any channel allocation policy, as shown our results in this and future sections.

The channel capacity estimation scheme of MPARC [18] involves communication between nodes in a neighborhood in order to compute the traffic load. This imposes an extra overhead, since the neighborhood involves not only 1-hop neighbors but also nodes outside the transmission range, but within interference

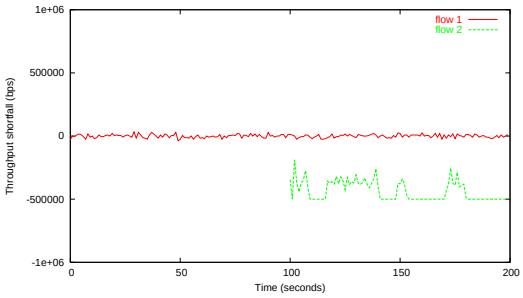
range of each other. Each node must update every node in a 3-hop radius of itself of its traffic load, so that overall traffic load in the neighborhood can be computed. Our scheme is simpler in that channel capacity estimation is done independently at each node, without any input from neighbors. The channel capacity estimation scheme of MPARC [18] also does not have any special provisions to deal with the CTS suppression (i.e. hidden flow effects) prevalent in multi-cell and multi-hop environments.

Figure 3.7: Scenario for channel estimation experiment in the presence of hidden flow effects.

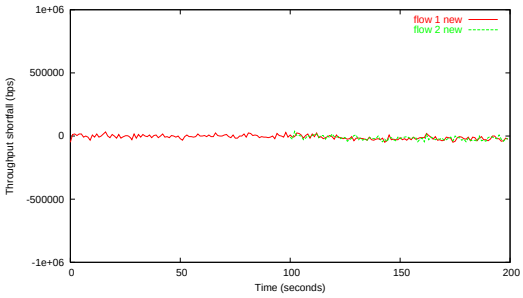
We present below experiments to show the working of our channel capacity estimation, and bandwidth allocation, scheme in a particular pathological hidden-flow scenario, as compared to the channel capacity estimation of MPARC [18]. The scenario is shown in Figure 3.7. Transmission range is 250 meters and interference range is 550 meters. We assume both flows have the same priority. Figure 3.8(a) shows the bandwidth shortfall experienced by each flow using the channel estimation of MPARC [18], when the requirement of each is 500 kbps. Since the effects of the flows on each other are not taken into account in the channel estimation, one flow grabs the channel and the other flow gets very little throughput. On the other hand, with our scheme (Figure 3.8(b)), the flows are not aware of each other but the effects they produce on each other in terms of CTS suppression, losses, and delays, is estimated and accounted for. A central BM (located at node 5 in Figure 3.7) allows each flow access to the channel taking its requirements and channel estimates into account.

Figures 3.8(c) and 3.8(d) show the results when the flow requirements are 800 kbps. Once again, with the hidden flow unaware channel estimation of MPARC [18], flow 2 experiences severe bandwidth shortfall. Our channel estimation takes both flows' effects on each other into account. Even though the estimated channel capacity is different for the two flows (see Figure 3.9) because of unpredictable hidden node channel effects, the bandwidth available to both flows is still the same. The channel estimate is better for the host that is able to grab the channel for itself more frequently, but our channel estimate accurately captures this, conveys it to the BM, and the BM makes sure this flow gets less access to the channel so the other can catch up. As stated earlier, our scheme assumes real-time flows are flexible upto a point.

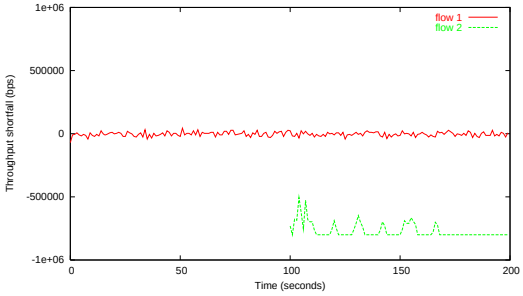
Figures 3.11(a) and 3.11(b) show the throughput shortfall when using the MPARC channel estimation and our scheme respectively, for a scenario without hidden terminal effects, shown in Figure 3.10. The



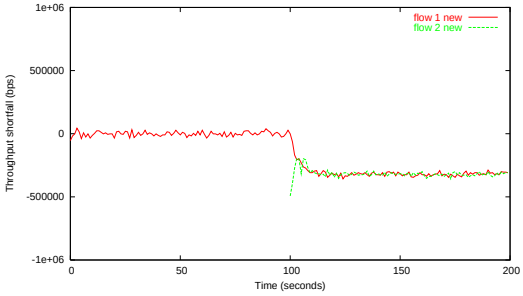
(a) Throughput shortfall when using [18]: 500 kbps requirement.



(b) Throughput shortfall when using our channel estimation: 500 kbps requirement.



(c) Throughput shortfall when using [18]: 800 kbps requirement.



(d) Throughput shortfall when using our channel estimation: 800 kbps requirement.

Figure 3.8: Comparative results in the presence of hidden flow.

requirements of the two flows are 500 kbps each in these figures. Figures 3.11(c) and 3.11(d) show the respective throughput shortfalls with two flows with 800 kbps requirements. The throughput stability achieved using our scheme is much better.

Note that in this chapter, we have only experimented with real-time, CBR flows. In subsequent chapters we also present experimental results with best-effort TCP traffic.

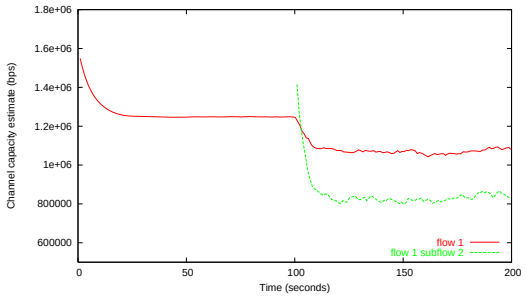


Figure 3.9: Channel capacity estimates using our scheme in the presence of hidden flow: 800 kbps requirement.

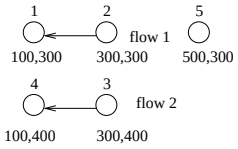
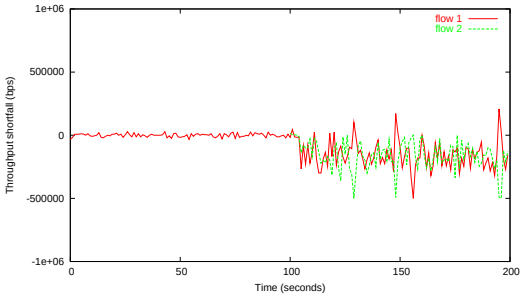
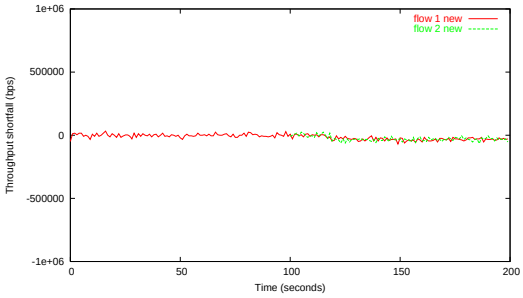


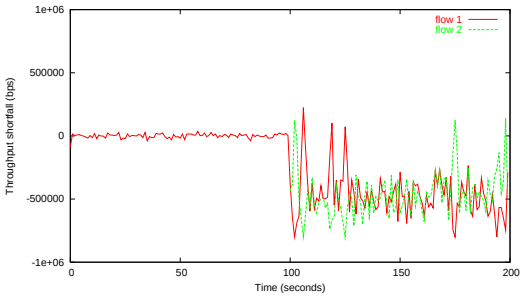
Figure 3.10: Scenario for channel estimation experiment in the absence of hidden flow effects.



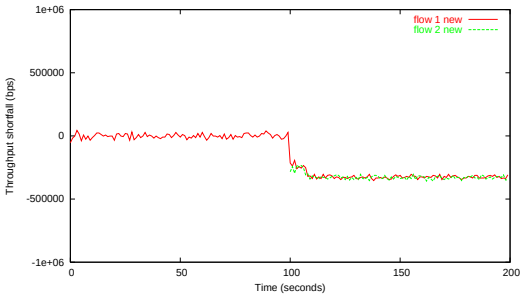
(a) Throughput shortfall when using [18]: 500 kbps requirement.



(b) Throughput shortfall when using our channel estimation: 500 kbps requirement.



(c) Throughput shortfall when using [18]: 800 kbps requirement.



(d) Throughput shortfall when using our channel estimation: 800 kbps requirement.

Figure 3.11: Comparative results in the absence of hidden flow.

Chapter 4

Dynamic Bandwidth Management Architecture and Protocol

4.1 Bandwidth Management System Architecture

The architecture of the bandwidth management system consists of three major components as shown in Figure 4.1: (a) the Rate Adaptor (RA) at the middleware layer, (b) the per-node Channel Capacity Estimator (CQE) at the MAC-layer and (c) the Bandwidth Manager (BM), which is unique in the entire single-hop wireless subnet or cell. Our system takes advantage of *cross-layer interaction* between the application/middleware and link layers.

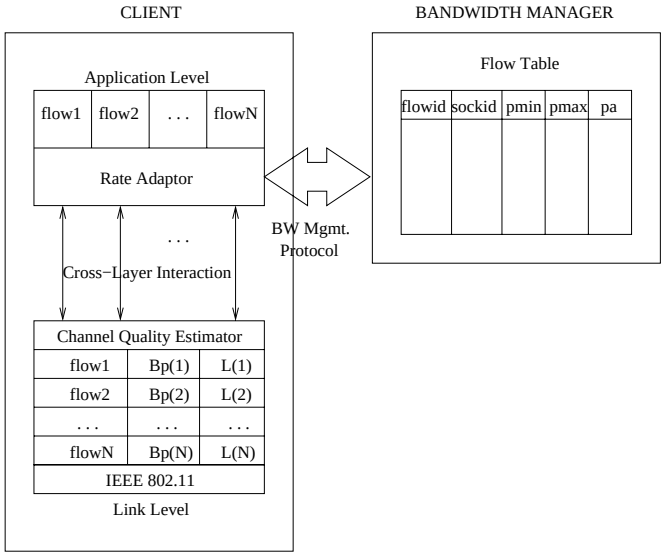


Figure 4.1: Bandwidth Management System Architecture.

4.1.1 Rate Adaptor (RA)

In our design, we assume the absence of fair scheduler at the MAC layer. Hence, a flow's bandwidth consumption in accordance with its allotted CTP is regulated only by the Rate Adaptor (RA). The RA converts a flow's bandwidth requirements into CTP requirements, communicates this to the BM, and obtains an allocated CTP for this flow from the BM. It then controls the transmission rate of each flow depending on its allotted CTP, using a traffic control method like leaky bucket [1] For our simulation experiments, we implement the RA, which includes a leaky-bucket algorithm for rate-control, at the network interface queue. It thus functions above the network interface, but below the application. In real implementations, the traffic control can be performed by loadable kernel modules under Linux. The RA periodically determines a flow's allocated rate from the BM and updates its leaky-bucket with this rate. There exist queue-based rate-control schemes of various disciplines employing kernel-level queues configured through user-level programs, that can be used in practice [19]. These queues throttle traffic in accordance with configured flow rates at the time of dequeuing to the network interface.

In case fair scheduling is present at the MAC layer, shaping the traffic and enforcing flow rates can be left to it. The RA's function is deprecated to merely performing the bandwidth requirements to CTP requirements conversion, communicating with the BM, and determining the allocated CTP for each flow. Once it determines the allocated CTP for a flow, it must pass this down to the scheduler at the MAC layer as the weight of this flow.

4.1.2 Channel Quality Estimator (CQE)

We described the channel quality estimation mechanism in detail in the previous chapter. A CQE is co-located with the IEEE 802.11 MAC protocol of each host. It *continuously* estimates the channel capacity $B_p(f)$ and loss rate $L(f)$ for each flow f originating at the host it is situated on. It periodically passes this up to the RA. The RA uses it in the translation of flow f's bandwidth requirements to its CTP requirements. When the channel quality $B_p(f)$ or $L(f)$ perceived by flow f changes, the channel time requirements calculated using $B_p(f)$ also change. The CQE informs the RA of the new $B_p(f)$ or $L(f)$. The RA may now need to re-negotiate on behalf of flow f with the BM, using flow f's new CTP requirements that are

[1] For the sake of simplicity, in our testbed experiments, the RA is built into the UDP application itself, to adapt its data generation rate.

calculated with the new $B_p(f)$ and $L(f)$ estimates. Since CTP allotted to flow f is directly related to its share of total network bandwidth, if a flow perceives the channel capacity as having decreased, its share of the bandwidth will also decrease. This may cause it to fall substantially below its minimum bandwidth requirements. Hence the re-negotiation. We do not wish to re-negotiate for small changes in $B_p(f)$ or $L(f)$, however, in order to keep re-negotiation overhead small. The RA's not reacting to small changes in $B_p(f)$ may thus cause small violations of the minimum bandwidth requirements. (But not minimum CTP requirements.) The moment a large violation occurs, the RA immediately reacts and re-negotiates. The parameter that defines "small" and "large" is tunable. It trades off the hardness of the bandwidth guarantee with re-negotiation overhead.

In our implementation, the CQE is implemented under RedHat Linux release 7.1, kernel version 2.4.2-2, in the driver of the IEEE 802.11 interface (*/lib/modules/2.4.2-2/kernel/drivers/net/pcmcia/wvlan_cs.o*). It communicates with the RA at the application layer through the */proc* interface.

Example: Assume a flow f in a 2 Mbps wireless network has minimum bandwidth requirement 300 Kbps and perceives total network bandwidth of 1.5 Mbps. (That is, the flow f perceives this to be the total capacity of the 2 Mbps channel.) Assume further that the CTP allotted to it is 20%, thus ensuring it just meets its minimum bandwidth requirement. If the total network bandwidth, as perceived by f, decreases to 1.2 Mbps due to an increase in physical channel errors or contention, then the 20% channel time is no longer sufficient for the flow to meet its minimum bandwidth requirement. Its RA must then re-negotiate for at least a 25% of the channel time. Similarly, if a flow perceives the total network bandwidth to have increased, it must release any excess share of the channel it has been allotted, so that some other flow can use it.

4.1.3 Bandwidth Manager (BM)

There is a Bandwidth Manager (BM) to perform channel time arbitration for every wireless subnet or cell. The BM performs channel time allocation at the time of flow establishment and channel time redistribution at the time of flow teardown. CTP allocation to a new flow at the time of establishment may involve revocation of some channel time from existing flows and re-allocation of this portion to the new flow. The BM also performs re-negotiation either when some flow detects a change in its perceived bandwidth or when its traffic

characteristics change [2].

The BM may employ a fair or price-based algorithm for CTP allocation. In either case, the BM admits a flow only if it can allot *at least* its minimum CTP requirement. Otherwise, the flow is rejected. If a fair allocation policy is used, the remaining channel time as yet unallotted after all the admitted flows' minimum channel time requirements are satisfied, is allotted on a *max-min fair* basis. We therefore deem this channel time allocation scheme *max-min fair with minimum guarantee*. Each flow receives whatever CTP is allotted to it by the max-min fair algorithm, in addition to its minimum CTP request, which is automatically guaranteed when it is admitted. A detailed description of the max-min fair with minimum guarantees algorithm can be found in Chapter 5.

In the price-based CTP allocation policy, flows are allocated CTP based on the price they are willing to pay. The allocation policy centrally simulates a distributed, ascending, multi-unit, proportional auction. (See Chapter 5.) The CTP allocations determined by the algorithm are the same as those that would be determined by the auction, when bidding ceases. If the allocation determined is such that a flow's minimum CTP requirement cannot be satisfied, then this flow is allocated zero CTP and is rejected altogether. The remaining flows then once again participate in another auction. This continues until all admitted flows can have at least their minimum CTP requirements fulfilled.

Both CTP allocation policies provide all flows some minimum, channel-conditioned, throughput guarantee. This is particularly important to a multimedia flow because it requires some minimum QoS to be useful. Audio below a certain bit-rate is unintelligible and video below a certain frame-rate may be unacceptable. If the minimum quality of operation of a certain multimedia flow cannot be satisfied, then there is no point in allocating any throughput to it. This throughput may as well be allocated to other flows.

4.1.4 Deployment

In practice, both the software implementing the RA [19] and the IEEE 802.11 driver containing the CQE can be downloaded and installed by a user when he/she joins the wireless network. In current AP-based 802.11 wireless networks, authorized users are often required to download and install Virtual Private Network (VPN) clients when they join the network, for security reasons. The traffic control and channel quality estimation modules can also be similarly downloaded and installed. The BM can also be easily installed as

[2]The centralized BM does not take on the onus of channel bandwidth estimation, and leaves this to the individual per-node CQEs, because the available channel quality is different for different peer-to-peer flows, due to location-dependent physical errors.

it is a simple user-level program that listens on a well-known port for requests from the client RAs. Our bandwidth management architecture is thus highly deployable because it requires no major change to the current off-the-shelf hardware and software. The LinuxTC traffic control library [19] provides a suite of traffic control options such as leaky bucket and token bucket. It can be installed as a downloadable kernel module and configured using user-space scripts.

4.2 Bandwidth Management Protocol

This section describes the protocol used in the interactions between the various components of the bandwidth management architecture and the details of the BM's operation. The BM is invoked at the time of flow establishment, flow teardown, significant change in a flow's perception of channel quality, or significant change in a flow's traffic pattern. Figure 4.2 shows the actions that occur when these events happen.

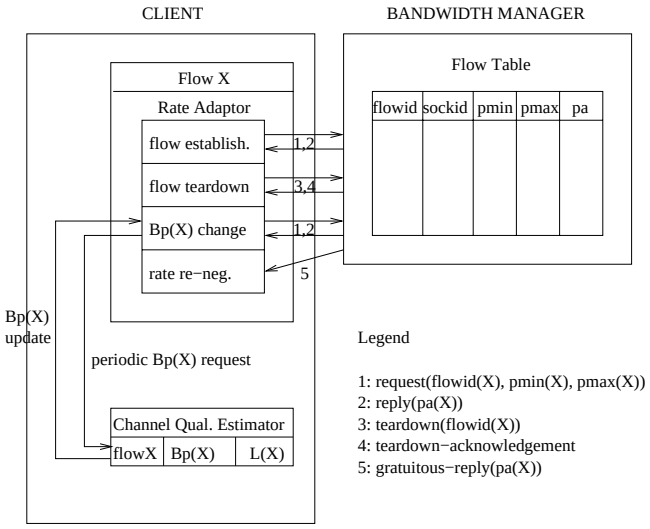


Figure 4.2: Bandwidth Management Protocol.

4.2.1 Flow Establishment

At the time of initiating a flow f, the application specifies its required minimum bandwidth $B_{min}(f)$ and maximum bandwidth $B_{max}(f)$, both in bits per second, to its RA. These values have to be first modified to $B'_{min}(f)$ and $B'_{max}(f)$, taking into account the loss rate estimate $L(f)$. Then each of these modified requirements is divided by the flow f's perceived channel capacity $B_p(f)$ to obtain its requested minimum

and maximum CTPs, $p_{min}(f)$ and $p_{max}(f)$, respectively. The total network bandwidth $B_p(f)$ perceived by a flow f is estimated by the CQE at the local node. A best-effort flow will have $B_{min}(f) = 0$.

Note that both the CTP consumed by the flow f's data packets in the forward direction as well as CTP consumed by the acknowledgements in the reverse direction, if any, must be included in f's CTP requirement. As mentioned in the previous chapter, bi-directional flows at the higher layers (e.g. TCP) must be considered as two uni-directional flows for the purposes of channel quality estimation. Channel quality should be estimated for the acknowledgement flow (i.e. acknowledgements belonging to the higher layer bi-directional protocol) at the destination must be transmitted in-band to the source. CTP computation for the flow as a whole must be done at the source taking into account the CTP requirement of both the higher-layer data packets, computed using $B_p(f)$ estimated at the source, and that of acknowledgement packets, computed using $B_p(f)$ estimated at the destination. However, the acknowledgement flow requires only fraction of the channel compared to the data flow. Hence, for simplicity, we add a small overhead to the CTP computed for the data flow, instead of precisely computing the acknowledgement flow's CTP.

The CQE returns the perceived channel capacity $B_p(f)$, for flow f, *normalized* to a standard packet size. It must be appropriately *scaled* for different packet sizes, using the reverse of the normalization procedure, before being in bandwidth to CTP conversion. For VBR-UDP flows, either the mean packet size can be used or the VBR flow can be split into CBR components, as described later in this section. For TCP flows, separate $B_p(f)$ values can be derived for data and acknowledgement packets from the single normalized value returned by the CQE, since the data and acknowledgement packets are of different sizes.

It must be kept in mind that the CQE measures the perceived bandwidth $B_p(f)$ using MAC layer frames of flow f. These MAC layer frames include protocol headers from the intermediate layers of the protocol stack between the application and the link layers. The $B_p(f)$ scaling operation must take into account the fact that the lower layers of the protocol stack will add their respective headers to each packet, and thus consume some of the channel capacity. The size of the lower-layer headers must thus be added to the application packet size in the scaling operation.

The RA at a mobile host registers a new flow with the host's CQE. Initially, the CQE has no estimate of the channel quality as perceived by this newly beginning flow. This is because it has to use the flow's packets themselves for obtaining an estimate of the channel quality, based on the physical channel errors and contention these packets experience. But the flow has not sent out any packets yet and is still in the process

of establishment. So, when initially computing the flow's requested minimum and maximum CTPs, the RA has to use a hardcoded initial channel quality estimate [3]. Once the flow begins, a more accurate channel quality estimate will be available from the CQE. The requested minimum and maximum CTPs can then be modified using this new, more accurate estimate, and re-negotiation done with these modified values.

Alternatively, in the case of a connection-oriented flow, the first few flow-establishing packets can be used in the channel quality estimation instead of the hardcoded estimate. For example, the physical channel errors and contention faced by TCP's three-way handshake messages can be used in the initial measurement. If the application involves some other control messages (e.g. client asking server if file exists or not), then these can be used. A current estimate being used by *other* flows between the same end-points can also be used initially. A fourth option is to have the BM maintain a list of current channel quality estimates for all flows. Then, a new flow can query the BM for an initial estimate. The BM simply returns the average of the list of all flows' channel quality estimates.

Let the initial channel quality estimate, how ever it may be obtained, for a new flow f be $B_p(f)$. The CTP $p_{min}(f)$, required to satisfy the new flow f's augmented minimum bandwidth requirement $B'_{min}(f) = \frac{B_{min}(f)}{1-L(f)}$, is $p_{min}(f) = \frac{B'_{min}(f)}{B_p(f)}$. $p_{min}(f)$ = 0 for best-effort flows. Similarly, the CTP $p_{max}(f)$, required to satisfy flow f's augmented maximum bandwidth requirement, is $p_{max}(f) = \frac{B'_{max}(f)}{B_p(f)}$. The RA of the new flow f sends the BM a `request` message containing the flow-id of f, $p_{min}(f)$, $p_{max}(f)$ and a timestamp for ordering.

The BM contains a channel time allocation algorithm that decides whether the new flow f is admitted or not, and if it is admitted, what is its allocated CTP $p_a(f)$. In this work, we present two channel allocation algorithms: a max-min fair with minimum guarantees algorithm and a price-based algorithm. Both these algorithms are presented in Chapter 5.

If the flow f is admitted, the BM adds it to its set of admitted flows F and allocates a certain CTP $p_a(f) \in [p_{min}(f), p_{max}(f)]$ to it. If flow f is rejected, then it is not added to F and its allocated CTP $p_a(f) = 0\%$.

After the new flow f is admitted, the BM registers an entry pertaining to it in its *flow table*. This entry consists of: (a) the new flow f's flow-id, (b) the socket descriptor of the socket used by the BM for communication with f's RA, (c) $p_{min}(f)$, (d) $p_{max}(f)$ and (e) $p_a(f)$. The socket descriptor is stored in the

[3] In our simulation experiments, we use a 2 Mbps network and we set this hardcoded value to 1.5 Mbps.

table so that if any re-negotiation needs to be done later with flow f's RA (for example, when newer flows arrive in future or existing flows depart), this socket can be used. In addition, a timestamp indicating the freshness of the latest `request` message is also maintained for each flow. This timestamp is used for two purposes: (a) timing out stale reservations, and (b) proper ordering of multiple outstanding re-negotiation requests from the same flow. Since reservations can time-out, the entries in the flow table are *soft-state* entries. If, for some reason, a flow's reservation has timed-out but the flow is still transmitting, this can be detected using a *policing* mechanism. (See Section 4.3.)

Finally, for every flow $g \in F$, the allotted CTP $p_a(g)$ is then sent to flow g's RA using a `reply` message. (Note that the name of the message is a misnomer in the case of all flows $g \in F$ except the new flow f because, in their case, the `reply` is gratuitous, not a response to any message they sent. The `reply` message indicates how the allocations of flows $g \in F$ change when the new flow f is admitted.) It may be the case that all flows $g \in F$ do not need to be sent a `reply` message. No `reply` message needs to be sent to a flow in F whose allotted CTP has not changed due to the arrival of the new flow f. The `reply` messages to individual flows' RAs, mentioning their new allocated CTP values upon the new flow f's admission, can be implemented as multiple unicast messages for reliability, or alternatively, as a single broadcast `reply` message for efficiency. The broadcast `reply` contains flow-id and $p_a(g), \forall g \in F$. If for a large number of flows in $g \in F$, the CTP allocated is not affected by the arrival of a new flow f, unicast `reply` messages may be preferable. Also, in a multi-cell environment, where a broadcast will be propagated over all the cells, unicase `reply` messages may be preferable. A flow f is rejected using a unicast `reply` with $p_a(f) = 0$. Other existing flows' allotted CTPs are not affected.

The RA of every flow that receives a `reply` message from the BM, gratuitous or otherwise, sets the transmission rate of its leaky bucket respectively to $p_a(g) \times B_p(g)$ bits per second (bps). Once again, the $B_p(g)$ value is appropriately scaled for packet-size variations before being used in this calculation. The new flow f can now begin operation whereas the older flows simply resume operation with their respective new rates.

The timestamp in the `reply` to flow g indicates the last `request` received from g by the BM. The value of $B_p(g)$ used to compute $p_{min}(g)$ and $p_{max}(g)$ for this `request` must then be used in the transmission rate formula above, since it is based on this value of $B_p(g)$ that $p_a(g)$ was calculated by the BM. As a new $B_p(g)$ is returned by the TBE periodically, a new rate is also used periodically. If the $B_p(g)$ or $L(g)$ change

is large since the last period, re-negotiation must occur, as explained below.

4.2.2 Flow Teardown

When a flow f terminates, its RA sends a `teardown` message to the BM. The BM removes flow f from the set of existing flows F i.e. $F = F - \{f\}$. It then redistributes flow f's allotted CTP $p_a(f)$ among the other flows using the max-min fair algorithm with minimum guarantees or the price-based CTP allocation algorithm. The RA of each flow $g \in F$ (the new set F) is told of its newly allotted CTP by the BM. The socket descriptors in the flow table are used to send gratuitous `reply` messages for this purpose. The entry for the terminating flow f in the BM's flow table is expunged. A `teardown-acknowledgement` message is sent to f's RA.

4.2.3 Change in a Flow's Perception of Total Network Bandwidth

The RA of every flow periodically obtains from the CQE the flow's current perceived channel quality. The CQE updates the RA with the mean of the transmission throughput measured for each packet successfully transmitted by the flow in recent history. The inter-update period could be in terms of number of packets transmitted or in terms of time. We recommend using a hybrid scheme for determining update period: it should be based on time when the transmission rate of the flow is low and based on number of packets transmitted when it is high. In our experiments, we use high transmission rates in order to determine the performance of our scheme under high network loads. Therefore, we use a perceived bandwidth update interval based on number of packets. We use a default interval of 100 transmitted packets in our experiments, but we also measure how various other intervals affect the performance of the system.

In case a newly obtained perceived channel capacity value $NEWB_p(f)$ differs significantly from $B_p(f)$ or a new loss rate estimate $NEWL(f)$ differs significantly from $L(f)$, the RA must re-negotiate its flow's CTP with the BM, as indicated in the example in the previous section. It must also set the value of perceived channel capacity $B_p(f)$ to the newly obtained value $NEWB_p(f)$, and the value of loss rate estimate $L(f)$ to $NEWL(f)$. Note that the RA only sets $B_p(f)$ and $L(f)$ to the corresponding new values and re-negotiates with the BM using these new values when there is a significant change, not with every update. A new rate using the previously allotted CTP is however calculated with every update. In our experiments, we assume a deviation δ = 15% of $NEWB_p(f)$ from $B_p(f)$ or of $NEWL(f)$ from $L(f)$ as significant enough

to warrant re-negotiation. We also measure how other channel quality deviation tolerance (δ) percentages affect system performance.

If re-negotiation has to be done, the RA of flow f sends a `request` message to the BM with flow-id, $p_{min}(f)$ and $p_{max}(f)$. The values of $p_{min}(f)$ and $p_{max}(f)$ sent in the `request` message are re-calculated using the new values of $B_p(f)$ and $L(f)$. The rest of the re-negotiation procedure is almost identical to the one used for flow establishment, both at the BM as well as at the RA. (See Figure 4.2.) The only difference is that the BM does not have to add a new entry in its flow table for f; it only updates the already existing one.

