Mobile Networks and Applications 10, 199–217, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Dynamic Bandwidth Management in Single-Hop Ad Hoc Wireless Networks [*]

SAMARTH H. SHAH, KAI CHEN and KLARA NAHRSTEDT
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801, USA

Abstract. Distributed weighted fair scheduling schemes for Quality of Service (QoS) support in wireless local area networks have not yet become standard. Therefore, we propose an Admission Control and Dynamic Bandwidth Management scheme that provides fairness and a soft rate guarantee in the absence of distributed MAC-layer weighted fair scheduling. This scheme is especially suitable for smart-rooms where peer-to-peer multimedia transmissions need to adapt their transmission rates co-operatively. We present a mapping scheme to translate the bandwidth requirements of an application into its channel time requirements. The center piece of our scheme is a Bandwidth Manager, which allots each flow a share of the channel, depending on the flow's requirements relative to the requirements of other flows in the network. Admitted flows control their transmission rates so they only occupy the channel for the fraction of time allotted to them. Thus co-operation between flows is achieved and the channel time is fair shared. As the available channel capacity changes and the traffic characteristics of various flows change, the Bandwidth Manager dynamically re-allocates the channel access time to the individual flows. Our simulation experiments show that, at a very low cost and with high probability, every admitted flow in the network will receive at least its minimum requested share of the network bandwidth. We also present extensive testbed experiments with our scheme using a real-time audio streaming application running between Linux laptops equipped with standard IEEE 802.11 network cards.

Keywords: wireless single-hop ad hoc network, distributed weighted fair scheduling, bandwidth manager, max–min fairness

1. Introduction and motivation

In recent times, much effort has gone into solving the problem of transmitting multimedia data over wireless networks. Three mutually orthogonal factors make this problem challenging: (a) stringent QoS requirements of multimedia applications, (b) bursty nature of some multimedia traffic, and (c) unreliable and dynamic nature of the wireless network. Network-specific QoS requirements of multimedia applications include minimum throughput, maximum delay and maximum jitter.

In a wireless network, the minimum throughput requirement is more difficult to achieve than in a wireline network, because (a) this requires distributed co-operation between nodes sharing a wireless channel, and (b) the flows in the wireless network are exposed to various physical channel errors. In smart-rooms and "hot-spot" networks, wireless access-enabled nodes in a small area share limited channel bandwidth. Since the area is small, the wireless hosts *pervade* through the entire network and are all within each other's transmission range. There are a large number of hosts and hence connections. So, channeling all data through a single intermediate node, such as a base-station, is inefficient. Communication is *pervasive*, i.e., there are many source–destination pairs distributed throughout the network. The sources must not all rely on a single entity, the base-station, to relay their data to their respective destinations. They should be able to directly communicate with their destinations. If a base-station is used as an intermediary, direct one-hop transmissions are needlessly made two-hop. (The base-station must only serve as an access point to the wired Internet, not as a relay for peer-to-peer transmissions between mobile nodes within the wireless network.) Furthermore, in military and disaster rescue environments, a group of people carrying mobile handheld devices should be able to communicate with each other, with no time for planning and building a support infrastructure such as a base-station. The *single-hop ad hoc wireless network*, without a base-station, thus accurately represents the network used in smart-rooms, hot-spot networks, emergency environments, and in-home networking.

IEEE 802.11 has recently become the *de facto* Medium Access Control (MAC) standard in connecting mobile hosts in an ad hoc network environment. It relies on the Distributed Co-ordination Function (DCF) to resolve channel access contention in a distributed way. However, the IEEE 802.11 DCF does not currently have any provision to guarantee QoS, such as minimum throughput, to flows accessing the channel. Without any co-ordination, if the sum of transmission rates of all the hosts (or flows) is greater than the channel capacity, heavy channel contention will occur and thus QoS cannot be guaranteed for any flow. Much research has been done in the area of distributed weighted fair scheduling (DWFS) [3,13,18,19,24] for IEEE 802.11 networks operating in the DCF mode. In DWFS, each flow is assumed to have a weight which defines its importance relative to other flows. A scheduler combined with the MAC-layer IEEE 802.11 protocol then schedules the flows' packets on the channel such that

[*] This work was supported by the DoD Multi-disciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant NAVY CU 37515-6281, and the NSF EIA 99-72884 grant. Any opinions, findings and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

the throughput they receive is proportional to their weights. However, DWFS is not yet a standard part of the IEEE 802.11 MAC protocol.