Note that a flow f's re-negotiation request can be rejected by the BM, i.e., it can receive $p_0(f) = 0$, in response to the requested CTP. This means that the flow has been cut-off in mid-operation. Unfortunately, the nature of the wireless network is inherently unreliable and as network resources decrease, some flows will necessarily have to be cut-off in mid-operation so that others can be supported. Our scheme guarantees that each flow will obtain at least its minimum requested CTP for almost 100% of its active duration. If the system cannot guarantee the flow at least this level of QoS, it will drop it altogether. In other words, a flow will either receive (for nearly 100% of its active duration) at least its minimum requested CTP $p_{min}(f)$, or it will receive no channel time at all. The guarantee in terms of bandwidth is that the allotted bandwidth never falls more than a factor of δ below the minimum requested bandwidth $B_{min}(f)$, since if $B_p(f)$ or $L(f)$ change by a factor of δ, re-negotiation occurs.

Currently, we do not use any priority scheme to cut-off particular flows. If perceived bandwidth decreases for all flows, the first flow initiating re-negotiation is cut-off. Alternate flow dropping strategies are discussed briefly in Chapter 6.

It must be noted that the admission of a new high-bandwidth flow will increase contention for all existing flows, causing their $B_p(g)$ estimates to fall. There is thus likely to be a flood of re-negotiation requests for more CTP, upon the admission of a new high-bandwidth flow. The admission control at the time of flow establishment is inherently inaccurate because it is very difficult to gauge beforehand the effect, due to medium contention, that a new high-bandwidth data flow will have on the channel capacity estimates of previously existing flows. When a new flow is rejected by the admission control scheme, it means that the network can definitely not support its addition to the flow set. On the other hand, the successful admission of a new flow only means that under current channel conditions, its addition to the flow set can be supported.

Its admission may however alter existing channel quality, thus invalidating the channel conditions under which it was admitted.

We may admit a high-bandwidth flow based on its CTP requirements, but once it begins operation, it may increase contention and cause a substantial decrease in available channel capacity of the other flows sharing the channel. However, these flows can then request more CTP to compensate for the drop in their $B_p(g)$ values. Hence, the problem of inaccurate admissions in the BM scheme is thus solved by having dynamic re-negotiation in the presence of changed conditions, which were brought on by inaccurate admission control. In general, in wireless networks, one-time admission control is insufficient. Conditions are dynamically varying, so a provision must be present to allow flows to modify reservations through dynamic re-negotiations.

The flood of re-negotiation requests for more CTP that follows the admission of a new high-bandwidth flow might also require some flows to be cut-off mid-operation. Cutting-off flows mid-operation is thus a problem that can be expected to crop up relatively frequently. We find that a price-based strategy such as the one presented in Chapter 5 is particulary useful in choosing a flow to be terminated. The flow that needs its channel time least desperately, i.e. the one willing to pay the least for it, should be the one likeliest to be dropped. This is one important motivation behind the price-based channel time allocation algorithm.

4.2.4 Change in a Flow's Traffic Characteristics

When a VBR-UDP flow f (e.g. MPEG video stream) needs to send a burst of traffic at a rate different from its normal rate, it must inform its RA. The RA will re-negotiate for a larger CTP for flow f depending on the bandwidth of the burst. The re-negotiation procedure is the same as in the case of change in perceived bandwidth. At the end of the burst duration, the RA will again re-negotiate to release the excess CTP. This solution is equivalent to splitting up a VBR stream in the time domain into multiple CBR streams. There exists previous literature in the context of ATM networks [20] in which VBR streams are split into multiple CBR streams in the time domain. Since this scheme only involves re-organizing the traffic rather than the network, it can be directly applied from ATM networks to wireless networks. Basically, if the mean throughput requirement of a flow varies dynamically, it can inform its RA of the changes and the RA can re-negotiate accordingly for CTP with the BM.

Figure 4.3 is an MPEG-4 trace of an hour-long, 25 frames per second, medium-quality, clip of the movie

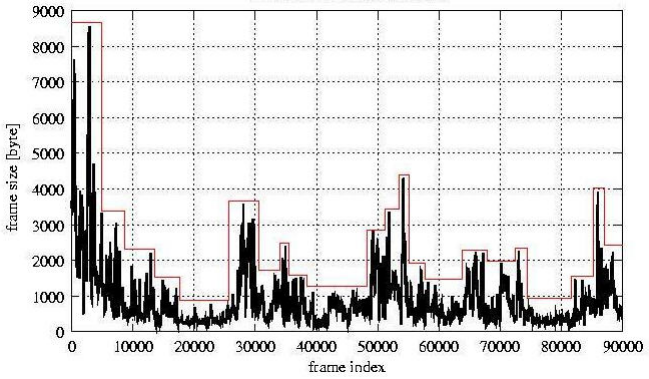


Figure 4.3: MPEG-4 trace of “Silence of the Lambs” clip with corresponding CBR components.

“Silence of the Lambs.” The trace was taken from [21] and the references therein. On the x-axis is a running count of the frame number. On the y-axis is the frame size averaged over non-overlapping blocks of 50 frames. One possible way to split up this VBR flow into multiple CBR components is shown in Figure 4.3 as the contour of the plot. The CBR bandwidth component thus obtained is then used as the *minimum* bandwidth requirement $B_{min}(f)$ in negotiating with the BM.

Frequent bursts could result in an explosion in re-negotiation overhead. We deal with the problem of frequent bursts in one of two ways: (a) setting $B_{min}(f)$, at the time of burst-induced re-negotiation, large enough to engulf multiple bursts and (b) having large buffering at the receiver to deal with the burst.

4.3 Policing

Traditionally, in networking literature, policing refers to the task of rate control to ensure that users shape their traffic conform to their allocated rate. This is usually achieved via a leaky bucket or a token bucket scheme. In our bandwidth management architecture, we have traffic shaping via leaky bucket [19], and this module is downloaded and installed as part of the RA when the user enters the network. (The driver augmented with channel quality estimation is also downloaded and installed at this time.) We call all flows that correctly control their rates with the help of the leaky bucket traffic shaper as “managed” or ”co-operative.” However, this managed or co-operative rate control is only one part of the policing scheme. In case flows refuse to control their rates (i.e. remove the leaky bucket function), either maliciously or in ignorance of the existence of the BM, they must be *detected* and removed from the network, so that they do not adversely affect the managed flows. We call the flows that do not have traffic control as “unmanaged” flows. The

co-operative rate control has been mentioned in detail earlier, when we described the RA. In this section, we specifically deal with the non-co-operative case, i.e., detection of unmanaged flows.

The BM can operate in two modes: normal and policing. In the *policing mode*, the BM listens promiscuously to the network traffic, and checks whether any registered flow is consuming a larger share than it has been allocated. A flow is identified by the source and destination addresses and port numbers in its packet headers. Additionally, the BM can also catch those flows in the network that are not registered at all with it. This can be some type of "denial of service" attack by a malicious users, or caused by some unmanaged applications.

Operating in policing mode is expensive. Therefore, the bandwidth manager should operate in this mode only when necessary. To this end, the bandwidth manager relies on the sudden, sharp decrease of channel bandwidth as an indication, in the re-negotiation process. If there is a sudden flock of re-negotiation requests due to reduction in $B_p(g)$, it is likely that abnormally high channel contention has occurred. Subsequently, the bandwidth manager switches into policing mode to monitor the activity of the network. It may be that the channel contention is due to a sudden increase in contention produced by the arrival of a new high-bandwidth flow. Alternatively, it may be that it is due to a malicious or unmanaged flow. The policing scheme can identify which of the above is the cause - the former is a normal condition whereas the latter is an anomalous condition. It could also happen that the unreliable broadcast `reply` message did not reach a particular RA, so a flow is continuing to transmit packets faster than its re-allotted rate.

Figures 4.4 and 4.5 illustrate the need, and the opportunity, to switching into policing mode: they show that when an unmanaged flow starts to transmit, the quality of the managed flows worsens. Furthermore, the managed flows can detect the interference in the network by the drop in their channel capacity estimates.

We assume 3 TCP flows in a single-hop, single-cell scenario, the last arriving one being a malicious or unmanaged flow. The unmanaged flow does not have a leaky bucket whose rate is controlled via feedback from the BM. It simply pumps data into the network as it is generated. Unlike the managed flows, this flow does not register with the BM upon startup and does not co-operatively control its rate. We can see from Figure 4.4 that it causes unfairness in the network and adversely affects the managed flows. (Compare with the correct operation illustrated Figures in 4.6 and 4.7, where the third flow is also managed.) Figure 4.5 shows how channel capacity drops sharply after 56 seconds, at which point the unmanaged flow arrives. This steep drop causes rapid re-negotiations with the BM. The BM must switch into policing mode at this

point and detect the unmanaged flow. It must then deal with the unmanaged flow offline, i.e., the system administrator must mention to the user initiating the non-co-operative flow that it is beneficial to everyone to have a co-operative environment. The non-co-operative flow can also be asked to use a different frequency, if the user does not wish to participate in the bandwidth management.

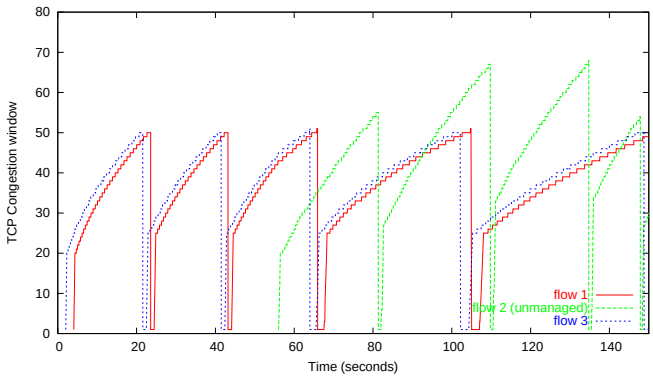


Figure 4.4: Effect of unmanaged TCP flow on managed TCP flows.

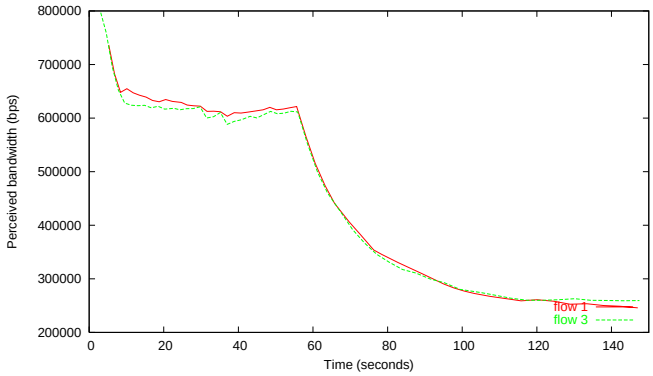


Figure 4.5: Effect of unmanaged TCP flow on perceived channel capacity of managed flows.

It could happen that the unmanaged flow *cannot* be stopped via offline mechanisms. In this case, it has to be assumed as co-channel interference causing variations in channel quality estimated. (See again Figure 4.4.) Fairness between the managed flows continues to hold in such a case. One option to deal with the scenario where it is impossible to stop unmanaged flows via offline mechanisms is to have the managed flows become more aggressive and try to suppress the unmanaged flows. However, *this is not a bandwidth-management problem*. Figure 4.8 explains why: it shows that even in when flows are allowed to compete freely, the unmanaged flow is not suppressed. In this plot, the two flows arriving first are *also* allowed to

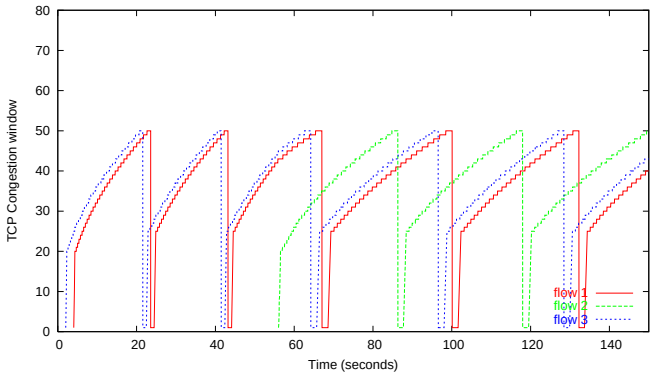


Figure 4.6: All flows managed.

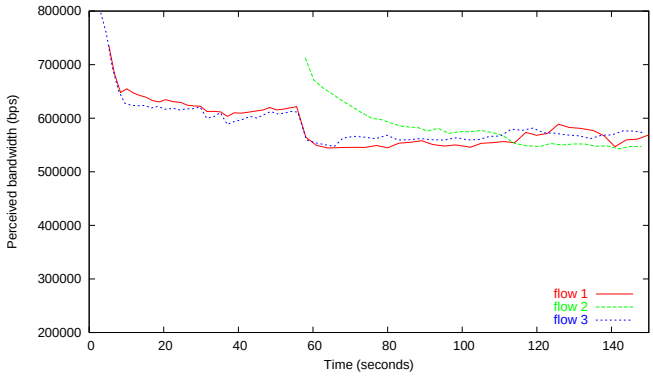


Figure 4.7: Perceived channel capacity when all flows are managed.

transmit as fast as possible, without any rate control. (In other words, they are also unmanaged.) All three flows are thus allowed to compete freely; the first two flows are given a chance to suppress the unmanaged flow. However, even in this case, when the first two flows transmit as fast as possible, the third flow (rogue flow) cannot be suppressed.

Thus, whether co-operative rate control and bandwidth management are present or absent, the rogue flow cannot be suppressed. Figure 4.9 illustrates the same point using a combination of UDP and TCP traffic. The two UDP flows are managed, while the TCP flow is unmanaged. The UDP flows each have minimum bandwidth requirement $B_{min}(f)$ = 0 and maximum bandwidth requirement $B_{max}(f)$ = 200 kbps. At equilibrium, the TCP flow grabs channel time from the UDP flows. The bandwidth management ensures fairness and throughput stability among the co-operative flows. Figure 4.10 shows that even in the absence of bandwidth management, when competition is free, the unmanaged TCP flow cannot be suppressed. Elimination

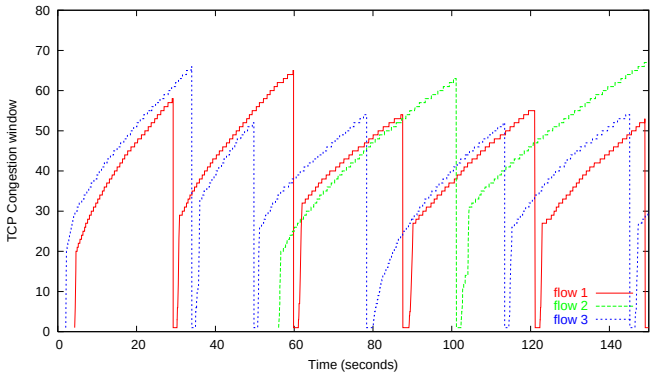


Figure 4.8: All flows unmanaged.

of a rogue flow is hence a problem independent of bandwidth management: the rogue flow's presence causes undesirable behavior whether the other flows are bandwidth-managed or they are not, whether they are TCP or they are UDP.

It is because of this rationale that we decided on offline mechanisms to eliminate unmanaged flows. As shown in Figure 4.5, our channel capacity metric and centralized management afford us the opportunity to design an elegant, reactive, policing strategy. Another possible online mechanism to reject an unmanaged sending flow is to have the MAC protocol at the receiver not send it CTS frames. Thus, it will be automatically receive no service. This assumes co-operation from the receiver, however, and also involves changing the MAC protocol.

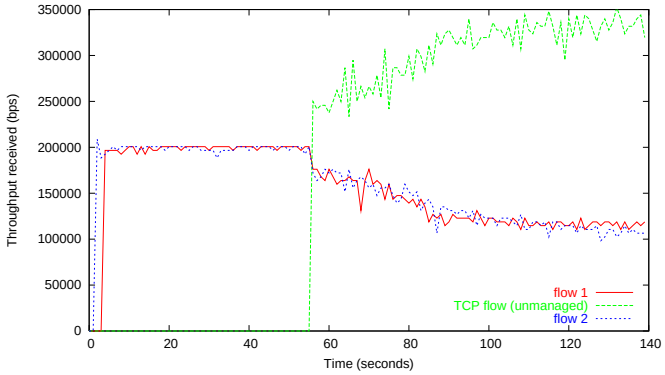


Figure 4.9: Effect of unmanaged TCP flow on managed UDP flows.

In general, in a bandwidth management scheme, as long as there is co-operative rate control, any type of bandwidth allocation policy can be implemented. In [18], the authors use a priority based bandwidth

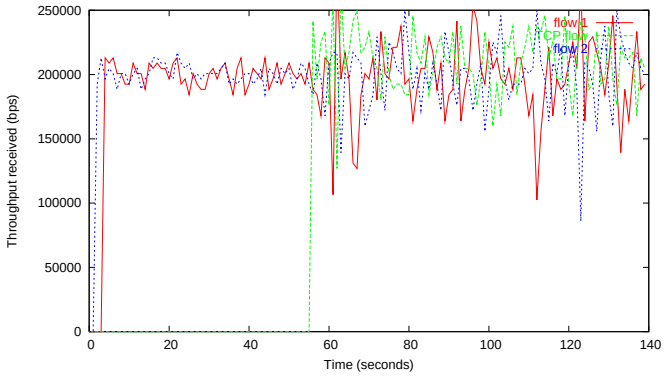


Figure 4.10: All flows unmanaged: mixed UDP and TCP traffic.

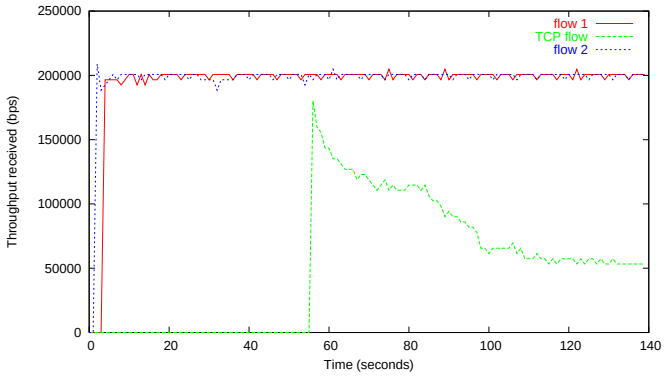


Figure 4.11: UDP real-time traffic getting priority over TCP best-effort traffic.

allocation policy where real-time flows get priority over best-effort traffic. Thus, the maximum requirement of the real-time flows is satisfied before best-effort traffic gets any service. Here, both real-time and best-effort traffic are assumed to be co-operative (managed). We can also implement this priority-based policy using our architecture, if we assume full co-operation of all flows. This is illustrated in Figure 4.11. In the subsequent chapters we discuss both fair and price-based (which is equivalent to priority-based) bandwidth allocation policies, in a co-operative environment.

Chapter 5

Channel Time Allocation Policies

One of the major contributions of this work is the design of channel time allocation strategies for flows with different requirements in a dynamic single-hop IEEE 802.11 network environment. We pay particular attention to providing some type of minimum throughput guarantee to real-time multimedia flows so that they can at least minimally function at their lowest acceptable quality setting. Flows whose minimum CTP requirements could not be supported were thus allocated no CTP at all. Consequently, the question arose as to how to distribute the remainder of the channel time after all admitted flows' minimum CTP requirements had been met. We decided upon sharing this remaining channel time fairly among the flows, taking into account their CTP request over and above the minimum requirement. We adopted a *max-min fair* [22] sharing model for this remaining channel time. We thus developed the *max-min fair with minimum guarantees* channel time allocation policy.

In a dynamic wireless network setting, the channel quality perceived by each flow varied continuously with time. Thus, in order for flows to be able to sustain their QoS, re-negotiation for channel time became necessary when the channel quality changed dramatically. Such a re-negotiation request in response to changed channel conditions necessitated a fresh round of channel time allocation. During this channel time re-allocation, due to the changed channel conditions, some flows were required to be blocked in mid-operation so that other flows' minimum CTP requirements could be supported. The question arose as to which flows should be blocked. This question motivated the design of a priority model for channel time allocation. Price was chosen as a signal in this new priority-based channel time allocation scheme. The flow that was willing to pay the lowest price corresponding to its requirements was the one that was blocked. One added advantage of a price-based channel time allocation policy was that even a malicious flow now had the incentive to request *smaller* CTP when its channel quality improved, since this would result in a monetary

gain.

Our price-based channel time allocation algorithm is a *centralized* auction of channel time, with flows specifying their CTP requirements and maximum bid limits to the BM before the auction begins. The BM then plays out the auction centrally, to avoid the messaging overhead and instability produced by a distributed auction, and determines what fraction of the multi-unit channel time resource is allocated to each flow. An auction is a natural way of resource allotment when each bidder values the resource differently. This is the case with wireless channel time, especially in a hot spot network application of the IEEE 802.11 protocol. The reason is that the same amount of bandwidth may be worth a lot to some flows (e.g. a stock-broker who needs to make an important buying/selling decision based on the price of a certain stock) and not so much to others (e.g. a person who is checking his/her e-mail for the fourth time in an hour). While the fair channel time allocation policy takes into account flow satisfaction and channel utilization, the price-based scheme is, in addition to the above, also concerned with system revenue. We measure satisfaction, as a percentage, for an admitted flow f as: $\frac{p_a(f)}{p_{max}(f)} \cdot 100$. The measure does not take into account rejected flows, which is why in our experiments in the next chapter, we also measure blocking factor. We measure channel utilization as: $\sum_{f \in F} p_a(f)$. The instantaneous revenue of the network provider, in cents/min, is $R = \rho \cdot \sum_{f \in F} p_a(f)$, where F is the set of admitted flows. The total revenue is R aggregated over the entire time of operation of the network.

In this chapter, we describe the algorithms in detail, with pseudo-code. We explain the workings of the algorithms and show how they meet their stated goals.

5.1 Max-min Fairness with Minimum Guarantees

The BM maintains a set of admitted flows F which is initially empty. At the time of flow establishment or CTP re-negotiation, a flow f specifies to the BM its minimum and maximim CTP requirements $p_{min}(f)$ and $p_{max}(f)$. These are calculated by the flow's RA from its augmented minimum and maximum bandwidth requirements, $B'_{min}(f)$ and $B'_{max}(f)$, and its perceived channel capacity $B_p(f)$.

The BM checks whether, for all flows g in the set F of previously registered flows, $1 - \sum_{g \in F} p_{min}(g) \geq p_{min}(f)$. If this is true, the new flow f is admitted ($F = F \cup \{f\}$), else it is rejected and a `reply` message offering it zero CTP is returned to its Rate Adaptor. Note that a best-effort flow with $p_{min}(f) = 0$ is always admitted. A rejected flow may attempt again later to gain access to the channel. Flows are admitted

strictly in the order they arrive, to alleviate starvation of previously rejected real-time flows. The problem of starvation of a best-effort flow after admission is dealt with in Section 5.1.2.

Once the new flow f is admitted, the BM must redistribute channel time within the new set of existing flows F. Since the original admission test was passed by flow f, accommodating it will not cause the CTP allotted to any flow $g \in F$ to fall below its minimum CTP request. Hence, the BM initially sets allotted CTP $p_a(g) = p_{min}(g),\ \forall g \in F$. The remaining channel time, $p_{rem} = 1 - \sum_{g \in F} p_{min}(g)$, is distributed among the flows $g \in F$ in *max-min fair* fashion. The maximum CTP requirement for each flow $g \in F$ in the max-min fair computation is set to $p_{newmax}(g) = p_{max}(g) - p_{min}(g)$. This is because $p_{min}(g)$ has already been allotted to it and it only needs $p_{newmax}(g)$ more to fulfill its maximum CTP requirement. Thus, knowing p_{rem} and $p_{newmax}(g)\ \forall g \in F$, the max-min algorithm can now proceed. Details of the max-min fairness algorithm can be found in the following subsection.

Suppose that out of the remaining channel time p_{rem}, the amount allotted to any flow $g \in F$ by the max-min algorithm is denoted by $p_{mm}(g)$. Now, $0 \leq p_{mm}(g) \leq p_{newmax}(g)$ and $\sum_{g \in F} p_{mm}(g) = p_{rem}$. Then, the total CTP allotted to each flow $g \in F$ is $p_a(g) = p_{min}(g) + p_{mm}(g)$. Note that for best-effort flows, since $p_{min}(g) = 0$, $p_a(g) = p_{mm}(g)$. In other words, channel time is allotted to best-effort flows only after all the higher priority real-time flows are allotted at least their minimum share.

5.1.1 Max-min Fairness

This section describes the max-min algorithm to calculate how much channel time each flow gets beyond its guaranteed minimum requested channel time, after the flow is admitted.

In max-min fairness [22], flows with small channel time requests are granted their requests first; the remaining channel capacity is then *evenly* divided among the more demanding flows. As described above, $p_a(f)$ is first set to $p_{min}(f)$ for all the flows. The channel time that remains, p_{rem}, after satisfying the flows' minimum requirements, is allotted to the flows in max-min fashion. The new maximum requirement for each flow in the max-min algorithm is $p_{newmax}(f) = p_{max}(f) - p_{min}(f)$, because $p_{min}(f)$ has already been allotted to it and must be subtracted from the original maximum requirement. We denote the channel time allotted to flow f by the max-min algorithm as $p_{mm}(f)$. This is in addition to $p_{min}(f)$ allotted before the max-min algorithm is even invoked.

The computation of the max-min allocation is as follows. Initially, the set of flows f, whose new

maximum channel time requirement $p_{newmax}(f)$ has been satisfied, is empty: $\mathcal{R} = \emptyset$. Then, we compute the first-level allotment as $CA_0 = p_{rem}/N$, where N is the total number of flows. Now we include all flows f with $p_{newmax}(f) < CA_0$ in set $\mathcal{R}$, and allot each of them $p_{mm}(f) = p_{newmax}(f)$. Next, we compute $CA_1 = \frac{p_{rem} - \sum_{f \in \mathcal{R}} p_{newmax}(f)}{N - ||\mathcal{R}||}$. If for all flows $g \notin \mathcal{R}, p_{newmax}(g) \geq CA_1$, then we allot each of them $p_{mm}(g) = CA_1$ and stop. Otherwise, we include those flows g with $p_{newmax}(g) < CA_1$ in set $\mathcal{R}$, allot each of them $p_{mm}(g) = p_{newmax}(g)$, and re-compute the next level CA_2. When the algorithm terminates, the allocation $p_{mm}(f)$ for all the flows is max-min fair. The pseudo-code for the algorithm is shown in Figure 5.1. It is clear that the computational complexity of this algorithm is $O(N^2)$. As mentioned earlier, after every flow f's $p_{mm}(f)$ has been determined using the max-min algorithm, the BM sets $p_a(f) = p_{min}(f) + p_{mm}(f)$ and returns this value to flow f's RA.

```
Max-min fair resource allocation algorithm
Input: channel time: p_rem; set of requests: p_newmax[f]
Output: set of allocations: p_mm[f]
proc Max-min(p_rem, p_newmax[f]) ≡
   R := {}; //set of satisfied flows
   N := size_of(p_newmax[f]);
   p_mm[f] := 0;
   while (true) do
         total_satisfied = 0;
         foreach f ∈ R do
            total_satisfied += p_mm[f];
         od
         CA := (p_rem − total_satisfied)/(N − size_of(R));
         stop := true;
         foreach f ∉ R do
            if (p_newmax[f] < CA) then
                                        R := R + {f};
                                        p_mm[f] := p_newmax[f];
                                        stop := false;
            fi
         od
         if (stop) then
                     foreach f ∉ R do
                        p_mm[f] := CA;
                     od
                     break;
         fi
   od
.
```

Figure 5.1: Max-min Fair Resource Allocation Algorithm

5.1.2 Discussion

In our policy, as mentioned earlier, best-effort flows are only given access to the channel after all the real-time flows' minimum requirements are satisfied. This could lead to starvation of the best-effort flows, in the rare case that $\sum_{g \in F} p_{min}(g) \rightarrow 100\%$. One way to eliminate this problem would be to partition channel time into a large *minimum-guarantee* portion and a small *max-min fair* portion, similar to the bandwidth partitioning in [7]. The minimum requirements of the real-time flows, i.e., all $p_{min}(g) > 0$, are allotted *only* from the minimum-guarantee portion. The max-min fair portion, along with any left over minimum-guarantee portion, is used to allot the flows' extra CTP $p_{mm}(g)$, using just a max-min scheme. Both real-time as well as best-effort flows, i.e. all flows with $p_{newmax}(g) > 0$, can vie for this portion. The presence of a separate max-min fair portion ensures that, however large the minimum requirements of the real-time flows, some channel time is always available for best-effort flows to vie for, so they are never starved. The disadvantage of having a separate max-min fair portion is that the channel time available to satisfy minimum guarantees of real-time flows (the minimum-guarantee portion) is reduced, which could lead to more real-time flows being dropped. The BM could decide to adopt this partitioning strategy only when it detects that $\sum_{g \in F} p_{min}(g) \rightarrow 100\%$.

5.2 Price-based Channel Time Allocation Policy

5.2.1 Algorithm

For the price-based scheme, the RA of a flow f submits to the BM the maximum bid $D(f) = \rho_u(f) \cdot p_{min}(f)$ in (cents/min)/%CTP that the flow is willing to pay for 1% of CTP, in addition to the flow's minimum and maximum CTP requirements. The BM computes the price-index of the flow f, $mp(f) = \frac{\rho_u(f) \cdot p_{min}(f)}{p_{max}(f)} = \frac{D(f)}{p_{max}(f)}$. If $p_{min}(f) = 0\%$, i.e., for a best-effort flow, $D(f)$ is set directly by the flow based on the importance of the best-effort bandwidth to his/her activity. It is the maximum bid in cents per minute that the flow is willing to make towards whatever CTP he/she can obtain. Price-index $mp(f)$ is still $D(f)/p_{max}(f)$ for such a flow. The price-based channel allocation algorithm at the BM is responsible for determining the current system-wide channel price ρ_s in (cents/min)/%CTP and the individual flows' CTP allocations, based on these inputs from the flows' UAs. The pseudo-code of the price-based channel time allocation algorithm is shown in Figure 5.2, and explained below.

Price Setting

The network provider sets a fixed *reserve* price ρ_r which is derived from the cost of maintenance of the network. The system-wide price ρ_s per second of 1% of CTP is never lower than the reserve price, i.e. $\rho_s \geq \rho_r$. In the trivial case where the sum of flow requirements does not exceed the channel time capacity of the system, i.e., $\sum_{f \in F} p_{max}(f) \leq 100\%$, the BM can satisfy all flows and it sets the system-wide price as follows: the system-wide price ρ_s is just set to the maximum of the reserve price and $\underset{f \in F}{min}\{mp(f)\}$, i.e., $\rho_s = max\{\rho_r, \underset{f \in F}{min}\{mp(f)\}\}$. A reserve price ensures that flows do not get away with having very small $mp(f)$, and thus yielding very low revenue to the system. In the trivial case, by setting the price to $\underset{f \in F}{min}\{mp(f)\}$, when the latter is greater than ρ_r, our algorithm finds the largest price for which average flow satisfaction is 100% and channel utilization is maximum. Alternatively, in this trivial case, the network provider may opt to just set the price equal to the reserve price, irrespective of the $mp(f)$ values.

In the non-trivial case where the sum of requests from all flows f exceeds channel capacity, i.e., $\sum_{f \in F} p_{max}(f) > 100\%$, the BM must make hard choices and divide the set of candidate flows into a set of fully satisfied and a set of unsatisfied flows as follows: the BM allocates the CTP so that flow satisfaction is maximized while keeping channel utilization at 100%. Initially, the flows $f \in F$ are sorted in non-decreasing order of price-index $mp(f)$. Let V be this sorted set. In the algorithm, V represents the set of flows whose maximum CTP requirements are satisfied. At every iteration, we check if the current set of flows V can all be fully satisfied. Otherwise, we move the flow with the lowest $mp(f)$ into set W of unsatisfied flows, and try again to satisfy all remaining members of V.

Repeatedly, the flow with the lowest price-index (i.e., the flow at the front of set V) is removed from the set of fully satisfied flows V and added to a set of unsatisfied flows W until $\sum_{f \in V} p_{max}(f) < 100\%$. The importance of this operation is that it signifies the increase of the system-wide price ρ_s to keep channel utilization at 100%. If the price is low, then flows would get high allocations, and channel utilization would exceed 100%. Obviously, this cannot happen, so we must increase the system-wide price so that we can lower resource allocations and bring down the channel utilization to 100%. Set W contains all unsatisfied flows with $mp(f) < \rho_s$, and set V contains all satisfied flows with $mp(f) \geq \rho_s$.

For a given split of flows between V and W, the price is set to $\rho_s = \frac{\sum_{g \in W} D(g)}{100\% - \sum_{f \in V} p_{max}(f)}$ [1]. At a certain

[1]Notice that the system price ρ_s is determined using the total bid $\sum_{g \in W} D(g)$ and total allocation $100\% - \sum_{f \in V} p_{max}(f)$ of those flows (members of set W) whose requirements are *not* fully satisfied. This is similar to the case of a traditional *second-price* or

point in the algorithm, if $\rho_s \leq \min_{f \in V}\{mp(f)\}$, then the algorithm has successfully found a valid split of flows between V and W, and it terminates. If this condition is not satisfied, the current split of flows between sets V and W is invalid. It means, once again the flow with the minimum $mp(f)$ is moved from V to W and the price is recomputed for the new values of set V and set W. The procedure continues until a value of ρ_s is found, where $\rho_s \leq \min_{f \in V}\{mp(f)\}$, or until set V is empty. The last value of ρ_s becomes the price for the set of flows F. The portion of the algorithm described so far runs in $O(\|F\|)$ time.

Channel Time Allocation

The CTP allocated to each flow $f \in F$ is simply $p_a(f) = min\{p_{max}(f), D(f)/\rho_s\}$. For flows in set V, the value of $mp(f) = \frac{D(f)}{p_{max}(f)} \geq \rho_s$. Hence, $p_a(f)$ simply becomes $p_{max}(f)$. For flows in set W, the value of $mp(f) = \frac{D(f)}{p_{max}(f)} < \rho_s$. Hence, for these flows, $p_a(f) = \frac{D(f)}{\rho_s}$, since this is definitely less than $p_{max}(f)$. Now, for some flows g in set W, it may happen that $p_a(g) < p_{min}(g)$. In this case, the flow g_{rej} with the *lowest* value of $mp(g)$, among these flows g, is deleted from the original set F. It is *blocked* because it is not paying enough to even have its minimum CTP requirement be satisfied. The algorithm must run all over again for the new set $F = F - \{g_{rej}\}$, i.e., V is re-initialized to the new set F sorted in non-descending order of $mp(f)$, and set $W = \{\}$. The worst-case overall running time of the pricing algorithm is thus $O(\|F\|^2)$.

In summary, in the non-trivial case when $\sum_{f \in F} p_{max}(f) > 100\%$, the algorithm attempts to maximize both flow satisfaction and channel utilization, while maintaining *bid-proportional fairness* (explained later) among the flows. We formally characterize the optimization problem in the non-trivial case, as follows: the pricing algorithm attempts to maximize $\sum_{f \in F} \frac{p_a(f)}{p_{max}(f)}$, under the constraints (a) $\sum_{f \in F} p_a(f)$ = 100%, and (b) $\frac{R(f)}{p_a(f)} = \frac{R(g)}{p_a(g)}$, for all admitted flows f, g.

As mentioned in the next subsection, if the flows were to enter into an ascending, multi-unit auction of CTP, the price they will naturally settle upon at equilibrium is the price the algorithm ultimately settles upon, in the case where $\sum_{f \in F} p_{max}(f) > 100\%$. This price thus reflects the true worth of channel time amongst the flows in the network.

If the algorithm terminates with set V not empty, then it basically means that the CTP allocated to each flow in set W is limited by its maximum bid, and the CTP allocated to each flow in set V is limited by its

Vickrey auction [23]. This feature encourages *truthful bidding*. Flows are encouraged to honestly bid high, in the knowledge that they will not be over-charged if they "win." The charge will be determined not from their bid, but from that of the "losers."

maximum requirement. In other words, the flows in set V have paid more than they require to satisfy their maximum CTP requirement and will hence receive a refund $\lambda = D(f) - p_a(f) \cdot \rho_s$. Conversely, the flows in set W have paid less than they require to satisfy their maximum CTP requirement and will have to settle for less CTP, while their entire bid is consumed by the system [2]. Incidentally, the algorithm ensures that flows with the same value of $mp(f)$ are, ultimately, either placed all together in V or all together in W. Thus, when initially sorting the flows in non-decreasing order of $mp(f)$ in V, ties can be broken arbitrarily.

At any instant, the price for 1% CTP is the same for all flows. Incidentally, the price for 1 bit per second (bps) throughput is different for flows with different channel qualities. This is also the case in [24]. As flows arrive and leave, the price varies depending on the competing demand. The flow is billed the instantaneous price times its instantaneous allotment, aggregated over its entire session, when it leaves the network. As mentioned earlier, we adopt an "acceptable or nothing" QoS philosophy. A flow gets acceptable QoS (between minimum and maximum requirements) or no service at all. When demand increases and CTP becomes dearer, flows willing to pay too little for their minimum requirement are blocked[3]. Such flows can either increase their maximum bid or decrease their minimum requirement and rejoin the network, after a time delay T. The algorithm works in such a way that, if $p_{min}(f) = 0\%$ and $p_{max}(f) = 100\%$, $\forall f \in F$, then the flows will be allocated CTP in the ratio of their $mp(f)$ and in the ratio of their $D(f)$ values. The price in this case will be $\sum_{f \in F} mp(f)$ (cents/sec)/%CTP. Our algorithm thus applies a simple *bid-proportional* allocation, where each flow is allotted CTP proportional to its bid, to our utility function.

5.2.2 Auction of Channel Time

In this section, we show that our scheme is equivalent to an "auction" in which flows repeatedly submit higher and higher bids until they are either satisfied or are unwilling to pay any more. The repeated bidding in an auction is not feasible in our scenario, and hence the need for our centralized algorithm.

Hypothetical Auction

Let us consider a hypothetical, ascending, multi-unit auction [23] of channel time between the flows, such that each flow is allotted a CTP proportional to its bid. (CTP allotted at any instant to a flow is equal to the

[2] It is not logical for a flow with a larger $mp(f)$ to have its maximum requirement satisfied while one with a smaller $mp(f)$ value is not fully satisfied. This is the reason why flows are moved from V to W only in non-decreasing order of $mp(f)$.

[3] Best-effort flows, since they have $p_{min}(f) = 0\%$, are always admitted. Recall also that they set $D(f)$ differently from flows with non-zero minimum CTP requirements.

Iteration	Action	Bids (c/min)	Price ((c/min)/%CTP)	Allocations (%CTP)	Comments
0	Initial	2,4,6	Reserve price: 0.1	17,33,50	none satisfied
1	f_1 bids	2.5,4,6	0.125	20,32,48	f_1 satisfied
2	f_2 bids	2.5,5.67,6	0.142	18,40,42	f_2 satisfied
3	f_3 bids	2.5,5.67,12	0.202	12,28,60	f_3 satisfied/exhausted
4	f_1 bids	4.42,5.67,12	0.221	20,26,54	f_1 satisfied
5	f_2 bids	4.42,10,12	0.264	17,38,45	f_2 exhausted
6	f_1 bids	5.5,10,12	Eqm. price: 0.275	20,36,44	f_1 satisfied f_2 exhausted f_3 exhausted

Table 5.1: Example of hypothetical auction.

flow's bid divided by the sum of all bids at that instant.) Now, in such an auction flows would increase their bids in order to obtain more and more CTP. This bidding would continue until all flows have either obtained their maximum CTP requirement $p_{max}(f)$ or have reached the maximum they are willing to bid $D(f)$, i.e., the maximum sum of money they can afford. At this point, the system reaches equilibrium until a new flow arrives, an existing flow leaves, or some other parameter, such as a flow's maximum bid, changes.

Table 5.1 illustrates an example of the hypothetical, proportional, ascending, multi-unit auction. Three flows, f_1, f_2 and f_3 compete for CTP. Assume that the minimum CTP requirements $p_{min}(f)$ of the three flows are 5%, 10%, and 30% respectively and their maximum CTP requirements $p_{max}(f)$ are 20%, 40% and 60% respectively. Assume further that their $\rho_u(f)$ values in cents per minute per %CTP are 1.2, 1.0, and 0.4 respectively. Thus, their respective maximum bids $D(f)$ are 6, 10 and 12 cents per minute, giving them the respective $mp(f)$ values of 0.3, 0.25 and 0.2 (c/min)/%CTP. The table shows how the auction plays out. Each flow starts bidding at the reserve price of 0.1 (c/min)/%CTP. At each stage, the %CTP available to a flow is proportional to its instantaneous bid. We assume only one bid per iteration. The order of bidding has no bearing on the ultimate outcome of the auction, although the individual iterations may differ for different orders. Without any loss of generality, we assume a round-robin order of bidding between the three flows. The only information required by a flow to compute its next bid on its next turn is the current price ρ. Using this alone, the flow can figure out what should be his/her next bid in order to obtain either the maximum requirement or exhaust his/her maximum budget $D(f)$.

Note that the hypothetical auction described here is a proportional one instead of a traditional one (e.g. Vickrey auction [23]) in which the highest bidder's maximum request is satisfied, then the next highest bid-

der's maximum request is satisfied, and so on. The bid-proportional allocation is hence not an "auction" in the traditional sense of the word, but we call it an auction to represent the scenario where flows repeatedly bid higher and higher in order to satisfy themselves, until they reach a particular limit. If we used a traditional "all-or-nothing" auction, several flows would be rejected while others would be operating at their highest possible quality. In other words, blocking factor would be very large. Such a scheme would not take advantage of the *adaptive* nature of multimedia flows, whereby many flows can be simultaneously accommodated if they all agree to operate at a lower quality. Our scheme improves blocking factor considerably while still allowing flows that insist on operating at their highest possible quality to pay a large enough sum and achieve it.

Centralized Simulation of Hypothetical Auction

Now, obviously, an auction of the type described above is infeasible in our hot spot scenario. The repeated bids, each resulting in a different CTP allotment, constitute a very large overhead. The number of bids or iterations depends on the reserve price, the number of flows, and the respective requirements and maximum bids of the flows. The delay in attaining equilibrium can also be untenably long, and during this time, the flows' CTP allotment will be continuously varying. We *centralize* this auction by having all the flows provide the algorithm their limiting parameters: $p_{max}(f)$ and $D(f)$. The algorithm described earlier then *simulates* the ascending auction. It determines the same equilibrium price, i.e. 0.275 (c/min)/%CTP in the example, and CTP allotment as the hypothetical auction (see Section 5.2.3 for proof) in $O(\|F\|^2)$ time, while the hypothetical auction would have required several iterations of bidding from each flow. The auction in the example takes six iterations, while our algorithm takes only two for the same input. (Two flows are successively moved from V to W.) The formats of the hypothetical auction and centralized approach are compared in Figure 5.3.

One possible problem with the auction-based channel time allocation policy is that of *collusion*. If all bidders co-operate with each other and decide to make very small bids, then the system's revenue can be adversely affected. Having a reserve price mitigates this problem to an extent [25, 26]. Furthermore, in a hot spot network at a public place such as an airport or cafe, it is impractical for *all* the flows of the network to meet beforehand and agree to collude. The collusion fails if even a single flow from outside the group of colluders enters the network. Thus, if all the colluders have artificially small maximum bids ($D(f)$) and

only one flow makes a normal maximum bid, that flow will get most of the channel time and the others will all be adversely affected, forcing them to increase their maximum bids. The promise of more channel time for a larger bid, as well as the assurance that any extra money will be refunded, should encourage flows to set their maximum bids $D(f)$ high, and thus increase system revenue. The refund helps to avoid the "winner's curse" [27].

5.2.3 Properties of Price-based Channel Allocation Algorithm

Theorem 1 *For a given set of flows F, let the sum of the maximum CTP requests exceed channel capacity, i.e. $\sum_{f \in F} p_{max}(f) > 100\%$. For this non-trivial case, the price and channel time allocation obtained by the pricing algorithm is identical to the equilibrium price and equilibrium channel time allocation of the proportional, ascending, multi-unit auction described above.*

Proof: An equilibrium is reached in the hypothetical auction when it is either infeasible or impossible for any flow in the system to bid a larger amount. It is infeasible for flows to do so when they already have their maximum requirement. It is impossible to do so when they have already exhausted their entire maximum bid. Until it reaches one of these two conditions, each flow continues to bid higher and higher. At equilibrium, thus, all the flows in the hypothetical proportional auction can be classified into one of two sets - the set of satisfied flows S and the set of exhausted flows E. Our algorithm also distributes the flows into two sets V and W. We prove that set S in the hypothetical auction equals set V in the algorithm, and set E equals set W, and thus prove the equivalence of the two schemes.

Ultimately, as shown in Figure 5.4, both the hypothetical auction and pricing algorithm determine a valid splitting price ρ_s that splits a set of flows F arranged in sequence of non-decreasing $mp(f)$ into two subsets:

1. The subset of flows (V in the algorithm, S in the hypothetical auction) with $mp(f) \geq \rho_s$. Each flow in this subset obtains its maximum CTP requirement.

2. The subset of flows (W in the algorithm, E in the auction) with $mp(f) < \rho_s$. Each flow in this subset cannot afford his/her maximum CTP requirement but exhausts its maximum bid towards obtaining as much CTP as possible.

We have to prove that the splitting point of the sequence is the same for both the algorithm and hypo-

thetical auction schemes. We prove this by contradiction. The splitting point as determined by the auction at equilibrium cannot be lower in the non-decreasing $mp(f)$ sequence than that determined by the algorithm because the algorithm scrolls over all possible splitting points one by one from the lowest to highest value of $mp(f)$. If a valid split in the sequence existed earlier, such that all flows to its left were exhaused and those to its right were satisfied, the algorithm would have found it. The algorithm finds the lowest possible valid splitting point in the non-decreasing sequence.

The auction can find a higher splitting point only if flows ignore the lower one found (and hence proven to exist) by the algorithm and continue to bid even when fully satisfied or exhausted. Basically, the auction iterates over a number of stages, in each of which the current bidder is either fully satisfied or exhausted. It *must* hence encounter the stage found by the algorithm in which *all* flows are fully satisfied/exhausted. For the auction to continue beyond this point, and find a higher splitting point than the algorithm, bidding would have to continue even after equilibrium has been reached. So the auction does not find a higher splitting point than the algorithm, either. The auction needs the dissatisfaction and non-exhaustion of at least one flow to proceed. Hence it cannot proceed beyond the first point where everyone is either satisfied or exhausted. Thus both schemes split the set of flows arranged in sequence of non-decreasing $mp(f)$ values at the exact same point. (Recall also that, in both schemes, the split is such that flows with identical $mp(f)$ values end up on the same side of the split.)

Now, for a given valid split, there is only one possible price ρ_s, given that all 100% of the channel time is ultimately auctioned away. All the flows in V get allotted their maximum requirements. The remaining CTP is shared proportionately by all the flows in W who exhaust their respective maximum bids towards the share. Thus, the sum of the maximum bids of all flows in W, divided by the CTP remaining after all flows in V are satisfied, is the price. For a given price ρ_s, the CTP allocation for each flow f is the same, irrespective of what scheme is used: $min\{p_{max}(f), D(f)/\rho_s\}$. Flows rejected by the centralized algorithm also eliminate themselves from the auction by withdrawing their bid once they realize that their minimum CTP requirements cannot be met.

Thus, both the auction, when in equilibrium state, as well as our algorithm, return the same price and CTP allocation. □

Theorem 2 *The pricing algorithm of Figure 5.2 terminates in a finite number of steps.*

Proof: The outer loop of the algorithm removes flows from a sequence of length $\|F\|$ whose minimum

requirements cannot be satisfied. Even in the worst case only a finite number of flows can be thus removed. The inner loop iterates over all gaps between elements of a sorted sequence of finite length at most $\|F\|$, in order to find a valid split in the sequence. The number of such gaps is finite and equal to $\|F\|$-1. Thus the overall algorithm terminates in a finite number of steps. □

Theorem 3 *For a given set of flows F, let $\sum_{f \in F} p_{max}(f) > 100\%$. In this case, the algorithm finds the only price that yields 100% channel utilization and maximum flow satisfaction, using a bid-proportional allocation scheme.*

Proof: This theorem pertains to the performance of the algorithm in the non-trivial case. We prove this theorem by contradiction. In a bid-proportional allocation scheme, the allocation to each flow f is the ratio of its bid $D(f)$ to the system price ρ_s.

In the given case, our algorithm ensures 100% channel utilization because the higher bidding flows are fully satisfied and the remainder of channel (100% $-\sum_{f \in V} p_{max}(f)$) is fully utilized by the exhausted flows. Assume we randomly chose a system-wide price ρ_x smaller than that determined by our algorithm ρ_s, to use in place of ρ_s, i.e., $\rho_x < \rho_s$. Then, the ratio $\frac{D(f)}{\rho_x}$, i.e. the allocation using ρ_x for each flow would be greater than current allocation $p_a(f) = \frac{D(f)}{\rho_s}$. Since channel utilization is already 100% currently, an increase in allocation for each flow would only drive the utilization over 100%, which is not possible. Moreover, some of the flows who were previously rejected since their minimum requirements could not be met may now once again be allocated CTP using the new price, thus further increasing aggregate allocation, and consequently channel utilization. Hence there does not exist a price $\rho_x < \rho_s$ that yields 100% channel utilization.

Now, assume we randomly picked a system-wide price $\rho_x > \rho_s$ to use instead of that determined by our algorithm ρ_s. Then, this may or may not decrease the allocation of the flows satisfied by the original price ρ_s. This is because they might be able to dig into their refunds and still continue to be fully satisfied even at this greater price. However, their allocations cannot be increased since they are already fully satisfied. On the other hand, this new price will definitely decrease the allocation of the flows exhausted by using the original price ρ_s. There exists at least one of these since it impossible for all flows to be fully satisfied if $\sum_{f \in F} p_{max}(f) > 100\%$. Thus with $\rho_x > \rho_s$, the allocations of the flows $\in V$ fully satisfied originally does not increase, while the allocations of the flows $\in W$ exhausted originally definitely decreases. The overall allocation hence decreases, which only decreases both flow satisfaction and channel utilization.

Thus the price determined by the pricing algorithm is the only one which both keeps channel utilization 100% and maximizes flow satisfaction at this level, given a bid-proportional allocation scheme. □

Theorem 4 *For a given set of flows F, let $\sum_{f \in F} p_{max}(f) \leq 100\%$. In this trivial case, the pricing algorithm finds the largest price ($\geq \rho_r$), for which flow satisfaction is 100% and channel utilization is maximum.*

Proof: This theorem pertains to the performance of our algorithm in the trivial case. We prove this theorem also by contradiction. Assume that we replace the algorithm's price ρ_s with a randomly chosen larger price ρ_x, i.e. $\rho_x > \rho_s$. Then this larger price ρ_x would result in all flows being allotted CTP lower than they are currently allotted. Thus, their respective levels of satisfaction would only decrease. Channel utilization would also consequently only decrease.

Assume that we randomly pick a price lower than the one the pricing algorithm finds, in the trivial case, i.e. $\rho_x < \rho_s$. Then, firstly, this new price may violate the reserve price ρ_r. Assume further then that, although the new price ρ_x is lower than the one the pricing algorithm finds ρ_s, it is still not less than ρ_r, i.e., $\rho_r \leq \rho_x < \rho_s$. Then all flows are eligible to get CTP higher than their current allotment. However, since we are currently using the *minimum* of the $mp(f)$'s as the price, they are all *already* getting their maximum CTP requirement, currently. This is because they are all already paying equal to or more than the price set. There is no reason to increase allocations more than this maximum CTP requirement, by setting price lower. So, the current price results in 100% flow satisfaction and maximum possible channel utilization. The lower price ρ_x does not change allocations and hence does not change flow satisfaction and channel utilization.

Thus the current price, as determined by our scheme, is the maximum price, no less than the reserve price, for which flow satisfaction is 100% and channel utilization is maximum. □

5.3 Stepwise Utility Function

So far we have only considered a linear utility function in this work. This type of utility function is useful when continuous adaptation is possible as in the case of best-effort flows, and also some real-time flows. However, for several types of real-time multimedia flows, the adaptation can be better described by a *stepwise* utility function. For example, a video streaming application that can adjust its resource consumption by varying its frame rate over a range of discrete levels 5fps, 10fps, 25fps, etc. An example stepwise utility function is shown in Figure 5.5(a).

Our pricing algorithm can be extended to work with stepwise utility functions also. Stepwise utility functions can be constructed by a user by determining bandwidth requirements for uplink and downlink transmissions at various discrete application operating levels, converting these into CTP requirements, summing the requirements of both directions and getting a function shown in Figure 5.5(a). We allocate CTP in the case of stepwise utility functions running the same algorithm *twice* with different inputs. We thus use a *two-tier* allocation algorithm for stepwise functions, where each "tier" (i.e., phase or iteration) is an instance of the algorithm of Section 5.2.

Phase 1: The first iteration of the two-tier algorithm takes as initial inputs $D(f) = p_{min}(f) \cdot \rho_u(f)$ and the stepwise utility function. It determines an initial operating point $\omega(f)$ for each user f somewhere on the straight line joining the lowest and highest utility of the user, using a phase 1 system-wide price ρ_{s1}. Let the CTP corresponding to this operating point be $p_\omega(f)$. We allocate each user CTP $p_{a1}(f)$ corresponding to the highest step at or below the operating point $\omega(f)$ on the straight line. This is shown in Figure 5.5(a). Each user is charged $p_{a1}(f) \cdot \rho_{s1}$ by the system for this allocation, where ρ_{s1} is the price determined in the first iteration (i.e. phase 1) of the two-tier allocation algorithm. Phase 1 of the algorithm also removes all users whose minimum CTP requirement cannot be satisfied, as was the case in Section 5.2. At the end of Phase 1, all users are refunded $\lambda_1(f) = D(f) - p_\omega(f) \cdot \rho_{s1}$, as was the case in Section 5.2. The rest of the bid $p_\omega(f) \cdot \rho_{s1} - p_{a1}(f) \cdot \rho_{s1}$ is used as the user bid for the second iteration of the algorithm, i.e., $D_2(f) = p_\omega(f) \cdot \rho_{s1} - p_{a1}(f) \cdot \rho_{s1}$.

Phase 2: The available channel time for allocation in Phase 2 is unit channel time minus the fraction used up in Phase 1, i.e. the new available channel time is set at 100% $-\sum_{f \in F} p_{a1}(f)$. The second iteration (i.e. Phase 2) of the two-tier algorithm is not required if this value is zero, because there is no channel time left after Phase 1. Each user's bid in the second iteration of the two-tier algorithm $D_2(f)$ is calculated as described above. Each user is also assigned new maximum and minimum CTP requirements $p_{max2}(f)$ and $p_{min2}(f)$ respectively, for Phase 2 of the algorithm. The new value of $p_{min2}(f)$ and the new value $p_{max2}(f)$ are *both* set to the difference between the currently allocated CTP $p_{a1}(f)$ and the CTP corresponding to the next highest step in the stepwise utility function, normalized over the new available channel time: $p_{max2}(f) = p_{min2}(f) = \frac{p_{step}(f)}{100\% - \sum_{f \in F} p_{a1}(f)}$. This is shown in Figures 5.5(a) and 5.5(b). The value of $p_{step}(f)$ is the difference between the already allocated CTP from Phase 1 $p_{a1}(f)$ and the CTP corresponding to the next step on the stepwise function.

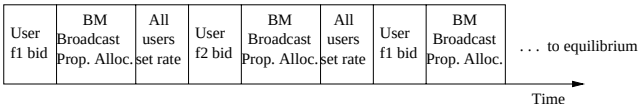
The minimum and maximum CTP requirements are set equal in Phase 2 to indicate that users may either make the step up on the utility function, or they may not. They cannot be allocated some intermediate value of CTP. Due to the way the new requirements and new bids are set up, the second iteration of the two-tier algorithm essentially determines which admitted users make the jump up from their current allocation step to the next step on the stepwise utility function. The second iteration of the algorithm returns a Phase 2 system-wide price ρ_{s2} and Phase 2 allocation to all users $p_{a2}(f)$ [4]. For users that do make the step up, the Phase 2 allocation $p_{a2}(f) = p_{max2}(f) = p_{min2}(f)$. For users that do not make the step up, Phase 2 allocation $p_{a2}(f) = 0$. Users that are rejected in the second iteration (i.e., $p_{a2}(f)$ = 0) still get the CTP allocated to them in the first iteration, i.e., overall CTP allocation $p_a(f) = p_{a1}(f)$. Users that are admitted in the second iteration jump up one step along their utility functions and their overall allocation $p_a(f) = p_{a1}(f) + p_{a2}(f)$. Phase 2 refund is calculated as $\lambda_2(f) = D_2(f) - p_{a2}(f) \cdot \rho_{s2}$. This is added to Phase 1 refund, so a user f's overall refund is $\lambda(f) = \lambda_1(f) + \lambda_2(f)$.

Using the above *two-tier* approach, we can thus adapt the basic pricing algorithm to a stepwise utility function. As in the case of the pricing algorithm in Chapter 5.2, when users arrive or leave, or user parameters change, the two-tier algorithm must be re-run in its entirety. Note that the two-tier scheme works even when some utility functions are stepwise while others are in the form shown in Figure 2.1. In such a case, users with the continuous straight-line utility functions simply do not participate in the second iteration. The two-tier scheme has the same order of time complexity as a single run of the pricing algorithm, because the $O(\|F\|^2)$ algorithm is simply run twice. The determination, for each user, of the highest step lower than operating point $\omega(f)$ and the step immediately higher than this point takes $O(\|F\|)$, assuming a constant number of steps in the stepwise function. Thus the overall order of time complexity of the algorithm is not increased. It should be observed however that the price of each unit of CTP is not the same in the two-tier algorithm, since ρ_{s1} and ρ_{s2} are different. This is a difference from the single-iteration case.