In the absence of distributed MAC-layer weighted fair scheduling in the current IEEE 802.11 standard, we propose a scheme at the *higher layers* of the OSI protocol stack to co-ordinate individual flows' channel access in the single-hop ad hoc network scenario, in order to promote co-operation between flows and provide minimum throughput guarantee for each of them. To this end, we first determine the flows' weights based on their relative channel access requirements. The flow weights, in turn, determine the transmission rate of each flow. The flows' transmission rates are controlled at the application or middleware layers, without any MAC-layer scheduling support. Therefore, our scheme can be used over the standard IEEE 802.11 protocol and is easily deployable using today's off-the-shelf 802.11 products. In case DWFS becomes available at the MAC-layer in the future, our scheme is still required in order to provide the MAC-layer scheduler with the flow weights, but enforcing the flow weights will be left to the MAC-layer scheduler.

The exact share of network bandwidth allotted to a flow depends on its requirements relative to the requirements of other flows. Each flow maps its minimum and maximum bandwidth requirements to its minimum and maximum *channel time proportion* (CTP) requirements, respectively. We propose the use of a centralized *Bandwidth Manager* (BM), which obtains from each flow its CTP requirements, at the start of its session. It uses this information to gauge what proportion of unit channel time (CTP) each flow should be allotted. The CTP allotted by the BM to each flow (i.e., its "flow weight") lies somewhere between the flow's minimum and maximum requirements. The term *channel time proportion* is defined as the fraction of unit time for which a flow can have the channel to itself for its transmissions. Since our network model allows only one node to transmit on the channel at a time, there is a direct correspondence between the channel time a flow uses and the share of the network bandwidth it receives. The BM may also *refuse* to admit a flow, i.e., allot 0% channel time. This can happen if the flow's minimum CTP requirement is so large that the network cannot support it, without violating some other flow's minimum CTP requirement.

The problem with the admission control solution described above, however, is that it is a one-time procedure performed before the flow starts. It does not take into account the changes in the wireless network over the duration of the flow's operation. Not only can the perceived channel capacity vary over time due to varying contention [6] as flows arrive and depart, but the channel capacity as perceived by different network nodes at the *same* time can also be different. The latter phenomenon is due to *location-dependent* fading errors and *location-dependent* interference from external objects.

When a new flow arrives and demands a share of the channel, the respective CTPs allotted to already existing flows may have to be reduced in order to accommodate it. This revocation of channel time should not, however, result in these existing flows ceasing to meet their minimum CTP requirement. Similarly, when a flow ends, its CTP must be suitably redistributed among the still existing flows so they can hope to achieve a better QoS.

The BM must therefore not just perform one-time admission control and teardown, but also perform *dynamic bandwidth management*. The BM must re-negotiate with each flow its CTP as its channel characteristics change, and as the number of active flows in the network varies. The detection of change in channel characteristics, and adaptation of the flow to this change, happen continuously through the course of the session. Bandwidth re-negotiation must also occur before a flow changes its packet transmission rate, as in the case of bursty VBR traffic.

The rest of the paper is structured as follows. The next section describes the overall network topology, the architecture of the bandwidth management system and the bandwidth management protocol. Section 3 presents our experimental results. Section 4 discusses some related work in the field. Finally, section 5 concludes the paper.

2. Bandwidth management system – design and implementation

In the previous section, we motivated the need for admission control coupled with dynamic bandwidth management in a single-hop ad hoc wireless network. In this section, we describe the characteristics of the network we are concerned with, the architecture of the bandwidth management system and the communication protocol.

2.1. Network model

We design and implement our bandwidth management scheme for a wireless network consisting of heterogeneous computers and devices connected together over the IEEE 802.11 MAC layer. The network in our prototype testbed implementation consists of handheld PCs and laptop computers with their 802.11 interfaces configured in peer-to-peer ad hoc mode. We assume that each node in the network is within the transmission range of every other node. Hence, only one node can transmit at a time over the channel. Since every node is within the transmission radius of every other node, routing is single-hop.