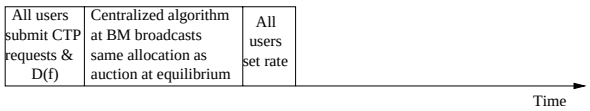
[4] It can be easily proven that $\rho_{s2} < \rho_{s1}$. This is because in the first iteration, when allocating along the continuous line joining highest and lowest utilities, the price ρ_{s1} could only yield $p_\omega(f) - p_{a1}(f)$ CTP. However, in the second iteration, the price ρ_{s2} yields $p_{a2}(f) - p_{a1}(f)$ CTP to successful users where $p_{a2}(f) > p_\omega(f)$. Hence, the price $\rho_{s2} < \rho_{s1}$.

proc *CTAlloc*$(V := Fsorted\ in\ non-decreasing\ order\ of\ mp(f)) \equiv$
do
$redo := false;$
if $(\sum_{f \in V} p_{max}(f) \leq 100\%)$ **then**
$\rho_s := max\{\rho_r, \min_{f \in V}\{mp(f)\}\};$
foreach $f \in V$ **do**
$p_a(f) := min\{p_{max}(f), D(f)/\rho_s\};$
od
else
$Move\ flows\ from\ front\ of\ V\ into\ W\ until\ \sum_{f \in V} p_{max}(f) < 100\%\ or\ V = \{\};$
while $(\left\lceil \rho_s := max\{\rho_r, \frac{\sum_{g \in W} D(g)}{100\% - \sum_{f \in V} p_{max}(f)}\} \right\rceil > \min_{f \in V}\{mp(f)\})$ **do**
if $(V = \{\})$ **then**
$break;$
else
$W := W + First\ flow\ \in V;$
$V := V - First\ flow\ \in V;$
fi
od
foreach $f \in V$ **do**
$p_a(f) := p_{max}(f);$
od
foreach $g \in W$ **do**
$p_a(g) := D(g)/\rho_s;$
od
fi
if $(\exists f \in V \mid p_a(f) < p_{min}(f))$ **then**
$Delete\ f\ from\ V;$
$redo := true;$
fi
while $(redo = true);$
.

Figure 5.2: The price-based channel time allocation algorithm.



(a) Ascending, proportional, multi-unit auction.



(b) Centralized channel-time allocation approach.

Figure 5.3: Comparison of hypothetical auction and centralized algorithm.

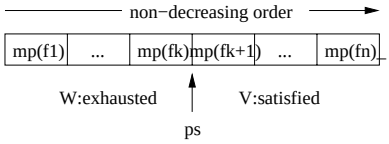
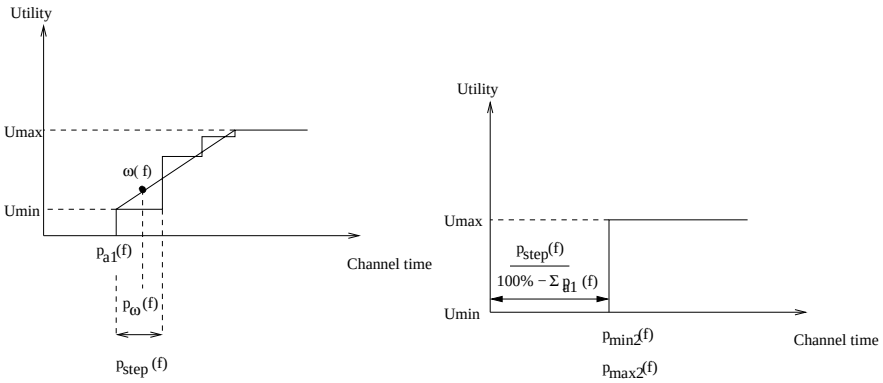


Figure 5.4: Split between exhausted and satisfied flows.



(a) Phase 1.

(b) Phase 2.

Figure 5.5: Two-tier pricing approach for stepwise utility functions.

Chapter 6

Experimental Results

We conducted experiments with both our overall bandwidth management architecture in a single-hop IEEE 802.11 environment as well as specifically with the price-based channel time allocation algorithm. This is because in our experiments with the overall architecture, the BM was running the max-min fair with minimum guarantees channel time allocation algorithm by default. Experiments on the channel capacity estimation scheme were presented in Chapter 3 itself. We evaluated the performance of our dynamic bandwidth management system using both a prototype testbed as well as through simulations using the *ns-2* simulator. We used our testbed when evaluating the performance of a flow in the presence of both physical channel errors caused by fading and interference effects as well as medium contention from two other active stations, because there is no way to set up physical obstacles such as walls, ceilings and doors that cause signal weakening in *ns-2*. We used *ns-2* simulations to evaluate the performance of the system when there is heavy medium contention due to the presence of a large number of active stations. Through our experiments with the price-based channel time allocation algorithm, we evaluated its performance under various metrics such as blocking factor, mean flow satisfaction and channel utilization, and compared its performance with that of two other competing schemes.

6.1 Bandwidth Management Simulation Experiments

For experiments with large numbers of nodes ($\geq$ 5 nodes) and flows, we used the *ns-2* simulator. We compared the performance of a Bandwidth Management-enhanced IEEE 802.11 network (henceforth called "enhanced IEEE 802.11 scheme") with an IEEE 802.11 network without bandwidth management (henceforth called "base IEEE 802.11 scheme"). We used a 170m $\times$ 170m network area and the transmission

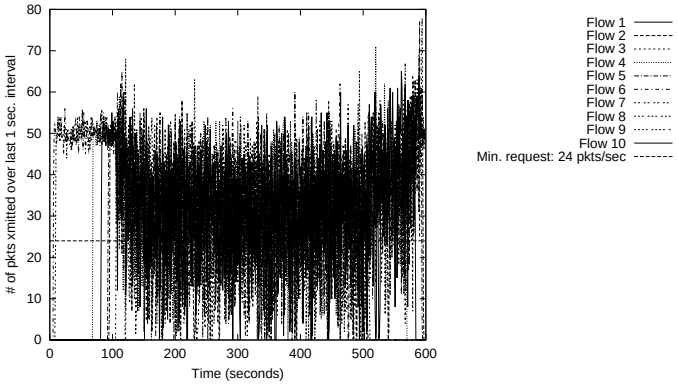
range of each mobile host was 250m. Hence, the entire network area falls within every node's transmission range. The maximum theoretical channel capacity was 2 Mbps. We used the random waypoint mobility model with moderate node speeds in our simulations.

6.1.1 UDP Throughput Performance

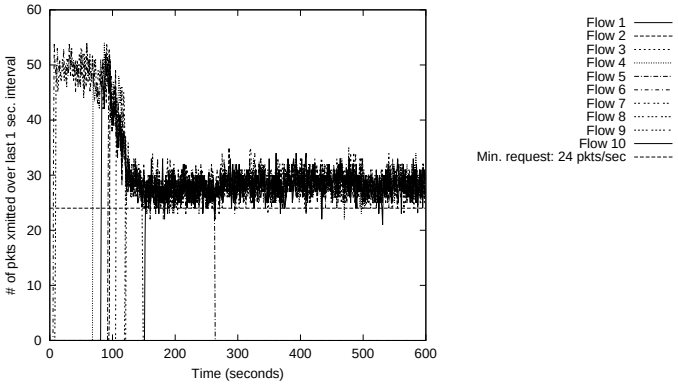
Our first simulation scenario consisted of a 20-node network with 10 flows. Each flow had a minimum bandwidth requirement of 100 Kbps and a maximum bandwidth requirement of 200 Kbps, which are typical of an audio streaming application. All the 10 flows used 512 byte packets. The simulation ran for 600 seconds. The transmission rate used by our scheme at any instant was determined using the method described in Chapter 4. The transmission rate used in the base IEEE 802.11 scheme was a constant set to the maximum requested rate of the CBR flow, as would be the case in an unmanaged application. The RA's inter-update interval was 100 packets and its perceived bandwidth variation-tolerance threshold δ = 15%, by default.

Figure 6.1(a) is a plot of number of packets successfully transmitted over every 1 second interval for each of the 10 flows using the base IEEE 802.11 scheme. Figure 6.1(b) is the same plot using the enhanced IEEE 802.11 scheme. Note that in our scheme two flows needed to be cut-off in mid-operation so that other flows' minimum CTP requirements are not violated. One of these is cut-off at time 149 seconds and the other at time 264 seconds. These times indicate the respective first occasions when the flows in question requested a minimum CTP that could not be supported. When a new flow is admitted, contention increases for all the existing flows. In general, the flow that notices an "unacceptably" poor channel quality and "complains" first is dropped. Alternate flow dropping strategies can also be employed, such as dropping the flow last admitted. Pricing could also play a role here: the flow paying the least can be dropped.

It is clearly evident from the plots that our protocol dramatically improves throughput fairness. In the base IEEE 802.11 scheme, flows often fall far below their minimum bandwidth requirement over the 1 second measurement interval, resulting in a chaotic plot. Using our scheme, flows almost never fall below their minimum bandwidth requirement shown with the horizontal line at 24 packets per second. (100 Kbps / 4096 bit packets is approximately 24 packets per second.) Even when they do, it is only by a small amount. Our scheme thus ensures that the minimum bandwidth requirements of the flows are met with a far higher probability than the base IEEE 802.11 scheme. Figure 6.2 is a 100-second snapshot from the combined plot of Figures 6.1(a) and 6.1(b) that shows the comparative behavior of a single flow (flow 1).



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 6.1: Comparative throughput performance of base and enhanced IEEE 802.11 for 10-flow scenario.

Improving fairness is essential for providing minimum throughput guarantees to wireless multimedia applications. The key factor enabling our scheme to provide minimum bandwidth requirement guarantees with a high probability, is its improved fairness. No flow takes up excess bandwidth during a particular interval thereby depriving another flow of bandwidth and resulting in a large throughput discrepancy (i.e. poor fairness) between the flows. Our scheme also reduces jitter in throughput as compared to base IEEE 802.11. Throughput jitter is the difference in throughput observed over two consecutive same-sized time intervals. It should be as low as possible for a CBR flow. We use 1 second time intervals. We thus designate fairness and throughput jitter as the *key* performance measures that characterize the performance of our system. The better these measures, the higher the probability of the flows meeting their minimum bandwidth

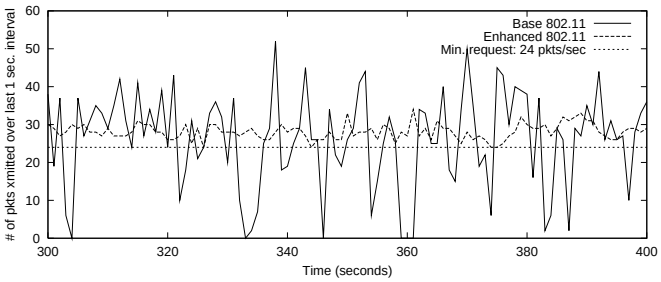


Figure 6.2: Comparative behavior of a single flow over base 802.11 versus enhanced 802.11.

requirements.

While our scheme focuses on ensuring that flows receive their minimum throughput, the delay and delay jitter are also improved as a by-product of our bandwidth management scheme. Since we co-operatively control the sending rate of the flows, we observe a negligible packet loss rate when using our scheme. Due to the rate control, queue length is uniformly short, queuing delay is small, and congestion loss is avoided. Since contention is uniformly low, delay jitter is also improved. With base IEEE 802.11, however, since the transmission rate is set to the maximum, a 33% packet loss rate results due to congestion and the resultant queue overflow. When using our scheme without perceived bandwidth smoothing, each flow re-negotiates its allotted CTP once every 14 seconds on average. In section 6.2.2, we determine that each of these re-negotiations can take upto 60 ms in the presence of contention. This does not affect the flow too much because it continues sending at a rate dictated by the previously allotted CTP and current value of $B_p(f)$ during this interval. It does however represent a small amount of network traffic overhead. The mean throughput of an active flow for our scheme in the above scenario is 8% lower than that of an active flow in base IEEE 802.11. We believe that this lower mean throughput is a small price to pay for the vastly improved *stability* in throughput. The latter property is essential for multimedia applications. In the next subsection, we will discuss the reasons for throughput deterioration and present mechanisms to reduce the flow-initiated re-negotiation overhead.

6.1.2 Overhead for UDP Experiments

There exists a trade-off between network traffic overhead and performance in terms of fairness and jitter. We need to be able to quantify the fairness and throughput jitter so that we can measure how much they are affected when we try to reduce overhead.

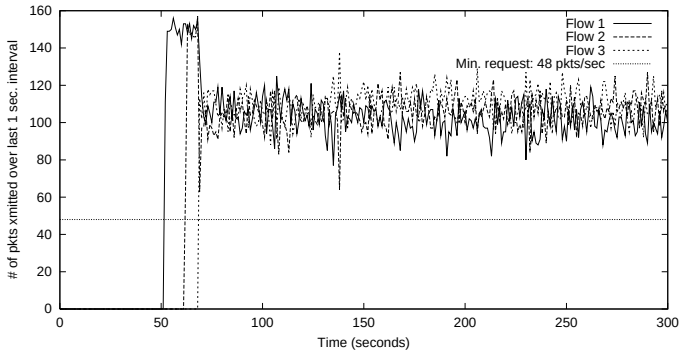
In our simulations, we measure the number of packets of each flow transmitted over each 1 second interval in the 600 second run. Let us denote the number of packets transmitted by flow f over second i as N_i^f. Let the average over all flows of number of packets transmitted in second i be denoted as $\hat{N}_i$. Let the set of *active* flows, i.e. flows that have been established but not yet torn down or cut-off, during second i be A. We only measure throughput per second for the duration in which all flows are active together. Assume that the number of seconds for which the measurement is done is n.

We define a fairness metric $FM = \frac{\sum_{f \in A} |N_i^f - \hat{N}_i|}{||A||}$. We also define a throughput jitter metric for a flow f, $JM_f = \frac{\sum_{i=1}^{n-1} |N_i^f - N_{i+1}^f|}{n-1}$. The overall jitter metric JM is the mean of the JM_f's, i.e. $JM = \frac{\sum_{f \in A} JM_f}{||A||}$.

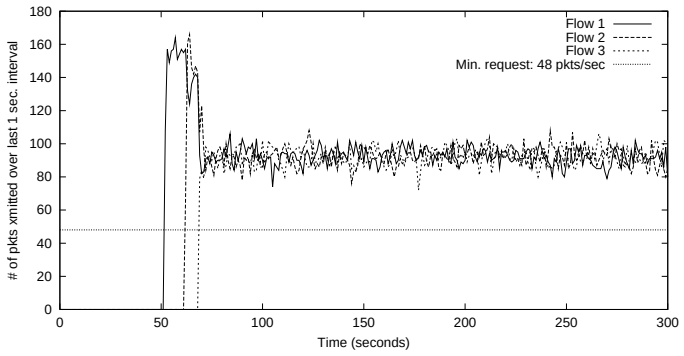
For the experiments in this subsection, we use a different network scenario in which there are 6 nodes in the *ns-2*-simulated wireless network and 3 flows. The flows each require a minimum throughput of 200 Kbps (approximately 48 packets/sec.) and a maximum throughput of 600 Kbps. We ran this simulation scenario for a duration of 300 seconds. All other simulation parameters exactly remain the same from the previous subsection. We used the period when all three flows are active for all our measurements.

Figures 6.3(a) and 6.3(b) show the number of packets transmitted over every 1 second for base IEEE 802.11 and enhanced 802.11 respectively. Once again, it is evident from the plots that our scheme performs better in terms of both fairness and throughput jitter. However, we apply our metric to determine exactly *how much* our scheme improves these performance measures. We obtained a value of FM = 6.72 packets for base IEEE 802.11 versus FM = 4.06 packets for our scheme. (Lower FM means better fairness.) We also obtained JM = 8.80 packets for base IEEE 802.11 vs. a JM = 4.93 packets for our scheme. (Lower JM means lower throughput jitter.) We conclude that for this particular scenario, our scheme results in a 60-80% improvement in performance. Each flow in our scheme requests a re-negotiation of CTP once every 7 seconds, without perceived bandwidth smoothing. This is lower than the 14 seconds for the scenario in the previous section because the transmission rate is higher and hence the 100-packet inter-update interval is reached faster.

As in the case of the scenario in the previous subsection, there is a 28% packet drop rate in the case of base IEEE 802.11, but negligible drop rate using our scheme. Also as in the previous scenario, the mean throughput of base IEEE 802.11 is 15% *higher* during the period under measurement (all 3 flows are active) than our scheme. This is because of three reasons: (a) the flows are pumping data into the network as fast as possible in order to get as much throughput as they can in the base IEEE 802.11 scheme while we are



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 6.3: Comparative throughput performance of Base and Enhanced IEEE 802.11 for 3-flow scenario with identical bandwidth requirements.

using rate control, (b) our TBE is configured to return a conservative estimate for $B_p(f)$, and (c) in the Dynamic Bandwidth Management scheme, the re-negotiation messages between the various RAs and the BM consume some network bandwidth.

The conservative $B_p(f)$ estimate was used to minimize packet drop rate. The cost of using such a conservative estimate is that our enhanced IEEE 802.11 scheme *under-utilizes* the network. Mean throughput is less than it would be under full network utilization. However, the TBE's estimate can be suitably tuned so that throughput of our scheme approaches that of the base IEEE 802.11 scheme and network utilization increases. On the other hand, this will also increase the packet drop rate of our scheme and thereby degrade performance as packets are dropped randomly from flows. So, there exists a trade-off between throughput and packet drop rate.

Inter-update period (pkts.)	FM (pkts.)	JM (pkts.)	Overhead (requests/flow/sec.)
50	3.62	4.37	0.5
100	4.06	4.66	0.143
150	4.15	4.93	0.059
200	4.18	5.10	0.019

Table 6.1: Effect of $B_p(f)$ inter-update period on performance and overhead.

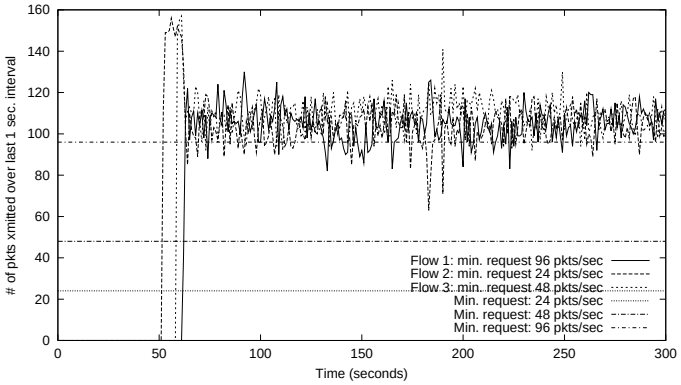
Tolerance level (%)	FM (pkts.)	JM (pkts.)	Overhead (requests/flow/sec.)
10	3.22	4.36	0.333
15	4.06	4.66	0.143
20	4.89	5.19	0.056
25	5.77	5.37	0.026

Table 6.2: Effect of various $B_p(f)$ variation tolerance levels δ on performance and overhead.

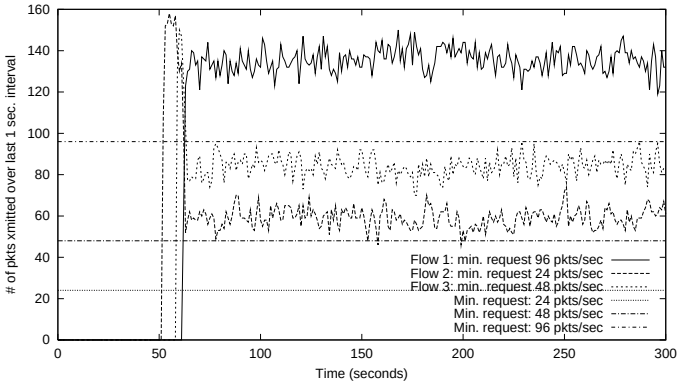
In addition to the perceived bandwidth smoothing described in the previous section, we now discuss two other methods to minimize re-negotiation overhead and hence the network bandwidth re-negotiation consumes. One method is to increase the inter-update period between successive perceived bandwidth updates from the TBE to the RA. Recall that we use 100 packets as the default inter-update interval in our experiments. Table 6.1 shows how overhead and performance vary with different inter-update intervals. As the inter-update interval increases, some changes in perceived bandwidth go undetected and cannot be responded to. Hence, the fairness and throughput jitter worsen while the overhead improves. The overhead is measured as the frequency of re-negotiation requests per flow. The threshold tolerance to perceived bandwidth changes was set at the default of δ = 15% for this experiment.

The other method to reduce re-negotiation overhead is to increase the tolerance to changes in perceived bandwidth $B_p(f)$. Recall that we define significant change as a δ = 15% change in perceived bandwidth. If we define significant change as, say, a δ = 25% change, then we can reduce re-negotiation overhead because the RA now waits longer and tolerates more $B_p(f)$ fluctuation before initiating re-negotiation. Again, this worsens the performance of the system because fidelity to bandwidth variations is reduced. Table 6.2 shows how overhead and performance vary with different levels of tolerance to $B_p(f)$ variation. The inter-update interval was set to 100 packets for this experiment. Tables 6.1 and 6.2 both show that for a small price in terms of performance, we can obtain large gains in overhead reduction.

6.1.3 Additional UDP Performance Results



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 6.4: Comparative throughput performance of Base and Enhanced IEEE 802.11 for 3-flow scenario with different minimum bandwidth requirements.

In this section, we present results for two additional scenarios: (a) when the flows have different minimum bandwidth requirements and (b) when the arrival time of the flows is staggered. We use the 6-node, 3-flow scenario used in the previous section, with the default perceived bandwidth tolerance of δ = 15% and the default inter-update interval of 100 packets.

Figure 6.4 shows the comparative base IEEE 802.11 and enhanced IEEE 802.11 throughput performance when the 3 flows each have different minimum bandwidth requirements. The minimum requirements of the 3 flows are 100 Kbps, 200 Kbps and 400 Kbps respectively. The maximum bandwidth requirement, 600 Kbps, is the same for all 3 flows. The plots show that while no guarantee can be made with base IEEE

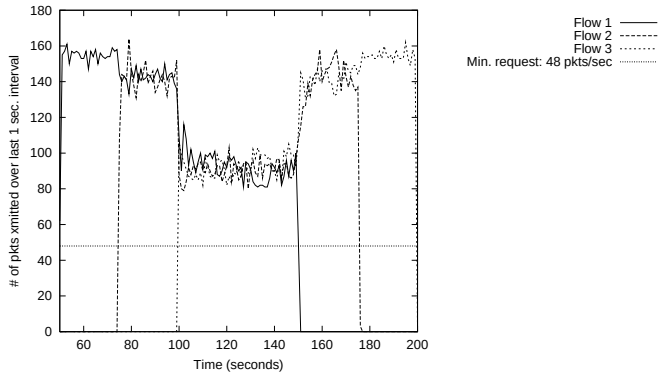


Figure 6.5: Enhanced IEEE 802.11 performance for 3-flow scenario with staggered start times.

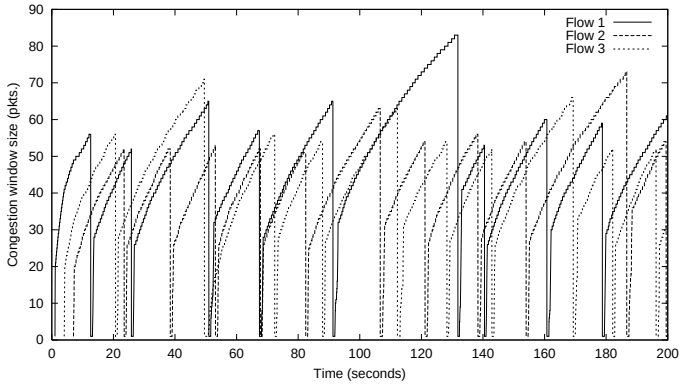
802.11, we can make coarse guarantees with our scheme.

While in all our previous scenarios, all participating flows started at around the same time, Figure 6.5 shows the throughput performance of the enhanced IEEE 802.11 scheme when the start times are staggered. All simulation parameters are identical to those in section 6.1.2, except the staggered start times and the length of the simulation run, which is set to 200 seconds. The bandwidth requirements are identical for all 3 flows, as in Section 6.1.2. This plot is similar to Figure 1 from [4] and Figure 11 from [28], which were for a AP-based wireless network with centralized scheduling. We have produced a similar effect for a single-hop ad hoc network that uses the IEEE 802.11 protocol's DCF.

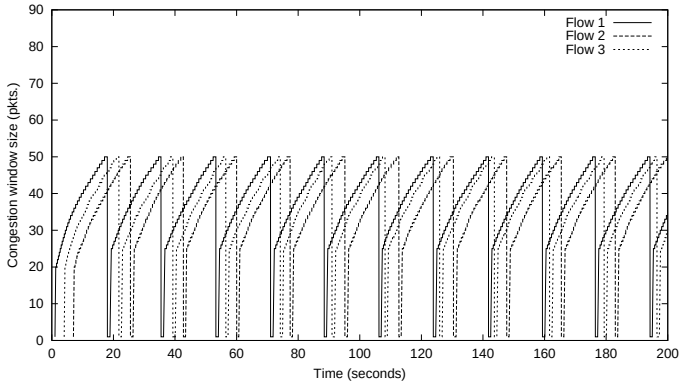
6.1.4 TCP Experiments

So far our simulation experiments have focused on multimedia applications and UDP flows. In this section we investigate the behavior of TCP flows and their interactions with the BM scheme. In general, it is more interesting to study the effect on the BM scheme on TCP's working, than it is to study the effect of injecting TCP traffic on overall throughput characteristics. This is because our traffic shaper converts TCP traffic into CBR traffic, with rate dictated by our BM. Thus, injecting TCP traffic is just like adding more CBR traffic. The effect of traffic shaping on the TCP flow however, is more interesting to study.

We simulate three TCP flows, each running between different nodes, in a single-hop ad hoc network managed by a BM, i.e., using enhanced IEEE 802.11. TCP traffic is best-effort and elastic, so $p_{min}(f)$ is set to zero and $p_{max}(f)$ to 100%. As mentioned in Chapter 3, different $B_p(f)$ values derived from the same normalized bandwidth estimate are used for data and acknowledgements, due to their different packet



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 6.6: TCP congestion window behavior when interface queue size is smaller than congestion window limit.

sizes, when obtaining their respective CTP requirements. The size of the network interface queue is 50 packets, and the maximum congestion window size for a TCP flow is 128 packets. The experiment lasts 200 seconds. While for the UDP experiments, rate-control using the RA is done in the UDP application, in the TCP experiments, leaky bucket rate-control is done per-node at the network interface queue. The interface queue only releases packets at the rate allotted by the BM.

Figure 6.6(b) shows the congestion window sizes of the three TCP flows, in the enhanced IEEE 802.11 case. They each expose the same behavior: the window size increases each time to 50 packets, cuts back and the cycle repeats. This behavior is due to TCP's additive-increase multiplicative-decrease (AIMD) congestion control algorithm, where the congestion window size will decrease *only* when a packet loss

Scheme	FM (pkts.)	JM (pkts.)	Pkts. dropped	T'put (total acks recvd.)
Base IEEE 802.11	6.70	9.40	565	45065
Enhanced IEEE 802.11	2.12	2.39	33	35698

Table 6.3: Performance and throughput loss comparison using TCP with interface queue size smaller than congestion window limit.

Scheme	FM (pkts.)	JM (pkts.)	Pkts. dropped	T'put (total acks recvd.)
Base IEEE 802.11	6.53	8.50	0	33804
Enhanced IEEE 802.11	2.51	2.72	0	26577

Table 6.4: Performance and throughput loss comparison using TCP with interface queue size larger than congestion window limit.

event is encountered. Packet loss occurs only when the queue overflows, because of co-ordinated channel access ensured by the RA. Queue overflow occurs only when congestion window exceeds the maximum queue size. A TCP flow will keep increasing its congestion window size up to the queuing limit. In fact, this "probing" of congestion window size is TCP's way of aligning itself to the available bandwidth of the network. Without knowing the BM's allocated rate for this node, a TCP flow has to fill the router queue before it cuts back its congestion window size, which incurs unnecessary long queuing delay for the packets. However, this behavior does not forfeit its allocated bandwidth, as TCP always keeps the queue non-empty.

As comparison, we run the same TCP experiments over a single-hop ad hoc network without the bandwidth management, i.e., using base IEEE 802.11. Figure 6.6(a) shows that the congestion window sizes of the three flows follow the same "saw-tooth" pattern as in Figure 6.6(b). But the maximum window size that each flow can reach may not be exactly the same, because of the unfairness in the channel access, and hence in time to first packet loss, for each queue. Unmanaged release of packets from the queue results in unequal congestion window growth and causes unfairness. As a result, the fairness metric (FM) and jitter metric (JM) of the flows deteriorates, and the number of dropped packets are significantly larger than that in the BM managed scheme, as shown in Table 6.3. The total number of dropped packets is greater in the base IEEE 802.11 case because an entire window of packets may be dropped at a time before TCP resets its congestion window size, whereas in the enhanced IEEE 802.11 case, a single packet loss results in window reset. The overall throughput of the TCP flows in the enhanced IEEE 802.11 case, however, is smaller than that in the base IEEE 802.11 scenario. This is similar to the result for UDP flows as shown in Section 6.1.2. We also experimented with less conservative $B_p(f)$ estimates, which resulted in a decrease

in throughput disparity between the base and enhanced IEEE 802.11 cases, at the cost of some performance deterioration. Thus the $B_p(f)$ values can be used to trade-off performance (as measured by the FM and JM) and throughput loss, as with the UDP experiments.

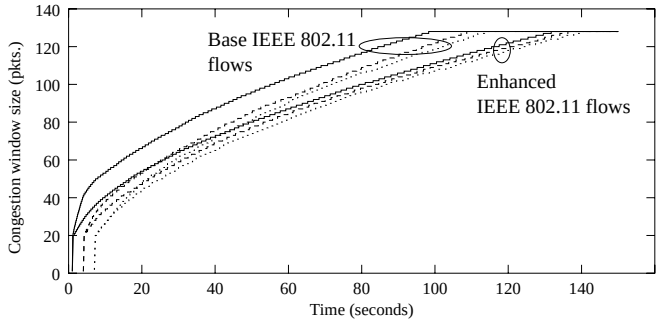


Figure 6.7: TCP congestion window behavior when interface queue size is larger than congestion window limit.

Another scenario of running TCP over BM is setting each node's interface queuing limit to be larger (150 packets) than the congestion window limit (128 packets) of a TCP flow. We run the experiments for this scenario for 150 seconds. TCP's congestion window size can never reach the maximum interface queue size, and hence there is no packet loss as result of queue overflow. In this case, we can expect TCP's congestion window size to stay at its maximum limit without fluctuating, because there is no packet loss at the MAC layer either. Figure 6.7 shows this behavior. Note that the slow convergence speed of TCP's congestion window size does not impact its throughput efficiency, as the interface queue is kept non-empty at all times. However, in order to minimize queuing delay, it is advisable to set TCP's congestion window limit to a small value when running over a bandwidth managed network. Table 6.4 compares the fairness performance and throughput loss for the base and enhanced IEEE 802.11 scenarios for the case where congestion window limit is less than the interface queue size. From the plot in Figure 6.7, it is obvious that the throughput disparity, as a percentage, between the Base and Enhanced IEEE 802.11 cases in this scenario, decreases with time.

6.2 Testbed Experiments

We used our testbed experiments to evaluate the throughput performance and the request-reply delay overhead in the presence of both physical channel errors as well as contention from a limited number of ac-

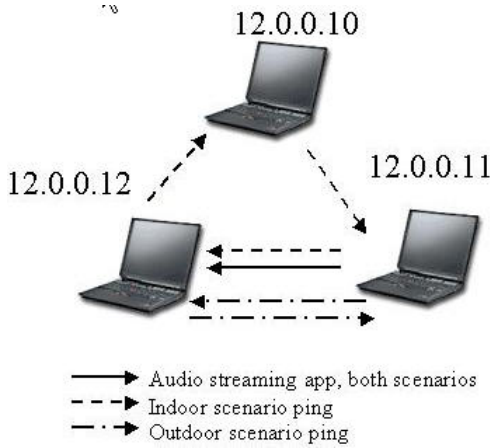


Figure 6.8: Single-hop ad hoc network testbed.

tive stations. Our testbed (see Figure 6.8) consisted of 3 IBM ThinkPad laptops, each equipped with an ORiNOCO PCMCIA 802.11b wireless card configured in peer-to-peer ad hoc mode.

We used a rate-adaptive CBR audio streaming application over UDP in our testbed experiments. We developed this Java application ourselves using the JavaSound libraries. The audio streaming application could operate at 5 different QoS levels between 32 Kbps and 256 Kbps depending on the available channel capacity perceived by the TBE. At the maximum QoS (256 Kbps), all audio samples were transmitted while at lower levels fewer samples were sent, and the audio was reconstructed through interpolation at the receiver. The purpose of the testbed experiments was to study the feasibility of our scheme in a testbed with a realistic single-hop ad hoc network environment. The RA in the application and the TBE communicated via the `/proc` interface.

6.2.1 Throughput Performance

We conducted two throughput experiments, one *indoors* and one *outdoors*. In each case, we started some unmanaged ping sessions, as shown in Figure 6.8, to bring about contention. The ping ICMP packet transmission on the channel also artificially reduced its bandwidth so that the bandwidth perceived by the audio streaming application actually fluctuated between 32 and 256 Kbps depending on the physical errors. In the absence of the pings, the reduction in perceived bandwidth brought about by the physical errors alone was not sufficient to cause the audio streaming application to adapt its quality. The physical errors, at their worst, reduced the perceived bandwidth by a few hundreds of Kbps. Given a 2 Mbps channel and an application

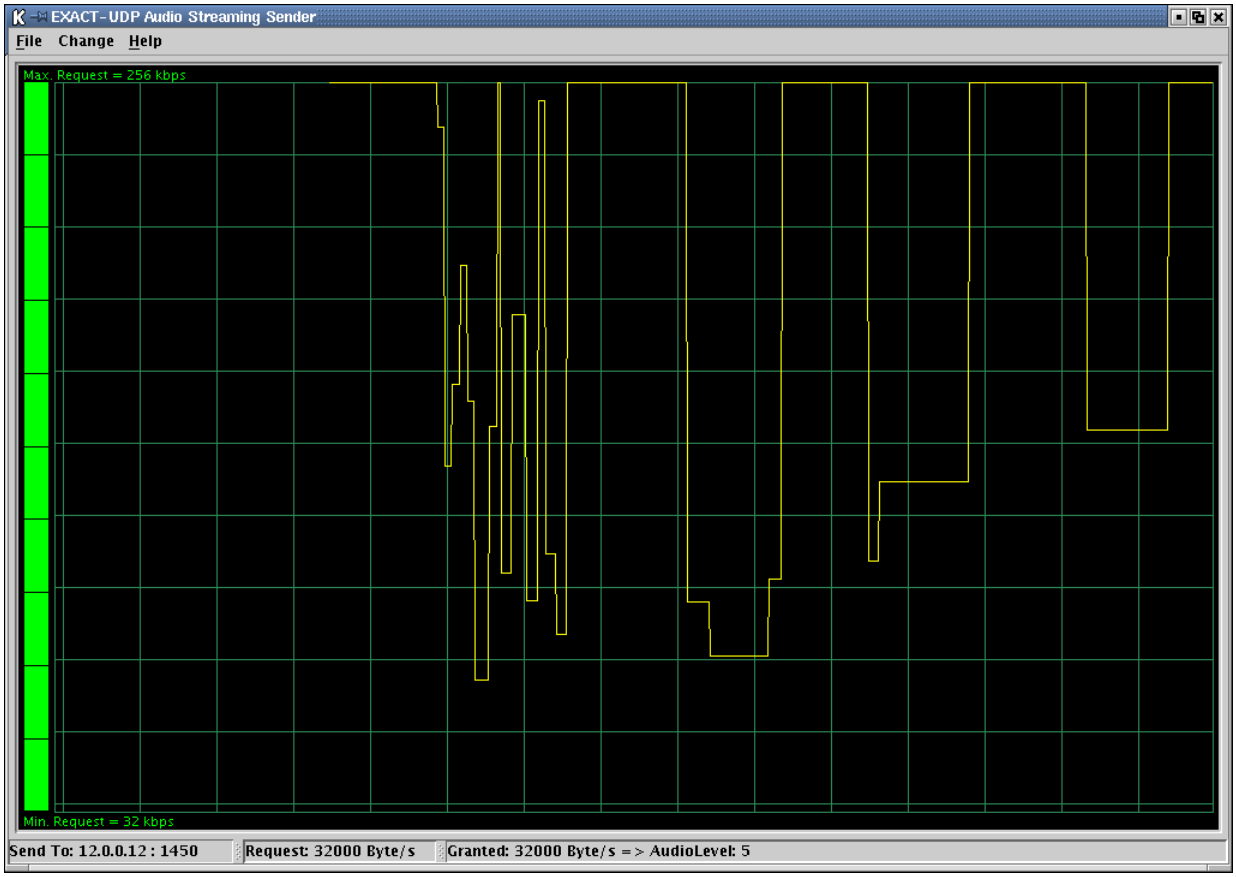


Figure 6.9: Indoor testbed experiment plot.

with a peak rate of 256 Kbps, these errors thus had no effect on the application. Its quality level did not fluctuate. To bring about adaptation on the part of the application, the physical errors had to vary the available channel capacity for the application between 32 and 256 Kbps. Hence, we used the pings to contend with the application for the channel and thus artificially reduce the available channel capacity it perceives to the necessary range. The pings brought down the available channel capacity to around 500 Kbps so that fading and interference errors could then reduce it further below the 256 Kbps threshold needed for the application to adapt.

Figure 6.9 shows the throughput performance for the *indoors* scenario. On the x-axis is time in 45 second units. The y-axis shows the adaptation of the audio streaming application, between 32 and 256 Kbps, to the change in available channel capacity. The channel bit-rate was fixed at 2 Mbps at the network cards. The perceived bandwidth variation-tolerance was set at δ = 15% and the inter-update interval was 100 packets. The BM was located on the same machine as the sender, 12.0.0.11.

The flurry of re-negotiations with the BM on the left hand side of the plot corresponds to our moving the sender (12.0.0.11) down to a secluded portion of the basement of the building while the receiver (12.0.0.12) and the third laptop (12.0.0.10) remained in the lab on the second floor. While in the basement, the sender moved around, down narrow corridors, over staircases and through fire doors. As the level of fading and interference changed drastically, the perceived channel capacity also changed drastically and hence the flurry of channel time re-negotiations. The contending pings also were affected by the physical errors and produced variable contention, thus inducing even greater instability in the application QoS.

We then brought the sender back to the second floor, the perceived bandwidth returned to around 500 Kbps, and the quality of the audio returned to its maximum. We then placed the sender and receiver next to each other so that physical errors were rare. The 3 dips in the graph on the right hand side correspond to experiments with no physical errors, but 3 different levels of contention due to 3 different ping rates. All 3 of these ping rates were greater than those used for the first part of this experiment. In the first part, the pings reduced the available channel capacity to around 500 Kbps and the physical errors dragged it further down. In this part there were no physical errors, but the larger ping rates themselves took the available channel capacity below 256 Kbps, causing re-negotiation from the application. The purpose of this experiment with no physical errors was to demonstrate the effect of the contending ping sessions: they produce a reduction in the perceived available channel capacity of the managed audio streaming application, in a controlled fashion, and the reduction is a constant one.

Next, we performed another set of experiments *outdoors*. The channel bit-rate was set at 5.5 Mbps for this experiment. As before, we had pings produce contention to artificially reduce available channel capacity for the audio streaming flow. Other parameters such as the value of δ and the inter-update interval were the same as in the indoor experiment. In the outdoor scenario, we used only two of the laptops. The BM was once again co-located with the sender, 12.0.0.11. At the start of the experiment, the sender 12.0.0.11 and the receiver 12.0.0.12 were next to each other on the sidewalk of a street. Then, keeping the receiver 12.0.0.12 stationary on the sidewalk, the sender 12.0.0.11 was moved away by a person walking at a normal pace down the street on the sidewalk. When the sender was around 150 meters away, the available channel capacity perceived by the audio flow began fluctuating due to signal fading effects. This resulted in a flurry of re-negotiations shown in Figure 6.10. The sender then wandered for a while around the point 150 meters away before returning to the starting position. As the sender moved closer to the receiver, at one point,

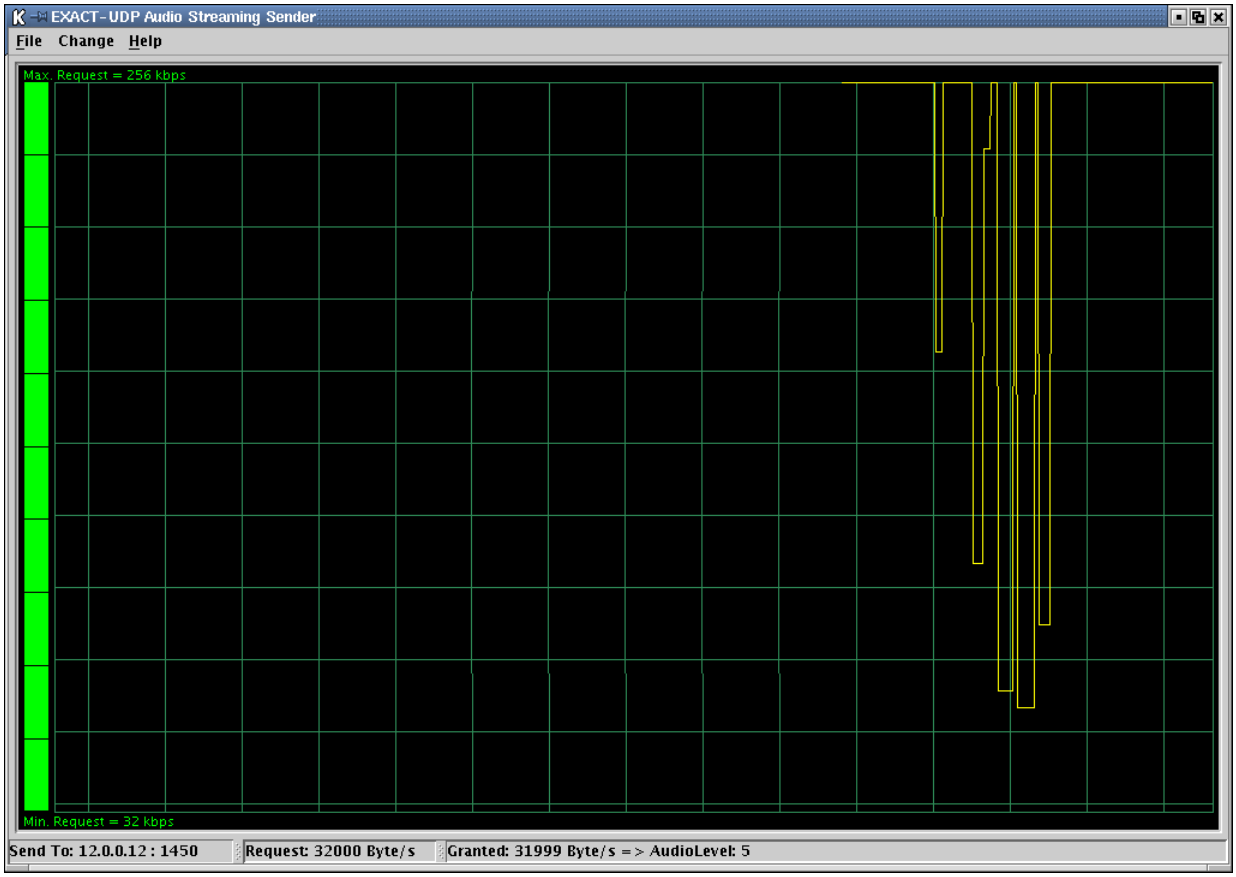


Figure 6.10: Outdoor testbed experiment plot.

the available channel capacity returned to its ping-induced constant level and the application returned to its highest quality level.

In both the indoor and outdoor experiments, increased signal fading effects cause the perceived channel capacity to fluctuate rapidly, instead of staying uniformly low. This is because a smoothed channel capacity estimate is not used in these experiments. Furthermore, the loss-rate threshold λ used to signify a broken link is set at 100%. We actually designed these two improvements to the channel capacity estimation mechanism that were suggested earlier in Chapter 3, based on these indoor and outdoor testbed experiments.

We repeated our experiments using ARS (auto rate selection) feature of the wireless card, instead of using constant rates 2 Mbps and 5.5 Mbps mentioned above. Our results were very similar when using ARS as compared to when using fixed rates. The ARS feature (also called multi-rate communication) involves changing the 802.11 transmission rate at the physical layer in response to poor SNR on the channel. Since we measure the channel capacity at the MAC layer, our scheme works independent of the ARS feature: we

measure the capacity available to the flow at the transmission rate chosen by the ARS feature in the physical layer. Our measured channel capacity is always a little lower than whatever is chosen by the physical layer, because of the overhead of the MAC protocol. We also experimented with the BM at the destination node, with no change in performance. The request-reply delay overhead for re-negotiation requests does not affect performance much because the application parallelly continues transmitting at its previously allotted CTP until the re-negotiation reply arrives, a few milliseconds later.

6.2.2 Request-Reply Delay

The request-reply delay is the time delay between the sending of a `request` message and the receipt of a `reply` message. All our control messages had a 32-byte payload. This exchange of messages occurs both during flow establishment as well as when perceived bandwidth changes significantly. We set the bandwidth of the network to be 5.5 Mbps, as in the case of the outdoor experiment. We used all 3 laptops for the request-reply delay experiments, with 12.0.0.11 being the sender, 12.0.0.12 being the receiver and the BM being located on 12.0.0.10. We found that, if there is no contention, each request-reply round-trip took 23 ms on average. In the presence of ping-induced contention, each request-reply round-trip took 61 ms on average. Flow establishment occurs only once per flow, obviously, and if the perceived bandwidth does not change much, then the 20-60 ms request-reply delay is a small one.

6.3 Pricing Algorithm Experiments

Our price-based channel time allocation algorithm is a *variable price* algorithm because the system price changes with time according to the "richness" of the users in set F. In this section, we evaluate the performance of this algorithm. We also compare it with two *fixed price* schemes and with a traditional Vickrey auction.

1. In the *fixed price proportional* (FPP) scheme, each user f is initially allotted CTP $p_a^r(f) = D(f)/\rho$ where ρ is the fixed system-wide price, i.e., the flat system price irrespective of number of users and their bids. If $\sum_{f \in F} p_a^r(f) > 100\%$, then the allocations $p_a^r(f)$ are scaled down proportionately so that their sum equals 100%. If, for any user, $p_a^r(f) > p_{max}(f)$, then $p_a^r(f) = p_{max}(f)$. If, for any user, $p_a^r(f) < p_{min}(f)$, then this user is blocked on the grounds of insufficient budget.

2. The *fixed price welfare-conscious* (FPWC) scheme greedily allocates CTP, while charging fixed price ρ, to the users with the lowest maximum requirements. Users with larger maximum requirements get blocked, once 100% of the CTP has been allocated. Users who have too small a budget to even satisfy their minimum requirements, at the fixed price ρ, are also blocked.

3. The Vickrey auction [23] is a traditional auction by which the highest bidder gets the goods and pays the bid of the second highest bidder as a cost. In the multi-unit variant of Vickrey auction, the highest bidders are successively awarded their maximum resource requirement, or the remaining resource, whichever is smaller, until the resource is exhausted. The payment of each "winner" is decided by its *opportunity cost*, i.e. from the bids of those bidders who have been denied the resource either in part or to the full extent of their maximum requirement. The price falls back to the reserve price if a winner's allocation is greater than the total resource request denied over all users. Vickrey auctions have been used previously in wireless bandwidth allocation, and an example Vickrey auction can be found in [29]. In a Vickrey auction, the price paid for the resource varies not only with time as the "richness" of the users changes, but also from winner to winner, as the opportunity cost of each winner's allotted resource is different. In contrast with both the variable and fixed price schemes, the price of every single unit of allotted CTP is not the same in a Vickrey auction. The Vickrey auction hence forgoes the *bid-proportional fairness* property of the FPP, FPWC, and our variable price schemes by which a user gets CTP in proportion to its overall bid $D(f)$. In our experiments, we simulate the Vickrey auction by sorting users by their $mp(f)$ values and consecutively allocating the maximum requirements to users with the highest values of $mp(f)$ until we run out of CTP. The opportunity costs are determined from the $mp(f)$ values of the denied users and from the reserve price. Ties in $mp(f)$ are resolved in favor of the user with the lower $p_{max}(f)$. If an allocation via the Vickrey auction is less than a user's minimum requirement $p_{min}(f)$, then this user is allocated zero CTP and removed from F, but it is still included in the computation of the opportunity cost.

Our simulation scenario consists of 100 users with random arrival and departure times in a 5-hour time window. The minimum CTP requirements - combined uplink and downlink - of the users ($p_{min}(f)$) are uniformly distributed in the range 0% to 2%, corresponding to 0 to 40kbps for a 2Mbps channel. The maximum CTP requirements - again, combined uplink and downlink - are uniformly distributed in the range 2% to 10%, corresponding to 40kbps to 200kbps for a 2 Mbps channel. The value of $mp(f)$, in

Algorithm	Price ((c/min)/%CTP)	Revenue (c)	Avg. Sat.	Chan. Util.	Users Blocked
Our Pricing Scheme		19617	71%	83%	24
FPP	0.2	4496	68%	80%	24
	0.75	15718	59%	74%	24
	1.5	21410	38%	51%	26
FPWC	0.2	3750	94%	69%	50
	0.75	13245	66%	64%	40
	1.5	20766	38%	50%	27
Vickrey Auction		5204	99%	72%	69

Table 6.5: Comparison of our scheme, fixed price schemes, and Vickrey auction based channel time allocation algorithms.

(cents/min)/%CTP, for each user f is randomly chosen from the set $\{0.1, 0.2, ..., 1.0\}$. The reserve price for the network is set at 0.1 (cents/min)/%CTP.

Figure 6.11(b) is a plot of the number of users requesting service and the number of users admitted, using our price-based scheme. The remaining users are blocked. For the sake of simplicity, we assume that blocked users do not return and request service later. (Alternatively, we can assume that a user arriving later is actually a returning user, with different parameters.) Figure 6.11(a) shows the variation in price as users arrive and leave. The average price for the entire simulation run is 0.75 (cents/min)/%CTP. Figure 6.11(c) illustrates the mean satisfaction of the admitted users. We measure satisfaction, as a percentage, for a user f as: $\frac{p_a(f)}{p_{max}(f)} \cdot 100$. Since this measure does not account for rejected users, we also keep track of the blocking factor. Figure 6.11(d) is a plot of the channel utilization, in percentage, over the course of the simulation. As more users are admitted, channel utilization increases but the individual users are allocated less CTP, so their satisfaction falls. The mean user satisfaction and channel utilization, averaged over the entire 5-hour simulation run, are shown in the "Our Pricing Scheme" row of Table 6.5. The "Revenue" column of Table 6.5 shows the total revenue earned at the end of the 5-hour simulation run.

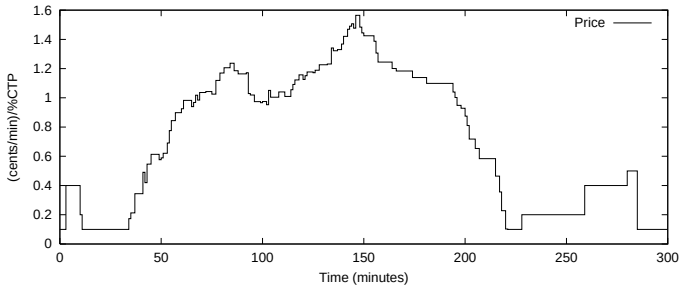
Table 6.5 presents the results of the comparison between our variable price channel allocation algorithm, the fixed-price algorithms, and the multi-unit Vickrey auction. For lower fixed prices, the revenue for the system is lower, but the mean user satisfaction and channel utilization are higher. This is because at lower prices, the users' budgets can buy them large CTPs. At higher prices, the revenue is higher, but mean user satisfaction, channel utilization and the blocking factor are worse. Our scheme attempts to simultaneously optimize all the performance parameters. While there may be prices for which the fixed price schemes

slightly outperform our scheme under one of the performance metrics, the corresponding penalty paid by the fixed price scheme in terms of the other metrics is large. For example, the FPP scheme, with a price of 1.5 (cents/min)/%CTP yields a 10% higher revenue than our scheme, but the mean user satisfaction is nearly halved, channel utilization falls from 83% to 51%, and blocking factor is also worse.

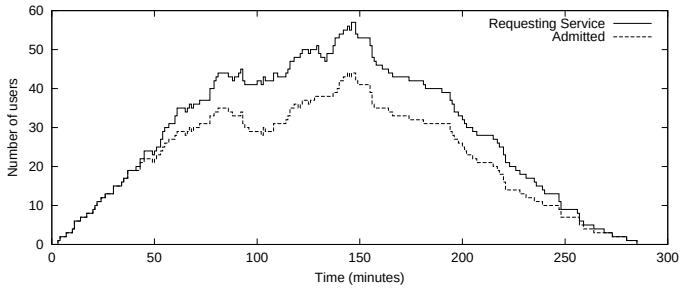
It should be noted that, even at the low fixed price of 0.2 (cents/min)/%CTP, the fixed price schemes do not necessarily perform better than our algorithm in terms of channel utilization and mean user satisfaction. This is because our reserve price is lower than 0.2 (cents/min)/%CTP. Thus, when our scheme defaults to the reserve price, it results in much better utilization and mean user satisfaction at those instants, which affects the overall mean utilization and mean user satisfaction comparisons.

Also note that the fixed price schemes cannot know the average worth of the channel time to the users before hand. But even if this knowledge *were* available, and the fixed price were set to this average worth (i.e., the average price of our scheme 0.75 (cents/min)/%CTP), the performance still does not match the performance of our algorithm.

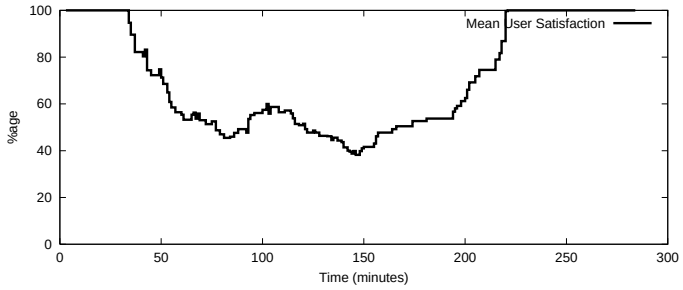
While the Vickrey auction modifies the price based on the "richness" of the contending users, it does not take advantage of the *flexibility* and *adaptive-ness* of the users. It is an *all-or-nothing* scheme which yields 100% satisfaction to all but one of the admitted users but also yields high blocking factor. Our scheme allows users to obtain 100% satisfaction if they outbid other users comprehensively, but also takes advantage of the adaptivity of network users and allots users with somewhat lower bids only a portion of their maximum requirement. The result is a trade-off between satisfaction and blocking factor, and an increase in revenue as users attempt to comprehensively outbid other users and obtain their maximum requirements.



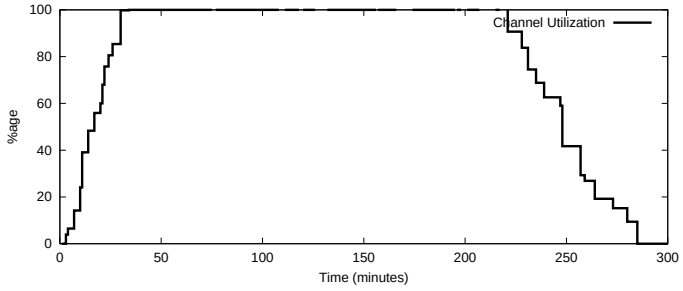
(a) Price.



(b) Blocking Factor.



(c) Mean User Satisfaction.



(d) Channel Utilization.

Figure 6.11: Performance evaluation of price-based channel time allocation algorithm.

Chapter 7

Bandwidth Management in Multi-Cell Environments

Wireless “hot spot” local area networks (LANs) consist of a group of mobile hosts connecting to the backbone wireline network via an *access-point* (AP). In order to increase coverage area, often multiple APs with overlapping service areas are employed. A mobile host can then roam seamlessly over the extended coverage area. At any time, the mobile host uses the AP in its vicinity that provides best signal-to-noise ratio (SNR) to connect to the wireline network. We term such an extended single-hop wireless network environment a *multi-cell* wireless LAN. In Figure 7.1, host 3 is initially in cell 1, using AP1 for its uplink transmissions. It then roams into cell 2, along the dashed line, and begins using AP2.

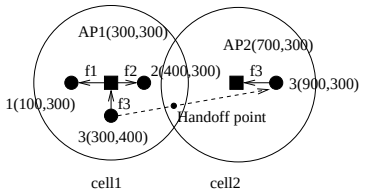


Figure 7.1: Multi-cell scenario.

Usually, to eliminate cross-cell interference, separate non-overlapping frequency sub-bands are assigned to adjacent cells in a multi-cell IEEE 802.11 LAN. However, as per the IEEE 802.11b specification [2], there can be only three such completely non-overlapping frequency sub-bands in the 2.4 GHz range. For complex geographical coverage regions, it may be impossible to assign non-overlapping frequency sub-bands to adjacent cells. Cross-cell interference can hence not be competely eliminated. Moreover, interference can also occur between nodes in non-adjacent cells because the interference range of 802.11 is much larger than the cell radius, which is equal to the 802.11 transmission range. Another factor that can affect interference

is the presence of multiple wireless LANs belonging to different providers, in the same geographical area, reusing the same frequencies. For illustration purposes, in this work we assume adjacent 802.11 cells (e.g. cells in Figure 7.1) use overlapping frequency sub-bands. This ensures that cross-cell interference *does* occur, and we show that our scheme works even in its presence.