Unlike in [5], where a base-station determines the schedule of transmission for the entire network and all communication is via the base-station, in our network, transmission is distributed and peer-to-peer. The IEEE 802.11 MAC protocol's DCF, which is the one relevant to our network model, does not have a provision for a fixed transmission schedule. A node can send when it senses that the channel is not busy. A binary exponential backoff mechanism resolves collisions that might occur as a result of nodes transmitting at random times. Moreover, any node in the network can transmit to any other node directly without using the base-station as an intermediary hop. Figure 1 illustrates our network model as compared to the base-station model. The distributed, peer-to-peer and ad hoc nature of our wireless network model makes

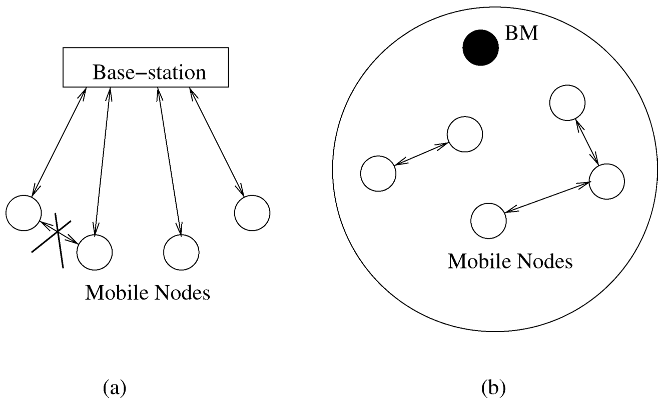


Figure 1. Comparison of network models: (a) base-station model, (b) single-hop ad hoc network model.

the bandwidth management problem significantly harder to solve than in the case of a base-station co-ordinated wireless network where the base-station has full control of the contending flows.

The wireless network has one system selected to host the *Bandwidth Manager* (BM) program. In our prototype implementation, we choose one of the more resource-rich nodes in the network, 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 [8,12] can be used to obtain the IP address and port number of the BM service. 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.

Note that the base-station network is merely a special case of the single-hop ad hoc network, but with no peer-to-peer communication between mobile nodes. (All communication, as mentioned before, is between the base-station and the mobile nodes.) Most current wireless LANs, which adopt the base-station network model, also use IEEE 802.11 DCF. Hence the contention characteristics are identical to those in a single-hop ad hoc wireless network. Our solution, which is basically designed for the single-hop ad hoc network, thus also works for the base-station network. Uplink and downlink traffic between a particular mobile node and the base-station can simply be considered as two separate single-hop flows, and their respective channel time requirements can be allotted accordingly by the BM. The BM in the base-station network can be situated at the base-station itself. In this paper, for brevity, we focus only on the single-hop ad hoc peer-to-peer network model.

We assume a network 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 band-

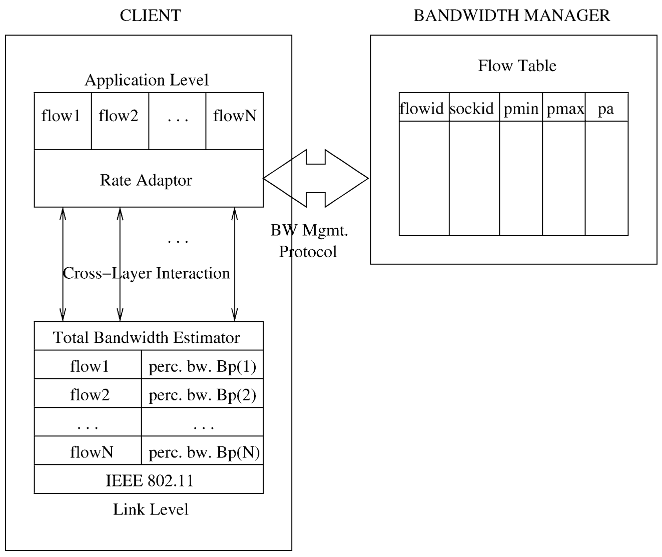


Figure 2. Bandwidth management system architecture.

width requirement $B_{\max}(f)$. The flow f also has an estimate of the total network bandwidth $B_{\mathrm{p}}(f)$. At the time of registration, it specifies its minimum and maximum CTP requirements, $p_{\min}(f)$ and $p_{\max}(f)$, to the BM. Section 2.3 discusses how $p_{\min}(f)$ and $p_{\max}(f)$ are obtained from $B_{\min}(f)$ and $B_{\max}(f)$, respectively. In response, the BM adds flow f to set F and allots it a certain channel time $p_{\mathrm{a}}(f)$, when the flow is admitted. Flow f then uses this allotted CTP $p_{\mathrm{a}}(f)$ to calculate its transmission rate. It transmits using this transmission rate until either it stops or until a new $p_{\mathrm{a}}(f)$ value is allotted to it. A new $p_{\mathrm{a}}(f)$ could be allotted to it when there is a change in the channel characteristics or in the network traffic characteristics.