In this chapter, we extend the concepts presented in the previous chapters (especially Chapter 3) in managing channel time in a multi-cell environment described above.

7.1 Bandwidth Management in Multi-Cell and Multi-Hop Environments

In the multi-cell environment, there is one central BM situated in the distribution system [2], as shown in Figure 7.2. It has a single flow table for the entire wireless LAN, i.e., bandwidth management is performed uniquely in the entire LAN. It is impossible to predict the interference in a cell from flows in neighboring cells. This depends entirely on the location of the individual mobile hosts. As explained in the previous chapter, however, the channel quality estimation scheme at the individual hosts takes into account the effect of interference from neighboring cells on the hosts' perceived channel quality.

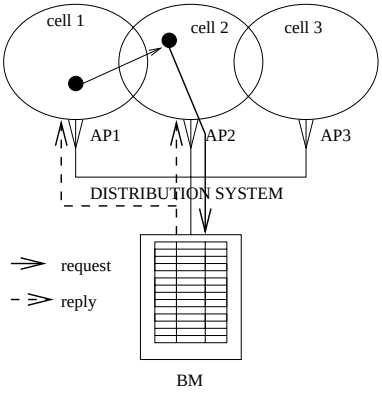


Figure 7.2: Bandwidth Management in a Multi-cell Environment.

When a mobile host moves from one cell to another (i.e., it detects re-assignment of its AP), it initiates a CTP `request`, specifying the ID of its new AP to the BM. The CTP requirements in this `request` message are computed with a default, possibly hardcoded, $B_p(f)$ and $L(f)$ estimates, as would be the case for a brand new flow. This is because the channel conditions are likely to be different in the new cell and need to be estimated from scratch.

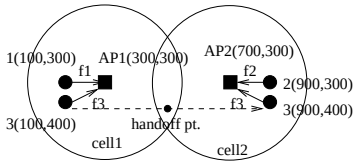


Figure 7.3: Spatial reuse.

A broadcast `reply` message may go to all cells, but this does not cause a problem, because it contains the flow-id of all flows whose allocations are being conveyed via the message. Even flows that are being cut-off, i.e. being allocated zero CTP, have their flow-ids in the `reply` message. Thus any flow that receives the broadcast `reply` message, but does not find its flow-id in the message, can just ignore the message.

In multi-cell wireless LANs, the phenomenon of spatial reuse enables simultaneous transmissions by different hosts in the network and thus increases channel capacity. For example, in Figure 7.3, flows $f1$ and $f2$ do not share the channel and their transmissions hence do not interfere with each other at any time. Two flows interfere with each other if one of the end-points of one flow lies within the interference range of the source of the other. The flow set F used in the bandwidth allocation algorithm in Chapter 5 is not unique in the network. There are multiple flow sets F in the multi-cell network. Each flow set consists of a set of flows that all interfere with each other. Bandwidth allocation must be done in all flow sets. In our scheme, the BM uses *location* information in computing the flow sets. The flow sets are *maximal cliques* of the flow interference graph [12, 30], as described in the next chapter. Piggybacked with the current channel quality estimate, each flow also periodically updates the BM of its current location, determined using GPS or some other location mechanism. A flow f may be a member of multiple flow sets, and multiple allocated bandwidths ($B_a(f)$) may thus be computed for it. The BM assigns it the *minimum* of these multiple allocated bandwidths. This ensures that it does not violate the fair share of any other flow in *any* set. Given n flow sets, the time complexity of the algorithm to compute the flow sets F and allocate bandwidth (algorithm below) in each is $O(n^2)$.

In Figure 7.3, host 3 is initially in cell 1 at the co-ordinates shown. At time 20 seconds into the experiment, it starts to roam towards cell 2. It reaches its final position shown 100 seconds after the start of the experiment. When it is in the initial position, flows $f1$ and $f3$ share channel time, but flow $f2$ can spatially reuse the channel simultaneously. In the final position, flows $f2$ and $f3$ share channel time, but flow $f1$ can spatially reuse the channel simultaneously. Between 26 and 94 seconds from the start, flow $f3$ simul-

taneously shares the channel with both flows $f1$ and $f2$, since host 3 is within the interference range of the source of both these flows. Initially there are two flow sets $\{f1, f3\}$ and $\{f2\}$. Between 26 and 94 seconds from the start of the experiment, the two flow sets are $\{f1, f3\}$, and $\{f2, f3\}$. The change in flow sets is detected by the BM when it receives the *first* update of location and channel quality, after time 26 seconds, from host 3. The BM detects that at this time, flow $f3$ interferes with both flows $f1$ and $f2$. Similarly, via its first update after time 94 seconds, host 3 indicates to the BM that flow $f3$ and $f1$ no longer interfere with each other. From then until the end of the experiment, the two flow sets are $\{f2, f3\}$ and $\{f1\}$. At any time, the allocation to a flow is computed within each flow set F of which it is a member, and the minimum of these allocations returned to it.

We now describe the algorithm used by the BM to ensure that the n flows in a particular flow set F that share a channel, obtain throughput proportional to their weights. It is a different, simpler, algorithm than those described in Chapter 5, since the aim in this chapter is not to develop an arbitration policy, but to indicate that using our architecture, we can effectively implement a given policy to achieve desired throughput levels for contending flows in a multi-cell environment. The policy used in this chapter is *bandwidth-fair* as opposed to *CTP-fair*, as in previous chapters.

We assume in this section that for all flows f, $B_{min}(f) = 0$. In case the minimum bandwidth requirement of a flow is greater than zero, and in case the allocated bandwidth is less than $B_{min}(f)$, then the flow is automatically rejected. Without loss of generality, we assume below however than $B_{min}(f) = 0 \forall f$, in the rest of this chapter. We also assume that the flow weights ($w(f)$'s) are in the ratios of the respective maximum bandwidth requirements, although weights can also be defined externally. If $\sum_{i=1}^{n} \frac{B'_{max}(f_i)}{B_p(f_i)} = \sum_{i=1}^{n} p_{max}(f_i) \leq 1$, the allocation is trivial: all flows can be allocated their maximum requirement, i.e., allocated bandwidth $B_a(f_i) = B'_{max}(f_i)$. The allocation algorithm is non-trivial when $\sum_{i=1}^{n} \frac{B_{max}(f_i)}{B_p(f_i)} = p_{max}(f_i) > 1$. The flows' allocations must be scaled down from the maximum, but remain in the ratio of the flow weights. The allocated bandwidths ($B_a(f)$'s) must then satisfy the following constraints:

$$\frac{B_a(f_1)}{w(f_1)} = \frac{B_a(f_2)}{w(f_2)} = \cdots = \frac{B_a(f_n)}{w(f_n)} \tag{7.1}$$

$$\sum_{i=1}^{n} \frac{B_a(f_i)}{B_p(f_i)} = 1 \tag{7.2}$$

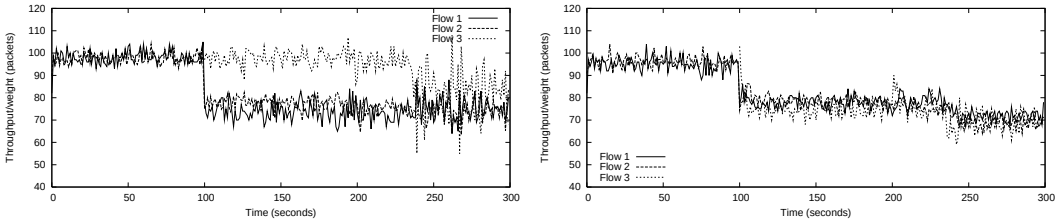


Figure 7.4: Throughput characteristics for single flow set scenario: (a) Base IEEE 802.11 and (b) Enhanced IEEE 802.11.

The former is the throughput fairness constraint and the latter is the channel capacity constraint. There is a unique solution to the above equations:

$$\forall m = 1, \cdots, n, B_a(f_m) = \frac{w(f_m)}{\sum_{i=1}^{n} \frac{w(f_i)}{B_p(f_i)}} \tag{7.3}$$

Thus, the BM can obtain the allocated rate for each of the n flows $\in F$. The allocation algorithm is run for all flow sets and an allocation obtained for each flow in each flow set. The minimum allocation for a flow, over all flow sets of which it is a member, is used for the flow's transmission rate, so it does not violate channel capacity in *any* flow set. The throughput fairness criterion may result in inefficient utilization of the channel because flows with lower channel capacity obtain a larger share of the channel to achieve their allocated throughput. We believe however that for high-bandwidth channels (e.g., 54 Mbps for IEEE 802.11g networks), fairness rather than efficiency will be the primary bandwidth allocation criterion. Our architecture is flexible in that any other type of fairness criterion can easily be incorporated in the BM, without affecting the other components of the system. Flows with hard minimum throughput requirements can be dropped ($B_a(f)$ = 0) by the BM if their fair share allocation falls below the hard minimum.

7.2 Multi-Cell Experiments

We illustrate the feasibility of our multi-cell bandwidth allocation architecture, and bandwidth-fair algorithm through simulations with the ns-2 network simulator [8]. We experimented with two scenarios in an IEEE 802.11 network with 2 Mbps theoretical channel capacity. We used UDP/CBR flows with application layer

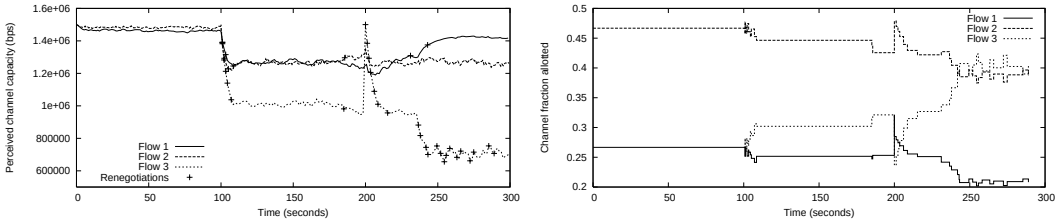


Figure 7.5: Channel characteristics with enhanced IEEE 802.11 for single flow set scenario: (a) Channel capacity estimate and (b) Channel fraction consumed.

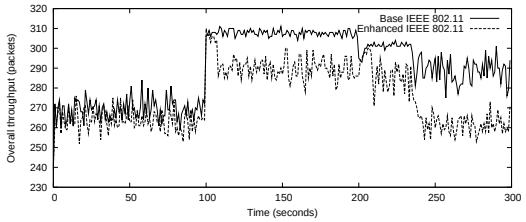


Figure 7.6: Overall throughput deterioration overhead in single flow set scenario.

packet size 512 bytes. The leaky-bucket traffic shaper injects packets at a constant rate into the network. In the first scenario, there is only a single flow set. We use this scenario to show the basic working of the algorithm. The second scenario contains multiple flow sets, and shows the working of our scheme in the presence of spatial reuse. In both cases, we compare our scheme (IEEE 802.11 enhanced with channel quality estimation and bandwidth management) with the base IEEE 802.11 scheme. Our two scenarios are comprehensive because they simulate the effects of channel contention, cross-cell interference (CTS suppression), uplink and downlink flows with unequal flow weights, mobility, traffic overload, and spatial reuse.

7.2.1 Single Flow Set Scenario

The single flow set scenario is shown in Figure 7.1. The $B_r(f)$ values of flows $f1$, $f2$, and $f3$ were 400 kbps, 700 kbps, and 400 kbps respectively, yielding weights in the ratio 1:1.75:1. The experiment lasted 300 seconds. Flow $f3$ starts 100 seconds into the run and its handoff from AP1 to AP2 occurs at time 200 seconds. Observe that, at all times, at least one of the end-points of each flow falls within the interference range (550 meters) of the source of all the other flows in the network. There is hence no spatial reuse

between the flows. The channel quality threshold that induces bandwidth reallocation is set at 5%. Using larger values for this threshold and for the update period reduces message overhead. Figure 7.4(a) is a plot of the throughput (packets received per second), normalized over flow weight, obtained by the flows, using base IEEE 802.11. In this experiment, each flow attempts to obtain its maximum rate. Figure 7.4(b) shows the normalized throughput obtained by each flow using IEEE 802.11 enhanced with our bandwidth allocation scheme. In our scheme, flows obtaining a higher throughput *forego* some channel time to allow increased throughput for other flows and improved fairness. However, the loss of throughput of the former (upto 20%) is greater than the gain in throughput of the latter (upto 6%), since flows with different channel qualities obtain different throughputs with the same amount of channel time. Figure 7.5(a) is a plot of the channel capacity estimate of each flow over time when using our scheme, with reallocation instances indicated. Initial channel capacity estimate is set to 1.5 Mbps at flow startup and handoff. Figure 7.5(b) shows the fraction of unit channel time each flow consumes. Since we use a bandwidth-fair allocation algorithm, the fractions consumed by the flows are different. If we had used a CTP-fair allocation algorithm, this plot would have had all flows coinciding, but the throughput plots would have revealed unfairness. Figure 7.6 compares the overall throughput of the network (total number of packets over all flows successfully transmitted per second) for the base and enhanced cases. This is a measure of the overhead of our scheme. Our scheme results in 5%-15% throughput deterioration, as we control flow rates and give more channel time to flows with poorer channel quality. One by-product of our scheme is that we achieve shorter queue lengths and very few instances of queue overflow, since we employ rate-control. Using the base IEEE 802.11 scheme, queue overflow is relatively common. Also, using our scheme, fewer packets are dropped at the MAC layer due to the RTS retransmit limit being reached.

7.2.2 Multiple Flow Set Scenario

The multiple flow set scenario is shown in Figure 7.3. We assumed all three flows shown had bandwidth requirement 800 kbps, i.e., flow weights were equal. The experiment lasted 120 seconds. All three flows began transmission at the start of the experiment. Host 3 began moving from its initial position after 20 seconds, and reached its final position 80 seconds later. The change in flow sets occured as described above. Figures 7.7(a) and 7.7(b) contrast the throughput performance of base and enhanced IEEE 802.11 for this scenario. (The steep fall in throughput of flow $f3$ around time 60 seconds in Figure 7.7(a) is because, after

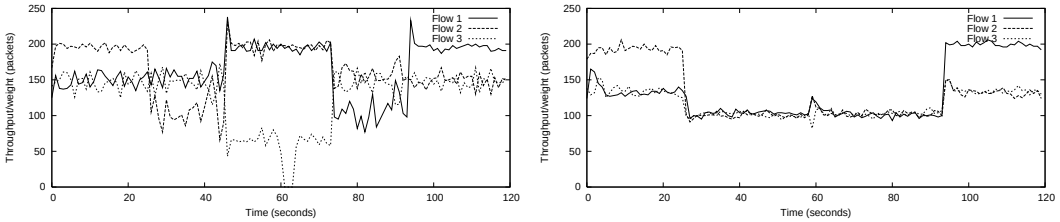


Figure 7.7: Throughput characteristics for the multiple flow set scenario: (a) Base IEEE 802.11 and (b) Enhanced IEEE 802.11.

handoff, host 3 takes some time to perform a successful RTS-CTS exchange in cell 2, in the presence of a high-bandwidth competing flow.) Again, our scheme maintains throughput fairness between the flows even as channel conditions vary due to host 3's movement.

Chapter 8

Bandwidth Management in a Static Multi-hop IEEE 802.11 Network

One possible application of the IEEE 802.11 MAC is in the field of wireless sensor networks. These networks consist of stationary or low-mobility, low-power, wireless nodes, sensing real-time data. The sensors may monitor natural phenomena such as temperature and humidity, or have video/audio capture hardware for surveillance. The sensed information may then be distributed via a publish-subscribe model. When a sensor is activated, it captures and transmits its real-time stream to an aggregation point or party interested in viewing the stream elsewhere in the network. The aggregation point or interested viewer may not be reachable directly, i.e., via a one-hop wireless transmission, from the sensor. The stream may have to be forwarded by multiple wireless hosts along the way from the source sensor to the destination viewer. Applications for such sensor fields comprising tens or hundreds of sensors include surveillance and climate-monitoring.

There are several problems in ensuring usable quality of the real-time stream transmitted to the viewer in this manner. Individual single-hop transmissions along a path are obstructed by contention in the network. The contention can be from other transmissions along the same path from source to destination ("self contention") or from transmissions along a different path. Contending traffic may interfere only with the transmission over the single hop, or only with the reception, or with both. Contention could result in unwanted delays, packet losses from both the front and back of the interface queues of transmitting nodes, and instability in end-to-end throughput.

In this chapter we address the problem of alleviating the effect of contention on the end-to-end flows in a multi-hop IEEE 802.11 wireless network. We estimate the effects of the contending flows on the individual single-hop transmissions of a particular end-to-end flow. Contention reduces the channel capacity available

to a particular end-to-end flow, and increases the packet loss rate. High traffic levels are undesirable because they result high contention. On the other hand, keeping traffic levels low results in poorer quality perceived by the end-user of the real-time data. Our problem is to balance these issues by determining optimum traffic levels. We address the *provisioning* problem: our aim is to maximize the aggregate utility of all the flows in the multihop wireless network, keeping in mind the observed available channel capacity and packet loss constraints. We also discuss modifying existing routing strategies so that the aggregate utility of the network flows can be further improved.

Our scheme still involves a centralized arbiter (BM) that takes flow requirements and channel quality at different points in the network as inputs. It determines an operational level for each network flow that yields globally optimal network performance. We assume in this chapter that the flows are co-operative, i.e., they do not violate their determined operational level. Flows are hence *guaranteed* a throughput corresponding to their allocated quality level, with negligible variation, and low delay and loss-rate.

8.1 Preliminaries

In this chapter, we assume a network grid topology consisting of a small number of nodes each equipped with IEEE 802.11 wireless network interfaces. The location of these nodes is fixed. Figure 8.4(a) shows an example of such a wireless grid with 25 nodes arranged in a 5 x 5 grid. Without loss of generality, we assume each node has transmission range of 250 meters and interference (carrier sense) range of 550 meters. In the grid in Figure 8.4(a), adjacent nodes are 200 meters apart along both the x- and y-axes. Each node is thus within the transmission range of nodes above, below, to the left of, and to the right of itself, if the respective nodes exist. Direct single-hop, transmissions are not possible to diagonally adjacent neighbors. The interference range of a particular node encompasses several nodes in its neighborhood. We also assume a 1 Mbps theoretical channel capacity in our example wireless grid. In Figure 8.4(a), two end-to-end multi-hop flows are also shown. We return to this figure later in the chapter.

Note that, as shown in [31], the transmission and interference ranges are not perfectly circular, as assumed by us. However, we assume that the ranges 250m and 550m are *pessimistic*: we assume that we may over-estimate the interference, but never under-estimate it. In general, the extent that we over-estimate interference depends on the Degree of Irregularity (defined in [31]), which in turn depends on the node density. In the scenario above where nodes are at least 200 meters from each other, the density is actually

very low, and hence the fact that the interference range is not perfectly circular does not play a big role. The ranges are typically skewed by no more than a 10-20 meters, so whether the interference range is 530 or 570 meters, a node 400 meters away is still inside, and a node 600 meters away is still outside. The accuracy of our results for this particular scenario is hence not affected by the radio range irregularity. In general, we may under-utilize the channel by using a pessimistic approach, but not overload the network and cause throughput, delay, and loss violations.

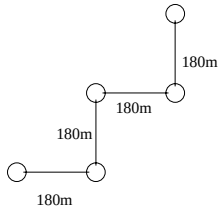


Figure 8.1: Interfering nodes 4 transmission hops away.

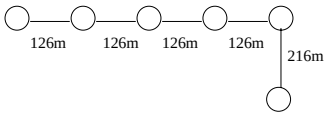


Figure 8.2: Interfering nodes 5 transmission hops away.

There are distributed schemes [18] where nodes rely on conveying MAC status information 3 hops away. It can be seen however, that the 3-hop flooding is overly optimistic. There exist scenarios - see Figures 8.1 and 8.2 - where interfering nodes are 4- or even 5-hops away. If MAC status is only transmitted 3 hops away, nodes that are 4- and 5-hops away will continue to interfere and not be accounted for. This may result in over-provisioning which can have serious consequences for the network as a whole. Both our centralized scheme and the distributed schemes suffer from the *asymmetry* problem described in [31]. However, in general, our pessimistic approach is better and safer for overall network performance.

Each wireless node in the network has a channel quality monitor in the MAC layer (i.e. 802.11 device driver), which estimates the effects of contention on its transmissions to each of its neighbors. The channel quality monitor is described in Chapter 3. It has an estimate B_p of channel capacity perceived and an estimate L of loss rate perceived by its transmissions along each of its outgoing links, i.e., separate values of B_p and L for each neighbor.

Note that in this chapter, each flow f has *multiple* channel quality estimates, one for each of its one-hop subflows. There is still one channel capacity per active single hop in the network, but flows are no longer

single-hop. The notation used for the channel capacity and loss rate estimates in this chapter is hence slightly different.

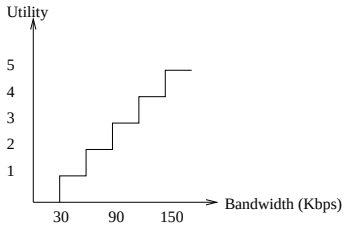


Figure 8.3: Stepwise flow utility function assumed in this chapter.

At an instant in time, we assume there exist k flows $F = \{f_1, f_2, ..., f_k\}$ in the network. Each flow has a stepwise utility function representing QoS of the real-time stream. Figure 8.3 shows a typical utility function for a real-time application that can operate at different qualities, e.g., different encoding rates for audio or different frame-rates for video. Stepwise utility functions [48] can also be obtained by quantizing continuous concave utility functions commonly used in QoS literature. A flow $f \in F$ has l possible operating bandwidth levels $B_1(f), B_2(f), ..., B_l(f)$, each corresponding to a unique level of utility to the flow's user. Note that the bandwidth requirements of the flow represent the bandwidth that should be received at the destination of the flow. The flow thus has l utility levels $Q = \{q(B_1(f)), q(B_2(f)), ..., q(B_l(f))\}$, each corresponding to a bandwidth requirement. In reality, each flow may have a different utility function, i.e. a different set Q. But without loss of generality, we assume in this chapter that each flow has the same utility function illustrated in Figure 8.3. We assume an end-to-end flow f encompasses s single-hop transmissions (s subflows). Obviously, for a flow f operating at utility level $q(B(f)) \in Q$, each single-hop subflow i has the same bandwidth requirement $B(f)$ as the overall end-to-end flow.

We still assume a centralized arbiter (BM) in the network that makes provisioning decisions. It determines operating QoS level for all flows in the network. The design of the BM, and the rationale behind centralized arbitration are discussed in Section 8.2. The BM is one of the more resource-rich nodes in the network. It knows the locations of all the nodes in the network. If a node moves, the BM is informed of its new location. The BM then determines *maximal interference cliques* based on node locations. The concept of maximal cliques representing subflow interference is well-explained in [12]. We represent one-hop subflows as points in a graph, and links between two points in the graph represent interference between the respective subflows. A maximal interference clique in such a graph represents a set of subflows wherein

each subflow interferes will all the others in the set. A maximal interference clique is a set of subflows (belonging to the same or different end-to-end flows) that share unit channel capacity. Only one subflow in the clique can have an active physical transmission on the channel at an instant in time. The maximal interference cliques determine *which* subflows in the network interfere with each other. A maximal interference clique comprises several subflows; a subflow can be a member of several maximal cliques.

The BM also receives periodically from the active interfaces their channel quality estimates (B, and L) to each neighbor. Since the channel quality is measured using data packets themselves, channel quality estimates are valid and relevant only for active interfaces. The channel quality estimate is a measure of *how* the subflows in a clique interfere with each other. The channel quality estimates are sent to the BM using a low-overhead distribution protocol. The overhead is controlled because only significant *variations* in channel quality, above a certain threshold, are reported. We measure the channel quality update overhead in an illustrative example in Section 8.3. All the nodes along a route periodically update the BM, thus improving reliability via redundancy, in case an update is lost. We also control loss rate via our scheme, so it is highly unlikely that all updates from a particular route will be repeatedly lost.

When a flow starts up, it transmits its utility function, and computed end-to-end source route to the BM using a reliable signaling protocol. Using the clique information, the channel qualities reported, and the requirements of the flows based on their utility functions, the BM determines an operating utility level for each flow in the network. The BM attempts to maximize the aggregate utility $\sum_f q(B(f))$ over all flows f in the network, where for a flow $f \in F$, $B(f)$ is the bandwidth receivable at the destination, and $q(B(f))$ is the allocated utility level according to the utility function. It keeps in mind that the bandwidth available to a particular subflow is dependent on the contention effects it experiences from flows in the various cliques of which it is a member. The utility level determination is modeled as a *multi-choice, multi-dimensional, knapsack* optimization problem (MMKP).

The BM returns the operating utility levels to the source nodes of the flows using the reliable signaling protocol. The sources generate traffic conforming to the allocated utility levels. The BM is invoked every time a new flow joins the network, leaves the network, or if a newly reported channel quality estimate from somewhere in the network causes a change in a flow's operating utility level. Although we currently use an existing distributed route-computation method in determining the end-to-end route for a flow, we eventually hope to have the BM provide route-computation suggestions to the routing protocol, taking into account

node locations and contention levels in various neighborhoods. This chapter addresses the provisioning problem assuming routes are known. Incorporating route-computation into the provisioning problem is left as future work. The provisioning problem needs to be addressed irrespective of how the routes are computed.

Note the differences in the multi-hop bandwidth management approach as compared with the single-hop approach: (a) In the multi-hop scenario, the translation of bandwidth requirements into CTP requirements is done at the BM, instead of at the clients, and (b) The BM attempts to maximize aggregate utility, instead of ensuring fairness/price-influenced fairness.

8.2 Centralized Arbitration

In our channel provisioning scheme, we use a centralized arbiter (BM). The functions of the BM are: (a) computing interference cliques, (b) receiving and tabulating channel state information, (c) receiving reservation/teardown requests and channel quality updates, (d) returning utility levels computed by the central optimization algorithm, and (e) providing routing hints. We use centralized arbitration because of the simplicity and low overhead involved. We assume the sensor nodes are constrained in terms of CPU power and memory available. Functions such as distributed clique computation, MAC channel state dissemination, resource-aware routing, distributed signaling, maintenance of reservation state, and distributed execution of MMKP optimization algorithm, can severely overload the resource-constrained wireless sensors. Communicating with a centralized BM in a multi-hop environment may appear inefficient at first, but for networks of a few tens or hundreds of sensors, the message overhead and signaling delay is quite acceptable, since all communication for a wireless node is streamlined to just one entity. (As shown in Section 8.3, the overhead is negligible so that no resource reservation is required for the reservation/teardown/update control packets.) All wireless nodes in our scheme only communicate with the central BM, whereas distributed clique computation [12] for example, requires a protocol in which every node communicates with each of its 1- and 2-hop neighbors, and makes a note of their state.

A centralized scheme does not scale well, but in any case, true scalability, as in the Internet, comes from hierarchy. Most practical multi-hop wireless networks do not involve routes longer than ten hops. We assume a relatively resource-rich node in the network can be made the BM. Usually, sensors are deployed in clusters, so a cluster-head can serve as the BM. In case this node fails, a new BM has to be elected. Alternatively, the BM can be a node not in the wireless network, but reachable with low delay from nodes

in the network. We now describe in detail the various functions of the BM.

8.2.1 Clique Computation

Two subflows are said to contend with each other if the transmitter of one is within the interference range of either the transmitter or the receiver of the other. A maximal interference clique is a set of subflows such that: (a) every subflow in it contends with every other, and (b) the addition of any subflow not originally in the clique, to the clique, violates property (a). Maximal interference cliques are used frequently in multi-hop wireless networks to determine which nodes contend for the channel [12, 30], and between which nodes spatial reuse of the channel bandwidth is possible. There already exist approximation algorithms for clique computation in multi-hop wireless network literature. We use one of these algorithms [30], which is based on locations, for clique computation, since we assume the location of the sensors is known at the BM. The main contribution of our work is not clique computation, which is a solved problem, but the novel usage of clique information in channel bandwidth provisioning, in conjunction with channel quality estimation. The clique computation algorithm divides the network into several maximal interference cliques. Since subflows in a clique share the unit channel capacity, the provisioning scheme must take into account this per-clique capacity constraint. It must also take into account the channel quality, which makes the channel capacity appear different to different subflows even in the same clique. This aspect is discussed below. Note that even though the nodes themselves are stationary, the cliques change with time, as flows arrive and depart, and contention-producing subflows are created and destroyed.

8.2.2 Use of Channel Quality Estimate

The translation from bandwidth to CTP requirements is the same as in Chapter 3. If a flow f needs end-to-end bandwidth $B(f)$ at a utility level $q(B(f))$, then to accommodate redundancy, we must reserve end-to-end bandwidth $B'(f) = \zeta(B(f), L_c) = \frac{B(f)}{1-L_c}$ where L_c is the *cumulative* loss rate for the end-to-end flow over all its subflows. In other words, $L_c = \prod_i L_i$ where L_i is the loss rate reported for subflow i of flow f. The bandwidth provisioned at utility level $q(B(f))$, for a particular one-hop subflow i, $1 \leq i \leq s$, assuming flow f has s subflows, is $B'(f,i) = \frac{B(f)}{1-\prod_{j=i}^{s} L_j}$. Essentially, the bandwidth provisioning for a particular one-hop subflow must take into account loss rate experienced by all succeeding subflows along the route. The bandwidth provisioning for the entire end-to-end flow must take into account the loss rate experienced by

all subflows along the route.