We assume that the flows in the wireless network are well-behaved and co-operative, i.e., they will refrain from exceeding their allotted channel share (eating into other flows' share) and will release any channel share allotted to them when they stop. If the flows are not well-behaved and co-operative, then a policing mechanism (see section 2.7) can be used to detect the "rogue" flows and eliminate them from the system.

2.2. Bandwidth management system architecture

The architecture of the bandwidth management system consists of three major components as shown in figure 2: (a) the Rate Adaptor (RA) at the application or middleware layer, (b) the per-node Total Bandwidth Estimator (TBE) at the MAC-layer and (c) the Bandwidth Manager (BM), which is unique in the entire single-hop wireless network. Our system takes advantage of *cross-layer interaction* between the application/middleware and link layers.

Rate Adaptor (RA). In our design, we assume the absence of DWFS 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 allotted CTP for this flow from the BM. It then controls the transmission rate of each flow depending on its allotted CTP. For the sake of simplicity, in our UDP simulation experiments and testbed experiments, the RA is built into the UDP application itself, to adapt its data generation rate. Ideally, however, to avoid changing the application, we recommend that the RA be implemented separately as a module and be linked to the application at run-time. It would thus function as middleware, just below the application layer, and shape the applications' traffic. Various queue-based rate controllers are available for this purpose [16]. Our interest is in the design of the overall bandwidth management architecture, rather than the implementation of individual rate control mechanisms. For our TCP simulation experiments, we simulate queue-based rate control by having an RA per-node at the network interface queue, rather than within the application.

Note that, in case DWFS 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 communicating with the BM and determining the flow rate.

Total Bandwidth Estimator (TBE). The per-node Total Bandwidth Estimator is co-located with the IEEE 802.11 protocol at the MAC layer. It estimates the total network bandwidth $B_\mathrm{p}(f)$ for each flow f sourced at the node it resides on.[1] $B_\mathrm{p}(f)$ is what flow f *perceives* to be the total bandwidth of the network at a particular time. In other words, at a particular instant in time, $B_\mathrm{p}(f)$ is equal to the theoretical maximum capacity of the channel (1, 2, 5.5 or 11 Mbps for IEEE 802.11) minus the bandwidth lost due to channel errors, caused by fading, interference and contention experienced by flow f's packets, at that instant. The physical channel errors and contention at a particular instant in time is estimated from the errors and contention experienced in recent history. Details of the estimation method of $B_\mathrm{p}(f)$ are in section 2.4. Note that the TBE is per-*node* whereas it performs total bandwidth estimation per-*flow* sourced at the node it resides on.

The TBE *continuously* measures the total perceived bandwidth for each flow. It periodically passes this up to the RA of the flow at the higher layers. The RA of a flow f uses it in the translation of flow f's bandwidth requirements to its CTP requirements. When the total bandwidth $B_\mathrm{p}(f)$ perceived by flow f changes, the channel time requirements calculated using $B_\mathrm{p}(f)$ also change. The TBE informs the RA of the new $B_\mathrm{p}(f)$. 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 calculated with the new $B_\mathrm{p}(f)$ estimate. Since CTP allotted to flow f is directly related to its share of total network bandwidth, if a flow perceives the total network bandwidth 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 $B_\mathrm{p}(f)$, however, in order to keep re-negotiation overhead small. The RA's not reacting to small changes in $B_\mathrm{p}(f)$ 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.

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.

Bandwidth Manager (BM). The Bandwidth Manager performs admission control at the time of flow establishment and bandwidth redistribution at the time of flow teardown. Admission control involves 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 admits a flow only if it can allot *at least* its minimum CTP requirement. Otherwise, the flow is rejected. 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 our channel time allocation scheme at the BM *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 fairness algorithm can be found in section 2.5 of the paper.

2.3. 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

[1] In a single-hop peer-to-peer wireless network, we perform bandwidth management per-flow, since each flow can have a different destination. In a base-station environment, we can perform bandwidth management per-node since every node only communicates with the base-station. In the base-station scenario, each node, rather than application, specifies its bandwidth requirements to its RA, and bandwidth estimation is done only for links between mobile nodes and the base-station.