Now, if a subflow i of flow f has this modified bandwidth requirement $B'(f,i)$ at utility level $q(B(f))$, and its perceived available channel capacity is B_p, then in effect, it needs to spend a fraction $p_i(B(f)) = \frac{B'(f,i)}{B_p}$ of unit time on the channel in *each* interference clique it is a member of, to achieve its bandwidth requirement. We have thus converted the bandwidth requirement of the subflows into a requirement of *channel fraction* in each contention clique, using the channel quality metrics B_p and L. We have thus accounted for *which* subflows contend with each other for the channel and *how* they contend (i.e., how they affect each other's perceived channel quality), in determining the per-clique channel fraction requirement $p_i(B(f))$. In the previous subsection, we had described how the network is divided into maximal interference cliques. If a subflow is not a member of a particular clique, then obviously, it requires zero channel fraction in that clique. Now, we are in a position to combine the clique information and the per-clique channel fraction requirement into an optimization problem.

8.2.3 Optimization Problem

The goal of the central BM is to maximize aggregate utility of all network flows so that the sum of the channel fractions allotted to subflows in any clique does not exceed unity. Recall that the subflows in a clique can belong to the same or to different end-to-end flows. Recall also that one subflow can be a member of many maximal cliques.

Assume there exists a function $\beta(f,i,c)$ returning 1 if subflow i of flow f is a member of clique c, 0 otherwise. The optimization problem can be formalized as:

$$\text{Maximize} \quad \sum_{f \in F} q(B(f)) \tag{8.1}$$

$$\text{So that} \quad \sum_{f \in F} \sum_{1 \leq i \leq s} \beta(f,i,c) \cdot p_i(B(f)) \leq 1, \forall \text{ cliques } c \tag{8.2}$$

where $q(B(f))$ is the utility level selected for flow f. This optimization problem can be mapped onto a multi-choice multi-dimensional knapsack (MMKP) problem. Essentially, a vector of utility levels, one for each flow in the network, is a *choice*, i.e., a candidate solution. Each clique in the network is a *dimension* of the multi-dimensional knapsack of unit capacity. Each choice can be associated with a value (sum of utilities

for that candidate solution), and a vector of the loads contributed to each dimension of the knapsack. We have to find the choice with the highest value such that no dimension of the knapsack is over-full. The MMKP problem is known to be NP-hard. However, there exists a polynomial-time heuristic solution [32]. We implement this heuristic to solve the optimization problem. The algorithm returns a vector of utility levels that represents the optimal *choice*. The source of each flow is then sent its allotted utility level. Note that, if a utility level is associated with a flow then the same utility level applies to *all* its subflows. No state information or intelligence is then required at the forwarding nodes. If the bandwidth corresponding to even the lowest utility level of a particular flow cannot be satisfied, then this flow is returned a utility level of zero. The BM has *refused* to admit this flow. The arbitration scheme is thus also an *admission control* scheme.

8.2.4 Routing

When a flow starts up, it determines a source route to the destination using a standard ad hoc routing protocol such as DSR [33]. Before transmitting data along this route, it informs the BM of this source route, and of its utility function, and obtains the data *rate* that it can transmit at along the computed route. The source routes to the BM are also obtained using DSR. The route-computation and provisioning are *decoupled*, as in [5]: provisioning occurs over whatever route is computed by the routing protocol. Most routing protocols attempt to find only the route with minimum hop-count without taking available resources into consideration. The optimal route for resource provisioning may hence be a different route. A better approach to the problem is to use resource-aware route computation wherein resource constraints are considered in route computation [34, 35].

In our scheme, the BM is aware of node locations, and also of traffic bottlenecks, due to the channel quality reports. It is hence in an ideal position to modify the shortest-path route computed by the routing algorithm, reported to it during flow startup, since it has a global picture of the network. It must route around bottlenecks but at the same time not increase hop count by too much, otherwise a penalty in terms of longer delay is incurred [36]. In case contention cannot be avoided, then it may be better to incur contention at the transmitter rather than the receiver, since the latter may cause packet loss as shown in Figure 3.5. There also needs to be co-ordination among flows in reacting to bottlenecks via re-routing: if two flows both re-route around a bottleneck in the same direction, then the bottleneck gets merely shifted elsewhere. (This problem of co-ordination is another reason why centralized arbitration is more advantageous. Co-

ordinated, distributed, route computation imposes high overhead on individual wireless nodes.) Resource-aware routing in wireless networks is thus a non-trivial problem, especially if augmented with clique and channel quality considerations.

This chapter only addresses the provisioning problem over a given route. We hope to address resource-routing issues as part of our subsequent research. Provisioning by selecting proper operating quality levels and not violating capacity constraints, is integral to providing QoS guarantees (e.g., stable throughput, bounded delay), irrespective of how the route is determined. Our work is thus relevant even in the presence of resource-aware routing. The presence of resource-aware routing closely coupled with rate-assignment merely ensures that the best possible provisioned route can be chosen for the stream.

8.3 Illustrative Example

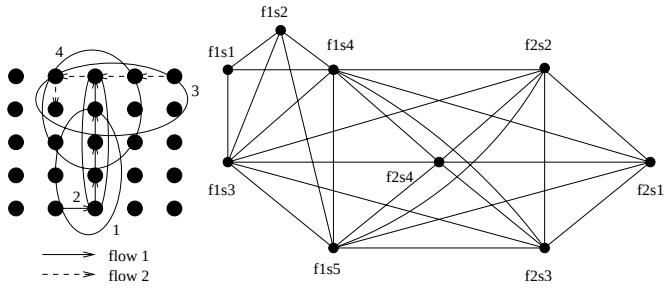


Figure 8.4: (a) Scenario used in our experiments. (b) Graph representation of subflows and interference.

In this section, we illustrate the working of our bandwidth provisioning scheme via simulations conducted with the ns-2 network simulator [8]. We used the network topology described in Section 8.1. We started two multi-hop UDP/CBR flows as shown in Figure 8.4(a), the first 2.5 seconds into the simulation and the second 56.3 seconds into the simulation. Both flows had the utility function shown in Figure 8.3. The subflows of each flow are numbered sequentially from source to destination. Figure 8.4(b) shows the graph obtained by representing each subflow in the network as a point, and interference between subflows as links. This graph has 4 maximal fully-connected subgraphs, i.e., 4 fully-connected subgraphs such that addition of a single point to the subgraph breaks the fully-connected property. Figure 8.4(a) shows the corresponding maximal interference cliques in the network topology, numbered 1 through 4, computed by the BM based on the node locations and subflows. When flow 1 is active, only cliques 1 and 2 exist. When both

flows are active, there are 4 maximal interference cliques as shown. The BM is located in the center node of the top row of the grid. The simulation run lasted 200 seconds.

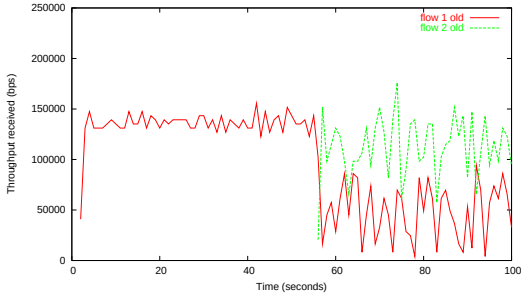


Figure 8.5: Base case throughput plot.

We first present the base case result. The base case involved no bandwidth provisioning. The flows transmitted at their highest rate, in order to obtain as much throughput as possible. The result was heavy contention, resulting in unstable throughput (no guarantee), untenable delay, and high loss rate. Figure 8.5 is a plot of throughput over the first 100 seconds of the simulation, in terms of data successfully received per second. The packet loss rate, combining MAC contention loss (see Figure 3.5) and congestion loss (packets dropped from the back of the FIFO queue at the source or intermediate host), was measured at 40%.

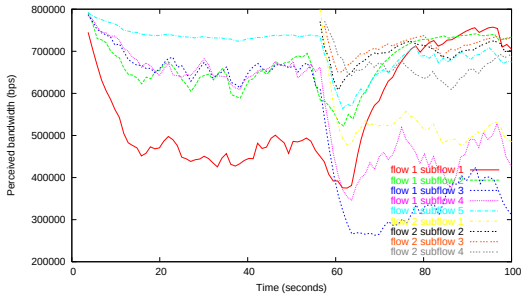


Figure 8.6: Plot of perceived channel capacity for each subflow in the example.

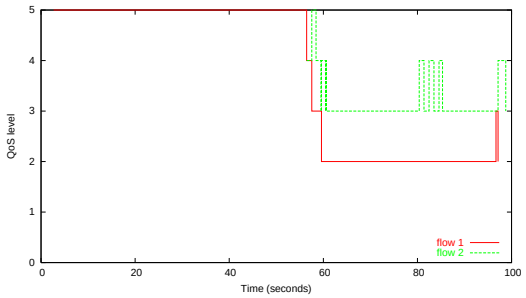


Figure 8.7: Utility level allocated to each flow in the example.

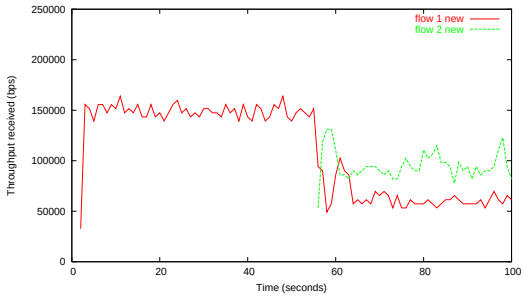


Figure 8.8: Throughput obtained by each flow in the example.

Next, we present the results with the same scenario, using our provisioning scheme. Figure 8.6 shows the B_p estimate for each subflow in the network in the first 100 seconds of the simulation. Figure 8.7 shows the allocated utility level for each of the two flows, based on the flow bandwidth requirements, channel quality estimates, and maximal interference cliques. Ideally, the allocated utility level to a particular flow should not vary too frequently, as it causes variation of throughput for that flow, which defeats the original aim of achieving throughput stability. In Figure 8.7 however, we show all the utility level jumps, without any smoothing. Figure 8.8 shows the throughput achieved by the two flows while transmitting at their respective allocated utility levels for the first 100 seconds. (The results were similar for the following 200 seconds also; we only show the first 100 seconds for more clarity.) There was some instability in throughput when flow 2 started, but apart from that instant, our scheme provided excellent throughput stability. We can thus provide throughput guarantees to the flows in the network: given a particular operating utility level for a flow, we can *guarantee* that the flow will *continuously* obtain network bandwidth corresponding to this operating utility level. Since we controlled contention, we were able to keep queue lengths short at all the forwarding nodes, and hence also achieved low end-to-end delays of 30-40 ms. The short queue lengths implied that packet loss from the back of the queue was eliminated. There was still some MAC contention loss (L; see Figure 3.5), but it was restricted to around 1%.

The overall number of packets successfully delivered, over the entire course of the simulation, was approximately the same (less than 0.5% difference) in both the base case as well when using our provisioning scheme, even though we controlled transmission rates in the latter case. In other words, we achieved the same network efficiency as the base case, while eliminating packet losses and contention delays, and while achieving throughput stability. This efficiency result illustrates the accuracy of our channel capacity estimation.

Renegotiation threshold	10%	15%	20%
Update frequency (updates/sec.)	0.86	0.63	0.51
Contention loss rate	0.87%	1.02%	1.13%

Table 8.1: Overhead vs. contention-control trade-off for various re-negotiation thresholds.

Once we established the performance of our scheme, the next step was to measure its overhead. Recall that if the channel quality values B_p or L for a particular subflow vary by a certain threshold, then the transmitter of this subflow needs to update the change to the BM. We varied the re-negotiation threshold between 10% and 20%, and measured the corresponding overhead in terms of frequency of channel quality updates. We also evaluated the contention-control versus overhead trade-off. The results are tabulated in Table 8.1. As the re-negotiation threshold was slackened, overhead in terms of update frequency gets reduced, but our control over contention in the network was also loosened. The update overhead of one update packet received by the BM (from anywhere in the network) every 3-4 seconds, is quite tolerable. Only a few updates (around 17% of updates in our experiment) actually caused a change in allocated utility levels, resulting in a message from the BM to the source of the flow(s) that needs to change utility. Since 100-150 data packets were transmitted end-to-end in the same 3-4 seconds, we found that the update and utility re-negotiation overhead is under 1.5% for this scenario, which is quite low. We also measured the round-trip delay of sending an update packet to the BM from the source node of flow 1 (one of the furthest active nodes from the BM), and receiving a new utility level in response. We found that this request-reply overhead was also feasible: 64 ms on average before flow 2 started and 74 ms on average after flow 2 added extra traffic to the network.

We compared the overhead of our scheme with that of distributed, localized, periodic MAC updates [18] and distributed, localized, triggered MAC updates (using Explicit Congestion Notification i.e. ECN) [7]. For the 25-node scenario, our scheme incurs total network overhead of 2.35 update packets per second. In localized, periodic MAC updating, even with 1 update per second periodic updates, the overhead when both flows are active is 27 packets per second. This is because there are 9 active nodes and each node must report its MAC information 3 hops away in MPARC [18]. Moreover, due to reasons explained earlier, even 3 hops is insufficient, since there are nodes 4 hops away that are within the interference range of an updating node. So despite the higher overhead, the update mechanism is inaccurate in MPARC. We

found that for MAC-updates using ECN followed by ”regulate” messages [7], the total network overhead was 0.85 updates per second. In the forward direction (reporting), the update is passed in-band using the data packets themselves. Only for rate regulation in response to the reporting are extra status messages explicitly generated. Furthermore, the ECN-based schemes take advantage of the best of both worlds: they are localized rather than centralized, and they are also trigger-based rather than periodic. In other words, only if there is something to report does the MAC-layer generate overhead messages, not periodically. And the message is sent only to nodes concerned with the end-to-end flow, not to a central arbiter. However, the disadvantage with SWAN [7] is that there is no co-ordination between flows in resource reservation, unlike with the MPARC scheme in [18] and our scheme.

For large-scale networks, distributed updates make more sense, since in the centralized case, individual updates may traverse tens of nodes before reaching the central arbiter and thus increase the total update packet overhead per second on the network. We placed the central channel arbiter (BM) 10 hops away from every active reporting node in our scenario (same scenario, larger network), and found the total network overhead for our scenario to be 8.25 updates per second on average. This is still slightly less than in the case of periodic, distributed updates. However, if we place the BM 100 hops away from every active reporting node, the total network overhead is 80 updates per second. Thus, we conclude that our centralized scheme scales to a few tens or hundreds of nodes but not beyond that. If the BM is placed more than 10 hops away from the reporting nodes, then even the periodic, localized MPARC scheme of [18] performs better in terms of overhead.

The network load in our scenario is actually quite high, since the flows are forced to operate below their highest utility level. We experimented with higher loads, which drove all admitted flows to their minimum utility level, and also with different utility functions. The update overhead relative to the data traffic was under 5% in all cases, which is acceptable. Based on the results with different loads, we estimated the overhead for the worst case theoretical scenario, wherein all 40 possible single hops in the 25-node grid are active in *both* directions, to be around 12%. Furthermore, the request-reply delay was also controlled because queues at all points in the network were very short. The overhead can be further reduced by using a milder history decay function in the channel quality averaging, but this affects the fidelity of the estimate. The bandwidth request and flow teardown signaling obviously take place only once for each flow run and thus constitute negligible overhead. Since both the signaling and update overhead is negligible, our design

decision to not reserve resources for control packets is validated.

Chapter 9

Related Work

We structure the related work in bandwidth management into three major categories: (a) fair scheduling in wireless networks, (b) cross-layer bandwidth management architectures, (c) pricing in wireless networks, and (d) transport layer and call admission schemes. We first summarize related work in distributed wireless fair scheduling because we believe this was a first step towards incorporating QoS in a dynamic wireless network environment. We then list and compare a number of cross-layer architectures for bandwidth management in various wireless network topologies. Cross-layer design has emerged as an effective strategy for wireless bandwidth management because the tasks involved - channel quality estimation, rate adaptation, etc. - are best performed at different layers of the OSI protocol stack. These tasks must then co-operate towards optimally using the wireless channel. This co-operation usually takes the form of exchange of key state information across these tasks. Finally, there has also been considerable research in using price as a signal in priority-based resource sharing in wireless networks. We take a look at some price-based resource-allocation models.

9.1 Fair Scheduling in Wireless Networks

Distributed weighted fair scheduling schemes in single-hop and multi-hop IEEE 802.11 wireless networks [1, 37, 38, 39, 40, 41, 42] were a first step towards providing QoS to applications in wireless networks. The main idea behind these schemes was a tweaking of the IEEE 802.11 protocol's backoff mechanism so that flows are only granted access to the channel in accordance with a pre-determined weight. Flows with a higher weight have intrinsically lower backoffs so that they get more frequent access to the wireless channel. For multi-hop fair scheduling, spatial reuse of the network must also be taken into account.

The question as to how to decide the flow weights at the scheduler and how to adapt them to variations in the traffic and channel conditions remains. This is where our bandwidth management scheme is important. As mentioned before, our bandwidth management scheme is required to assist the fair scheduling scheme when it is available. At the same time, as shown in our experiments, our scheme also works well without any underlying fair scheduling schemes, by employing rate-control at the higher layers of the OSI protocol stack. This is a very important feature because today's IEEE 802.11 network interface card only implements the standard DCF MAC protocol without any fair scheduling extensions. Therefore, our bandwidth management scheme, of which we already have a working prototype, is highly deployable in today's single-hop wireless networks.

A related MAC-layer wireless channel allocation scheme is the effort-limited fair scheduling by Eckhardt and Steenkiste [43]. It adjusts the "air time" of a flow to meet its minimum bandwidth requirement in response to channel error rates, only up to a certain factor (called the "power factor"), to avoid starving other best-effort flows. The usage of air time to measure the bandwidth requirement of a flow is similar to the "channel time" in our scheme. However, it is unclear how the power factor can be chosen for different flows because this will give preferential treatment to certain users. In our scheme, the limit upto which a flow can adapt is defined by its minimum throughput requirement. When a flow's minimum requirement cannot be satisfied, the flow is simply rejected. We give users the incentive to set a minimum channel time requirement as small as possible to reduce the possibility of being denied access to the channel. The ELF scheduling scheme is best implemented along with the IEEE 802.11 protocol's PCF mode. While it reacts only to physical errors that cause packets to be lost during transmission on the channel, our scheme detects and reacts both to channel errors as well as medium contention.

9.2 Cross-layer Bandwidth Management Architectures

There exist several cross-layer bandwidth management architectures for various IEEE 802.11 wireless network topologies [6, 5, 3, 7, 4]. These architectures, for the most part, even employ similar sub-tasks towards performing bandwidth management: available bandwidth monitoring, maintaining soft-state reservation, rate-control, adaptation to changes in the network environment, and fair bandwidth sharing. In this section, we compare and contrast the various cross-layer architectures with respect to how they perform their critical tasks, and also mention how they differ from our cross-layer scheme (BM). Figure 9.1 puts the comparison

in a nutshell.

The second row of the table pertains to the adaptivity of the flow. Most of the architectures assume highly flexible flows that are adaptive over a continuous range of bandwidths. INSIGNIA is an exception: it assumes a flow can have only three bandwidth levels of operation. Mobile flows (flows in which at least one end-point is mobile) in TIMELY are pinned to their minimum bandwidth request.

The third row of the table pertains to the fairness in bandwidth allocation. Most schemes provide some notion of fairness or price-weighted allocation. However, INSIGNIA and SWAN are exceptions. Admission control is first-come first-serve (FCFS) in these schemes: flows are admitted while resources last. Which flows must adapt in response to available bandwidth variations, and how much they should adapt, is not deterministic.

The fourth row indicates how the decision to allocate bandwidth is made. In the case of TIMELY, while the bandwidth allocation is decided by the weighted max-min servers, the reservation state is communicated via an RSVP-like signaling algorithm and maintained distributedly. The weighted max-min servers ensure co-ordination that is lacking in fully distributed allocation decisions. In fully distributed schemes, per-flow state is maintained at multiple nodes in the network. SWAN is an exception; it maintains aggregate flow state at each node in the network. In centralized schemes, per-flow state is maintained centrally. In both types of schemes, a signaling protocol is needed to communicate state information.

As evident from row 6 of the table, even some of the reservation-based schemes such as TIMELY and INSIGNIA provide differentiated services based on packet priorities. This is because reservations in a wireless environment are not hard, but are subject to dynamic variations in topology and channel characteristics. Packets are marked as low or high priority and when available bandwidth becomes unexpectedly scarce and rate control needs to be performed, preferably the lower priority packets are dropped.

9.3 Pricing in Wireless Networks

In this section, we survey the literature in the field of price-based bandwidth allocation in different types of wireless networks. We point out the differences from our work and highlight our contributions.

An early work on pricing and bandwidth adaptation in wireless networks was the TIMELY project [3]. Users benefitting from adaptation were charged and those suffering from adaptation were compensated. However, how the exact charges and credits were calculated, was not specified. In [24], the authors discuss

price-based resource allocation on the downlink of a time-slotted or CDMA-based wireless LAN. They assume that users do not know each others' utility curves. In [44], the authors divide the network bandwidth into stable (low bandwidth) and instantaneous (unstable, high bandwidth) classes, and broadcast a price-service menu for these classes periodically. The users are given an incentive to truthfully declare their required bandwidth and service class. As far as we know, the concept of *channel time proportion* is unique to our scheme. Our scheme also directly auctions channel time so that users who value it more obtain more of it.

There has also been research in the area of price-based resource allocation for wireless ad-hoc networks [12, 45]. In [12], the authors argue that the shared resource is not a link, as in the case of a wireline network, but a wireless neighborhood clique. On the basis of this, they adapt the concepts of Kelly et al's seminal paper [11] on price-proportional fair resource allocation for concave, elastic traffic in wireline networks to mobile ad hoc networks. In [29], the authors use a multi-unit Vickrey auction [23] to have flows bid for forwarding service at intermediate mobile hosts in a mobile ad hoc network environment. This work also uses the concept of CTP used in our work. A game-theoretic model for pricing in AP-based wireless LANs is described in [46]. The price is determined based on the modeling of the interaction between the AP and the mobile host as a 2-player game. The scheme does not currently take into account dynamic arrivals and capacity constraints. Furthermore, the authors only consider web-browsing and file-transfer applications, at present.

In [47], the authors present a complete bandwidth allocation and adaptive re-allocation model for different network topologies using 3G wireless technology. They use a stepwise utility function, quantized from a continuous, concave function [48]. They present an aggregate utility maximization algorithm for a single-hop 3G network, as well as a price-based approach for a multi-hop ad hoc 3G network. Their system does not involve cross-layering, however. Since there is no co-channel interference at the medium in 3G systems, they do not need to take into account channel effects in their resource allocation model. They perform resource allocation at the application level, taking user requirements alone into account.

9.4 Transport-Layer and Call Admission Schemes

In addition to the fair-scheduling schemes at the MAC-layer, several new transport [49, 50, 51, 52, 53, 54, 55] and hybrid link-transport [56, 57, 58] schemes have also been proposed to improve application performance,

fairness, and network efficiency in single-hop wireless networks. All these schemes rely on identifying and reacting to losses on the wireless link. On the other hand, our cross-layer scheme provides throughput guarantees and policy-based channel sharing, while still using existing transport and MAC technologies. Our adaptation scheme reacts both to wireless channel losses as well as location-dependent contention.

There have also been schemes proposed for one-time admission control in wireless networks. In [59], the authors propose a MAC-layer admission control scheme for a peer-to-peer, single-hop, wireless network model. Their scheme requires the use of special probe packets to obtain the service curve, which is an estimate of network load. Using the service curve, one-time admission control is performed. In contrast, our scheme estimates network load using the data packets of the connection itself. In [17], the authors perform one-time admission control for MANETs, based on the busy-idle channel monitoring described in Chapter 3. We perform dynamic bandwidth re-negotiation over the course of the connection, in addition to admission control at flow startup. Due to the channel and traffic variations in the network, dynamic adaptation and management of bandwidth is required during the operation of the flows, not just admission control at the time of flow establishment.

In [18, 60, 61], admission control is done for differentiated service at the MAC layer based on some priority for various topologies of 802.11 networks. The priority is determined by either the delay requirements of flows [61] or based on real-time or best-effort classes of traffic [18, 60]. In all cases, the 802.11 MAC protocol is modified in order to support service differentiation: the contention window is adapted in keeping with the priority of the flow. In MPARC [18], as mentioned in Chapter 3, a novel, predictive channel capacity estimation scheme is used which takes into account congestion load in an ad hoc network. Priority-based rate-control in the ad hoc network is then used such that real-time flows' bandwidth requirement is always satisfied, and admitted flows are never pre-empted. Best effort traffic scales itself to whatever bandwidth is available. All flows are managed, and rates for both real-time and best-effort flows are controlled via a leaky bucket mechanism. The channel capacity estimation scheme of MPARC is compared and contrasted with our scheme in Chapter 3. In terms of bandwidth allocation, this scheme does not assume scalable, non-adaptive real-time flows; the authors assume that all real-time flows can be classified into different priorities, but have one fixed bandwidth requirement.

	Criterion	TIMELY	Utility-fair	INSIGNIA	dRSVP	SWAN	BM
1	Network scenario	Wide-area wireline + last-hop wireless	Access-point based wireless LAN	MANET	MANET (or) wide-area wireline + last-hop wireless	MANET	Single-hop ad hoc (or) access-point based wireless LAN
2	Granularity of BW allocation	Mobile Flows: only min. requirement Static Flows: continuous min-max BW range	Continuous range of BW from min. utility to max. utility	3 discrete QoS levels: best-effort, min., and max. BW requirement	Continuous range from min. to max. BW requirement	Continuous range between 0 and max. BW requirement	Continuous range from min. to max. CTP requirement
3	Fairness criterion	Weighted max-min fairness with min. guarantees	Utility-fairness	No fairness	Request proportional fair with min. guarantees	No fairness	Max-min fairness with min. CTP guarantees
4	Allocation decision	Semi-centralized (Weighted max-min servers)	Centralized (Access-point)	Distributed	Distributed	Distributed	Centralized (Bandwidth manager)
5	Modules participating in cross-layer interaction	App. Layer: Reservation managers, weighted max-min servers T'port Layer: HPF Network Layer: RSVP-based advance reservation Link Layer: link monitoring and fair scheduling	App. Layer: Adaptive application Link Layer: adaptation handler, adaptation controller and MAC-layer signaling	App. Layer: Adaptive application Network Layer: In-band signaling, packet marking Link Layer: Fair packet scheduling, channel capacity utilization monitoring	App. Layer: Adaptive application Network Layer: RSVP-based reservation Link Layer: per-interface available bandwidth estimation	App. Layer: Admission controller Network Layer: probing, ECN, marking Link Layer: Shaper, 802.11 transmission delay monitoring, per-interface available BW estimation	App. Layer: Adaptive application Link Layer: per-neighbor available bandwidth estimation at MAC layer
6	Differentiation through packet classification	Yes	No	Yes	No	Yes	No
7	Available bandwidth partitioning	3 categories: per-flow reserved BW, common reserved BW, aggregate BW for best-effort traffic	3 categories: sustained rate BW active adaptation BW, passive adaptation BW	No partitioning	No partitioning	2 categories: real-time traffic upto admission control BW, best-effort from admission control BW to threshold BW	No partitioning of channel time

Figure 9.1: A study of various cross-layer bandwidth-management architectures

Chapter 10

Future Work

Our bandwidth management architecture is a very flexible one. Its major components, the CCE, RA, and the BM, can be easily replaced by alternate mechanisms to perform the same functions, without much change to the overall architecture. We have presented our channel capacity estimation algorithm, our approach to rate-control and our bandwidth management policies in this work. However, these can be replaced by more suitable schemes, where needed. The interfaces are clean and simple. This flexibility in our architecture thus easily accommodates incorporation of improvements and enhancements to the basic bandwidth management scheme.

In this section, we present areas of current and possible future research based on our bandwidth management architecture presented in Chapter 4. One of the enhancements mentioned in this chapter pertain to energy efficiency, which is important in a mobile network because it might comprise small, untethered computers that are low on battery power. Another enhancement pertains to improved performance via contention prediction in a sensor-network scenario. The contention prediction is based on knowledge of the individual mobile hosts' locations and their traffic profiles.

10.1 Energy-Efficient Bandwidth Management

In our bandwidth management scheme, we allocate each flow a share of the channel called its allocated channel time proportion (CTP). A flow can be actively transmitting on a channel for only a fraction of unit time, represented by its allocated CTP. The basic idea behind energy-efficient bandwidth management for multimedia traffic is to keep each network interface switched on only for a fraction of a time interval τ that is equal to its allocated CTP, and to power it down at other times, in order to conserve energy. In the basic

bandwidth management scheme, the allocated CTP is used via multiple packet transmissions spaced out over unit time. On the other hand, in the energy-efficient scheme, we give the flow continuous, unhindered access to the channel for a fraction of unit time equal to its allocated CTP, and power down the network interface when this interval of time is complete.

The interval τ must be sufficiently large, because for smaller intervals, the overhead of repeatedly powering up and powering down the network card will offset any energy gained through keeping the card powered down. At the same time, the size of the multimedia application's sender and receiver buffers, its playback rate, and its delay-jitter requirements must be also be taken into account in deciding the time that the card can be powered down.

Thus in the energy-efficient bandwidth management scheme, the BM not only returns the allocated CTP to each flow but also a *transmission schedule* for the flow's transmissions so that, when not scheduled, the flow can power down its network card. It must take as input the multimedia flow's sender and receiver buffer sizes, along with the CTP requirements. The BM transmits a beacon at the end of every interval τ. For a brief interval after the beacon is heard, flows can request or re-negotiate CTP. All flows must thus wait for this beacon before making any requests. The BM then runs a real-time scheduling and admission control algorithm to compute the allocations and the schedule. At the end of the brief interval, the schedule is transmitted through a broadcast, after which the next interval τ begins. In interval τ, each admitted flow gets a share of this time τ on a mostly contention-free channel, since the other flows have powered down their network interfaces when this flow is transmitting. When a flow is to be scheduled, its RA wakes its network interface up, releases a few packets to it, and then powers it down when its time is up. The function of the CCE is deprecated because the channel is largely contention-free and the only fading errors need to be estimated now. Figure 10.1 illustrates the working of the energy-aware bandwidth management protocol.

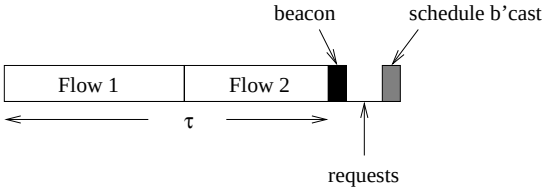


Figure 10.1: Energy-efficient bandwidth management

The time-division multiplexing of the wireless channel, over the IEEE 802.11 protocol, is similar to that achieved through scheduled polling in the IEEE 802.11e draft proposal [62, 63] or even the IEEE 802.11

PCF mode [2]. However, we avoid polling overhead by having RAs keep the schedule and updating it every time interval τ. The IEEE 802.11e draft also does not mention what type of utility curve (and hence bandwidth request model) the individual flows use and what type of bandwidth allocation policy is used at the BM. The time-division multiplexing of the wireless channel in these cases is to eliminate contention. In our case, we perform time-division multiplexing so that we can maximize host lifetime, while obtaining reduced contention as a by-product. We achieve time-division multiplexing of the wireless channel using off-the-shelf IEEE 802.11 products that implement only the DCF mode.

In general, the IEEE 802.11e draft attempts to provide QoS over a wireless channel by two mechanisms: (a) priority class based queuing and scheduling at the MAC layer, and (b) contention-free periods. Our scheme provides throughput guarantees, and limits delay, without changing the base IEEE 802.11 MAC layer's functionality, and without introducing time-division multiplexing of the channel (i.e. contention-free access). Our scheme can work in conjunction with 802.11e's EDCF. Since the latter provides improved MAC layer queuing and scheduling, the traffic shaper from our scheme is not required. The IEEE 802.11e EDCF basically provides QoS support via priority-aware rate-control, but does not have any provisions for bandwidth management. Hence, our scheme is still required. The IEEE 802.11e Hybrid Co-ordination Function (HCF) does have a very general and flexible bandwidth management mechanism built in, but needs to invoke periods of contention-free access to implement it.

10.2 Contention Prediction

One of the problems in our bandwidth management scheme is that, when a new flow is admitted, the effect of the contention it produces on the QoS of existing flows cannot be predicted in advance. In a multi-cell environment, the location of the mobile host also influences the contention its transmissions produce on flows involving neighboring mobile hosts. If we can use location sensors to detect a mobile host's location and if we can have a profile of its traffic beforehand, then we can predict in advance, or know from afar, what sort of contention the mobile host is producing in its neighborhood. We can also leverage our previous work [34] in location prediction to *predict* contention and improve QoS.

Chapter 11

Conclusion

Computer networks need to satisfy several specific requirements in order to be able to support real-time multimedia flows. Multimedia applications can have stringent bounds on minimum end-to-end delay, maximum delay-jitter, minimum throughput, etc., to enable them to function at, at least their minimally acceptable quality level. In current IEEE 802.11 wireless networks, there are no currently no provisions to guarantee flows these quality-of-service (QoS) measures. Not only is the overall channel bandwidth lower in wireless networks as compared to wireline networks, but available bandwidth to a particular flow also varies with time and location due to variation in contending traffic and in the fading effects of the wireless channel. In such a dynamic environment, providing multimedia flows QoS guarantees becomes a challenging problem. Furthermore, in order to provide as many flows their minimum acceptable QoS, the flows have to *co-operate* with each other so that they do not greedily attempt to maximize their quality while damaging that of others. Achieving such co-operation in a distributed environment is also a challenging problem. The fact that IEEE 802.11 networks are widely deployed today means that a QoS solution must be fully compatible with current off-the-shelf hardware and software products.

In this work, we present an architecture by which all flows co-operatively control their transmission rates so that all flows in the network are guaranteed, at the very least, a throughput within a pre-defined bound of their minimum requirement, for almost their entire duration of operation. The individual flows adapt to the fluctuations they perceive in channel contention and in fading errors on the channel. While providing a guaranteed throughput to real-time multimedia flows, our architecture also ensures that best-effort flows are not starved. Our solution is easily deployable with current off-the-shelf IEEE 802.11 products and works with common IEEE 802.11 network topologies. Our architecture comprises various tasks which span the OSI protocol stack at the mobile host and work in synchrony with each other towards the goal.

We perform channel capacity estimation, translation of bandwidth requirements to *channel time proportion* (CTP) requirements at the application layer, and rate-control. We also use an arbiter called the Bandwidth Manager (BM) that distributes unit channel time amongst contending flows and detects unmanaged flows that violate their allocated share.

At the central BM, we implement channel time allocation policies which ensure first and foremost that all admitted flows' minimum CTP requirements are satisfied. We modify the well-known *max-min fair* paradigm to incorporate this criterion, and also propose a novel price-based channel time allocation policy that guarantees minimum CTP to all admitted flows. The price-based approach is especially useful when users with different purposes value the channel bandwidth differently. It is also useful in deciding which flow to shut-off mid-operation in case channel conditions deteriorate so that all previously admitted flows can no longer be satisfied. Our price-based approach is novel because it *mimics* centrally a distributed, proportional auction of channel time amongst the users who each value it differently. It computes, with quadratic time-complexity, a price and channel allocation that is identical to the one the auction would reach at equilibrium. We contrast bandwidth-fair allocation with CTP-fair allocation, and also explore a utility-maximization arbitration algorithm. We extend our basic centralized arbitration architecture to more complex network topologies. We find that a centralized scheme scales well to multi-cell wireless LANs and static multi-hop wireless sensor networks comprising several tens or hundreds of nodes.

We thus accomplish our goal of providing some throughput guarantee to flows that require it, using off-the-shelf IEEE 802.11 products. In doing so, the major contributions we make through our scheme are: (a) bringing flows with different channel qualities on the same level playing field at the time of bandwidth allocation, through the concept of CTP, (b) co-operative and adaptive rate-control for fair sharing of channel time, at higher layers of the OSI protocol stack, and (c) designing fair and price-based channel allocation algorithms that take into account flows' minimum throughput requirements.

11.1 List of Publications

The following is a list of publications of the author, not necessarily pertaining to the topic of this work.

Book Chapter

- K. Nahrstedt, S. Shah, and K. Chen. ”Cross-Layer Techniques for Bandwidth Management in IEEE 802.11-based Wireless Networks,” book chapter, ”Resource Management in Wireless Networking,” ed. M. Cardei, I. Cardei, and D. Du, Kluwer Academic Publishers, February 2004 (to appear).

Journal

- S. Shah and K. Nahrstedt, WPrice: Price-based Channel Time Allocation in Wireless LANs, submitted to IEEE Journal on Selected Areas in Communication (JSAC), Issue on Price-based Access Control and Economics for Communication Networks, 2006.

- S. Shah, K. Chen, and K. Nahrstedt, Dynamic Bandwidth Management for Single-hop Ad Hoc Wireless Networks, ACM/Kluwer Mobile Networks and Applications (MONET) Journal, Special Issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks, 2005.

- K. Chen, Y. Xue, S. Shah, and K. Nahrstedt, Understand Bandwidth-Delay Product in Mobile Ad Hoc Networks, Elsevier Computer Communications (ComCom) Journal, Special Issue on Protocol Engineering for Wired and Wireless Networks, 2004.

- K. Chen, S. Shah, and K. Nahrstedt, Cross-Layer Design for Data Accessibility in Mobile Ad hoc Networks, Journal of Wireless Personal Communications, Special Issue on Multimedia Network Protocols and Enabling Radio Technologies, Kluwer Academic Publishers, vol. 21, pp. 49-75, 2002.

Conference and Workshop

- S. Shah and K. Nahrstedt, Guaranteeing Throughput for Real-time Flows in Multi-hop IEEE 802.11 Wireless Networks, submitted to Proc. of IEEE Military Communication Conference (MilCom 2005), Atlantic City, NJ, October, 2005.

- S. Shah and K. Nahrstedt, Channel-Aware Throughput Fairness in Multi-Cell Wireless LANs, in Proc. of IEEE Vehicular Technologies Conference (VTC-Fall 2004), Los Angeles, CA, September, 2004.

- S. Shah and K. Nahrstedt, Price-based Channel Time Allocation in Wireless LANs, in Proc. of 2nd International Workshop on Mobile Distributed Computing (MDC 2004) in conjuction with IEEE

ICDCS-2004, Tokyo, Japan, March, 2004.

- S. Shah, K. Chen, and K. Nahrstedt, Available Bandwidth Estimation in IEEE 802.11-based Wireless Networks, in Proc. of 1st ISMA/CAIDA Workshop on Bandwidth Estimation (BEst 2003), San Diego, CA, December, 2003.

- S. Shah, K. Chen, and K. Nahrstedt, Dynamic Bandwidth Management for Single-hop Ad Hoc Wireless Networks, in Proc. of IEEE International Conference on Pervasive Computing and Communications (PerCom 2003), Dallas-Fort Worth, Texas, March, 2003.

- S. Shah and K. Nahrstedt, Predictive Location-Based QoS Routing in Mobile Ad Hoc Networks, in Proc. of IEEE International Conference on Communications (ICC 2002), New York, NY, April 28th - May 2nd, 2002.

- S. Shah, K. Chen, and K. Nahrstedt, Cross Layer Design for Data Accessibility in Mobile Ad Hoc Networks, in Proc. of 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, Florida, July, 2001.

References

[1] N. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless lan," in *ACM MobiCom*, Boston, MA, August 2000.

[2] (1999) Ieee 802.11, 1999 edition (iso/iec 8802-11: 1999) ieee standards for information technology - telecommunications and information exchange between systems - local and metropolitan area network - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. [Online]. Available: http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[3] V. Bharghavan, K. Lee, S. Lu, S. Ha, J. Li, and D. Dwyer, "The timely adaptive resource management architecture," *IEEE Personal Communication Magazine*, vol. 5, no. 4, August 1999.

[4] G. Bianchi, A. Campbell, and R. Liao, "On utility-fair adaptive services in wireless networks," in *IEEE/IFIP IWQoS*, Napa, CA, May 1998.

[5] S. Lee, G. Ahn, X. Zhang, and A. Campbell, "An ip-based quality of service framework for mobile ad hoc networks," *Journal of Parallel and Distributed Computing (Academic Press), Special Issue on Wireless and Mobile Computing and Communications*, vol. 60, no. 4, April 2000.

[6] M. Mirhakkak, N. Schult, and D. Thomson, "Dynamic bandwidth management and adaptive applications for a variable bandwidth wireless environment," *IEEE JSAC*, vol. 19, no. 10, October 2001.

[7] G. Ahn, A. Campbell, A. Veres, and L. Sun, "Swan: Service differentiation in stateless wireless ad hoc networks," in *IEEE InfoCom*, New York, NY, June 2002.

[8] (2004) The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[9] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R. Katz, "An architecture for a secure service discovery service," in *ACM MobiCom*, Seattle, WA, August 1999.

[10] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service location protocol, version 2," in *RFC 2608*, June 1999.

[11] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of Operational Research Society*, vol. 49, 1998.

[12] Y. Xue, B. Li, and K. Nahrstedt, "Price-based resource allocation in wireless ad hoc networks," in *IEEE IWQoS*, Monterey, CA, June 2003.

[13] S. Shah, K. Chen, and K. Nahrstedt, "Dynamic bandwidth management in single-hope ad hoc wireless networks," *ACM/Kluwer MONET*, vol. 10, no. 1-2, February 2005.

[14] F. Cali, M. Conti, and E. Gregori, "Dynamic tuning of ieee 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Trans. on Networking*, vol. 8, no. 6, pp. 785–799, October 2000.

[15] M. Kazantzidis, M. Gerla, and S. Lee, "Permissible throughput network feedback for adaptive multimedia in aodv manets," in *IEEE ICC*, Helsinki, Finland, June 2001.

[16] K. Chen and K. Nahrstedt, "Exact: An explicit rate-based flow control framework in manet," Dept. of Computer Science, UIUC, Tech. Report UIUCDCS-R-2002-2286/UILU-ENG-2001-1750, July 2002.

[17] Y. Yang and R. Kravets, "Contention-aware admission control for ad hoc networks," Dept. of Computer Science, UIUC, Tech. Report UIUCDCS-R-2003-2337, April 2003.

[18] ——, "Throughput guarantees for multi-priority traffic in ad hoc networks," in *IEEE MASS*, Fort Lauderdale, FL, October 2004.

[19] A. Kuznetsov. Linux traffic control (tc). [Online]. Available: http://www.sparre.dk/pub/linux/tc

[20] M. Grossglauser, S. Keshav, and D. Tse, "Rcbr: A simple and efficient service for multiple time-scale traffic," in *ACM SigComm*, Cambridge, MA, August 1995.

[21] F. Fitzek and M. Reisslen, "Mpeg-4 and h.263 video traces for network performance evaluation," *http://www-tkn.ee.tu-berlin.de/research/trace/stat.html, IEEE Network Magazine*, vol. 15, no. 6, pp. 40–54, November 2001.

[22] D. Bertsekas and R. Gallager, *Data Networks (2nd Ed.), Chapter 6*. Prentice-Hall, 1992.

[23] P. Klemperer, “Auction theory: A guide to the literature,” *Journal of Economic Surveys*, 1999.

[24] P. Marbach and R. Berry, “Downlink resource allocation and pricing for wireless networks,” in *IEEE InfoCom*, New York, NY, June 2002.

[25] R. McAfee and J. McMillan, “Bidding rings,” *American Economic Review*, vol. 82, 1992.

[26] K. Hendricks and R. Porter, “Collusion in auctions,” *Annales D’Economie et de Statistique*, vol. 15/16, 1989.

[27] J. Bulow and P. Klemperer, “Prices and the winner’s curse,” *RAND Journal of Economics*, 2002.

[28] O. Angin, A. Campbell, M. Kounavis, and R. Liao, “The mobiware toolkit: Programmable support for adaptive mobile networking,” *IEEE Personal Communications, Special Issue on Adapting to Network and Client Variability*, vol. 5, no. 4, pp. 32–44, August 1998.

[29] K. Chen and K. Nahrstedt, “ipass: An incentive-compatible auction scheme to enable packet forwarding service in manet,” in *IEEE ICDCS*, Tokyo, Japan, March 2004.

[30] R. Gupta and J. Walrand, “Approximating maximal cliques in ad hoc networks,” in *PIMRC*, Barcelona, Spain, September 2004.

[31] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, “Impact of radio irregularity on wireless sensor networks,” in *ACM MobiSys*, Boston, MA, June 2004.

[32] M. Akbar, E. Manning, G. Shoja, and S. Khan, “Heuristic solutions for the multiple-choice multi-dimension knapsack problem,” in *Intl. Conf. on Computational Sci.*, San Francisco, CA, May 2001.

[33] D. Johnson, D. Maltz, and J. Broch, “Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks,” *Ad Hoc Networking, ed. Charles Perkins*, Addison-Wesley 2001.

[34] S. Shah and K. Nahrstedt, “Predictive location-based qos routing in mobile ad hoc networks,” in *IEEE ICC*, New York, NY, April 2002.

[35] S. Chen and K. Nahrstedt, “Distributed quality-of-service routing in ad hoc networks,” *IEEE JSAC*, vol. 17, no. 8, August 1999.

[36] S. Lee and A. Campbell, “Hmp: Hotspot mitigation protocol in ad hoc networks,” in *IEEE IWQoS*, Monterey, CA, June 2003.

[37] B. Bensaou, Y. Wang, and C. Ko, “Fair medium access in 802.11 based wireless ad hoc networks,” in *IEEE MobiHoc*, Boston, MA, August 2000.

[38] H. Luo, S. Lu, and V. Bharghavan, “A new model for packet scheduling in multihop wireless networks,” in *IEEE MobiCom*, Boston, MA, August 2000.

[39] H. Luo, P. Medvedev, J. Cheng, and S. Lu, “A self-coordinating approach to distributed fair queuing in ad hoc wireless networks,” in *IEEE InfoCom*, Anchorage, AK, April 2001.

[40] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, “Distributed multi-hop scheduling and medium access with delay and throughput constraints,” in *ACM MobiCom*, Rome, Italy, July 2001.

[41] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, “Achieving mac-layer fairness in wireless packet networks,” in *ACM MobiCom*, Boston, MA, August 2000.

[42] S. Lu, T. Nandagopal, and V. Bharghavan, “Design and analysis of an algorithm for fair service in error-prone wireless channels,” *ACM Wireless Networks*, vol. 6, no. 4, pp. 323–343, 2000.

[43] D. Eckhardt and P. Steenkiste, “Effort-limited fair (elf) scheduling for wireless networks,” in *IEEE InfoCom*, Tel Aviv, Israel, March 2000.

[44] R. Liao, R. Wouhaybi, and A. Campbell, “Incentive engineering in wireless lan based access networks,” in *IEEE ICNP*, Paris, France, November 2002.

[45] Y. Qiu and P. Marbach, “Bandwidth allocation in ad-hoc networks: A price-based approach,” in *IEEE InfoCom*, San Francisco, CA, March-April 2003.

[46] J. Musacchio and J. Walrand, “Game theoretic modeling of wi-fi pricing,” in *Allerton Conference on Communcation, Control, and Computing*, Monticello, IL, October 2003.

[47] C. Curescu, “Utility-based optimisation of resource allocation for wireless networks,” in *Doctoral thesis, Dept. of Computer and Information Science, Linkopings Universitet*, Linkoping, Sweden, September 2005.

[48] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek, “On quality of service optimization with discrete qos options,” in *IEEE RTAS*, Vancouver, Canada, June 1999.

[49] P. Sinha, N. Venkitaramanan, R. Sivakumar, and V. Bharghavan, “Wtcp: A reliable transport protocol for wide-area wireless networks,” in *ACM MobiCom*, Seattle, WA, August 1999.

[50] A. Bakre and B. Badrinath, “I-tcp: Indirect tcp for mobile hosts,” in *IEEE ICDCS*, May 1995.

[51] R. Caceres and L. Iftode, “Improving the performance of reliable transport and handoff performance in cellular wireless networks,” *IEEE JSAC*, vol. 13, no. 5, pp. 850–857, June 1995.

[52] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro, “Delayed duplicate acknowledgements: A tcp-unaware approach to improve performance of tcp over wireless,” Computer Science Dept., Texas A&M University, Tech. Report 99-003, February 1999.

[53] L. Magalhaes and R. Kravets, “Transport-level mechanisms for bandwidth aggregation of mobile hosts,” in *IEEE ICNP*, Riverside, CA, November 2001.

[54] ——, “Mmtp: Multimedia multiplexing transport protocol,” in *Workshop on Data Communication in Latin America and Caribbean (SIGCOMM-LA)*, 2001.

[55] Y. Yang, H. Zhang, and R. Kravets, “Channel quality-based adaptation of tcp with loss discrimination,” in *IEEE Globecom*, Taipei, Taiwan, November 2002.

[56] H. Balakrishnan, S. Seshan, and R. Katz, “Improving reliable transport and handoff performance in cellular wireless networks,” in *ACM MobiCom*, Berkeley, CA, December 1995.

[57] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, “Improving tcp/ip performance over wireless networks,” in *ACM MobiCom*, Berkeley, CA, December 1995.

[58] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, “A comparison of mechanisms for improving tcp performance over wireless links,” in *ACM SigComm*, Stanford, CA, 1996.

[59] S. Valaee and B. Li, “Distributed call admission control in wireless ad hoc networks,” in *IEEE VTC*, Vancouver, Canada, September 2002.

[60] Y. Yang and R. Kravets, “Distributed qos guarantees for realtime traffic in ad hoc networks,” in *IEEE SECON*, Santa Clara, CA, October 2004.

[61] K. Chen, Y. Xue, and K. Nahrstedt, “Distributed end-to-end proportional delay differentiation in wireless lan,” in *IEEE ICC*, Paris, France, June 2004.

[62] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor, “Ieee 802.11e wireless lan for quality of service,” in *European Wireless*, Florence, Italy, February 2002.

[63] P. Garg, R. Doshi, R. Greene, M. Baker, M. Malek, and X. Cheng, “Achieving higher throughput and qos in 802.11 wireless lans,” in *IEEE IPCCC*, Phoenix, AZ, April 2003.

Vita

Samarth Harish Shah
509 E. White St.
Apt. 4
Champaign IL 61820
Phone 217-355-4476
E-mail shshah@cs.uiuc.edu
URL http://cairo.cs.uiuc.edu/adhoc

Education

Ph.D. in Computer Science
May 2005 University of Illinois Urbana-Champaign
GPA: 3.8

Thesis: Dynamic Channel-Aware Bandwidth Management in IEEE 802.11-based Wireless Networks. Advisor: Prof. Klara Nahrstedt.

B.E. in Computer Science and Engineering
June 1998 University of Madras Chennai, India
I Class

Work experience

Teaching Assistant
August 1998 - December 1999, Dept. of CS, UIUC, Urbana, IL
Conducted lab sessions, set exams, and managed course webpage for introductory computer programming class for non-CS majors.

Summer Intern
May - August 1999, Fujitsu Network Communications Inc., Richardson, TX
Worked on an output-port scheduling algorithm for an ATM switch that was part of a hybrid transport and access product. Developed test-cases, wrote test code in C. Employed ClearCase for version management and ddd for debugging.

Research Assistant
January 2000 - current, MONET research group, Dept. of CS, UIUC, Urbana, IL
See list of publications and projects.

Summer Intern and Research Programmer
May - August 2001, HRL Laboratories LLC, Malibu, CA
Ported HRL's Multi-Layer IPsec (ML-IPsec) implementation from FreeS/WAN version 1.1 to 1.9. Implemented the functionality of an ML-IPsec enabled intermediate host in FreeS/WAN 1.9. This involved some RH 6.0 Linux kernel programming. Also worked on the design of a Group Key Mgmt. Protocol for ML-IPsec. Details at: http://www.yongguangzhang.net/papers/jsac04.html

Skills

Programming Languages: C, C++, J2SE, Perl, Visual C++, Embedded Visual C++, x86 Assembly.
Operating Systems: Windows 9x/NT/XP/CE, SunOS 5.8, RH Linux 8.0, MS-DOS, LynxOS.
Other: Sockets, Signals, Posix Threads, CORBA, ns-2 network simulator, HTML, Unix shell script, sed, awk, ANTS active network toolkit, MATLAB/Simulink, LaTeX, FreeS/WAN, h.263/MPEG-2 encoders.

Recent Projects

1. Co-operative energy-constrained network-bandwidth and CPU-usage management
Modified a Linux-based h.263 video encoder to adapt sending rate to available wireless bandwidth variations, and encoding rate to available CPU and energy resources. Wrote systems-level, multi-threaded, synchronized, code in C to interface the application to the 802.11 device driver using ioctl()s, and also to the OS scheduler. Added streaming functionality to standalone application. 2-person team, research project. Except for the OS scheduler, the rest of the project was my work.

2. Audio adaptation to available bandwidth variations in IEEE 802.11 testbed
Modified Linux-based proprietary JavaSound audio application to interface via to 802.11 wireless driver via /proc file system. The multi-threaded application negotiated with a bandwidth arbiter for wireless network bandwidth, based on available bandwidth feedback from the driver. 2-person team, research project. Setting up the testbed, and writing the application-level code, including the bandwidth arbiter, was my part. Also simulated the testbed experiments using ns-2, including in a multiple access-point topology.

3. Embedded, multi-threaded real-time control of water-seesaw
Wrote low-level C and some assembly code to read from a device register the level of a seesaw. Applied a PID control system to determine how to bring it to a level. Wrote to a device register controlling a pump a value indicating how much water to pump in order to bring the seesaw to a level. The entire system was developed on the LynxOS RTOS. 3-person team, semester project.

4. Using active networks for multi-layer multimedia multicast
Used the ANTS active network toolkit. Put some data into each "capsule" indicating whether the frame being carried in it was an I, P, or B MPEG frame, and the action to be taken for each. When congestion was encountered at an intermediate multicast router, it would intelligently forward the I frame, but drop P or B frames. 2-person team, semester project.

5. Development of Solaris-based Multimedia Surveillance Application
Built on the audio/video capture and playback example code provided with Solaris to add streaming functionality. Then used EDF scheduling to synchronize audio and video within streams and across multiple streams. Finally, added a Java-based GUI for VCR-like functions such as rewind and pause. 6-person team,

semester project. My part was the building of a suite of wrappers in C around standard Berkeley socket calls for easy setup and teardown of control- and data-connections during media streaming.

6. Exploring TCP Timers, Collecting Network Interface Statistics, and Routing Using Congestion as a Metric
Instrumented Linux kernel to modify TCP timers and gauged effect of changes using sock and netstat shell programs. Used net-snmp to collect network interface statistics, detect congestion, and used route shell command to modify routing tables accordingly. This was a suite of 2-person projects for a semester-long network programming laboratory class.

Major Publications

Papers

* S. Shah and K. Nahrstedt. "Predictive Location-Based QoS Routing in Mobile Ad Hoc Networks," Proc. of IEEE International Conference on Communications (ICC 2002), New York City, NY, April 2002.
* S. Shah, K. Chen and K. Nahrstedt. "Dynamic Bandwidth Management for Single-hop Ad hoc Wireless Networks," Proc. of IEEE Intl. Conf. on Pervasive Computing and Communication (PerCom 2003), Fort Worth, TX, March 2003. (Extended version to appear in ACM/Kluwer MONET journal special issue on Algorithmic Solutions for High-Performance Wireless, Mobile, Ad Hoc and Sensor Networks, February 2005.)
* K. Chen, Y. Xue, S. Shah and K. Nahrstedt. "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks," Elsevier journal on Computer Communications and Protocol Engineering, Winter 2003.
* S. Shah and K. Nahrstedt. "Price-based Channel Time Allocation in Wireless LANs," Proc. of 2nd Intl. Workshop on Mobile Distributed Computing (MDC 2004), affiliated to IEEE ICDCS 2004, Tokyo, Japan. (Extended version submitted to IEEE Jnl. on Selected Areas in Commns., Spl. Iss. on Price-based Access Control and Economics for Commn. Networks, 2006.)
* S. Shah, K. Chen and K. Nahrstedt. "Available Bandwidth Estimation in IEEE 802.11 Wireless Networks," Proc. of ISMA/CAIDA Bandwidth Estimation Workshop (BEst 2003), San Diego, CA, December 2003.
* S. Shah and K. Nahrstedt. "Channel-Aware Throughput Fairness in Multi-Cell Wireless LANs," Proc. of IEEE Vehicular Technologies Conference-Fall (VTC-Fall), Los Angeles, CA, September 2004.
* S. Shah, K. Nahrstedt and W. Yuan. "Co-operative Energy-Constrained Resource Management in Mobile Multimedia Environments," submitted to ACM Multimedia Conference, Singapore, November 2005.

Demos

* C. Ong, S. Shah and K. Chen. "Integrated Quality of Protection and Bandwidth Management Demo for MPEG Traffic in Gaia Smart Space Environment," NSF site visit, Dept. of Comp. Sci., UIUC, April 2003.
* S. Shah and K. Chen. "Dynamic Bandwidth Management in Single-hop Ad Hoc Wireless Networks," ONR-MURI review, Cornell University, Ithaca, NY, August 2003.

Relevant courses

* Computer Networks and Distributed Systems - Prof. Vaduvur Bharghavan
* Communication Networks for Computers - Prof. Steve Lumetta
* Mobile Computing and Communications - Prof. Robin Kravets

* Computer Networking Laboratory - Prof. Benjamin Wah
* Advanced Topics in Computer Networking - Prof. Jennifer Hou
* Mobile Computing and Wireless Networking - Prof. Nitin Vaidya
* Embedded Systems Architecture and Software - Prof. Lui Sha
* Multimedia Systems - Prof. Klara Nahrstedt

References

Available upon request.