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

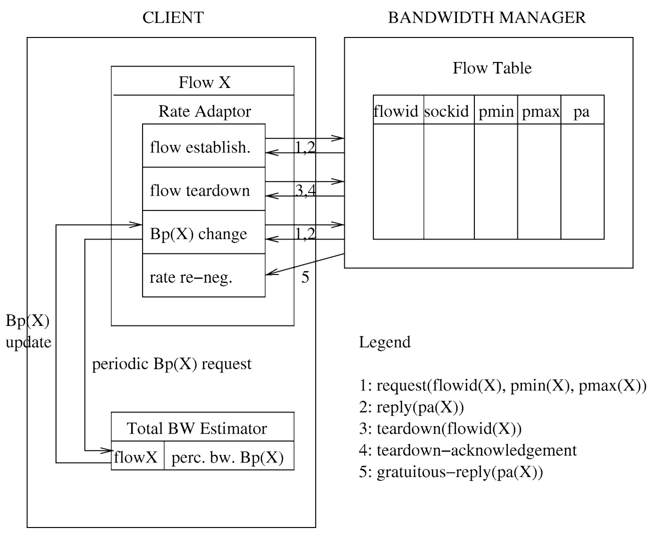


Figure 3. Bandwidth management protocol.

Table 1
Explanation of notation used in Bandwidth Management protocol.

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
$B_{\text{p}}(f)$	Total network bandwidth as 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_{rem}	$1 - \sum_{g \in F} p_{\min}(g)$: channel time remaining after $p_{\min}(g), \forall g \in F$ is met
$p_{\text{newmax}}(f)$	$p_{\max}(f) - p_{\min}(f)$: maximum channel time proportion requirement for f that is input to max–min algorithm because $p_{\min}(f)$ is already allotted
$p_{\text{mm}}(f)$	Channel time proportion allotted to flow f by max–min algorithm. This is in addition to $p_{\min}(f)$ which was already allotted before max–min algorithm began
$p_{\text{a}}(f)$	Total channel time proportion allotted to flow f, i.e., $p_{\min}(f) + p_{\text{mm}}(f)$

invoked at the time of flow establishment, flow teardown, significant change in a flow's perception of total bandwidth, or significant change in a flow's traffic pattern. Figure 3 shows the actions that occur when these events happen. Table 1 is an explanation of the notation used in the protocol description.

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. The dRSVP [21] scheme also uses maximum and minimum bandwidth requirements as the specification of utility. These values have to be each divided by the flow f's perceived total network bandwidth $B_{\text{p}}(f)$ to obtain its requested minimum and maximum CTPs, $p_{\min}(f)$ and $p_{\max}(f)$, respectively. The total network bandwidth $B_{\text{p}}(f)$ perceived by a flow f is estimated by the TBE at the local node. A best-effort flow will have $B_{\min}(f) = 0$. Figure 4 shows the shape of the utility curve of the application.

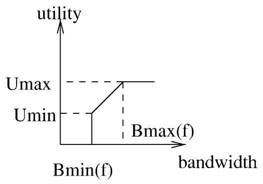


Figure 4. Utility curve of users.

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. Still, it is sufficient to do bandwidth estimation at only one of the end-points of the link. This is because both types of packets traverse the same wireless link, and hence face the same level of contention and physical errors. The TBE simply quantifies the effect of these phenomena. We perform bandwidth estimation, using the TBE, at the source. Of course, the data and acknowledgements may be of different sizes and packets of different sizes are affected differently by the same level of physical error. Hence $B_{\text{p}}(f)$ is different for different packets of the same flow. The TBE returns a single bandwidth estimate $B_{\text{p}}(f)$, for the link flow f traverses, *normalized* to a standard packet size. (See section 2.4.) It must be appropriately *scaled* for different flow packet sizes, using the reverse of the normalization procedure, at the time of flow establishment and re-negotiation. 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 $B_{\text{p}}(f)$ values can be derived for data and acknowledgement packets from the single normalized value returned by the TBE.

It must be kept in mind that the TBE of flow f measures the perceived bandwidth $B_{\text{p}}(f)$ 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 $B_{\text{p}}(f)$ 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 be added to the application packet size in the scaling operation.

The RA of a node registers a new flow with the node's TBE. Initially, the TBE has no estimate of the total network bandwidth 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 total network bandwidth, 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 total bandwidth estimate.[3] Once the flow begins, a more accurate total bandwidth estimate will be available from the TBE. The requested minimum and maximum CTPs can then be modified using this new, more ac-

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

curate 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 total bandwidth 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 total bandwidth 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 total bandwidth estimates.

Let the initial total bandwidth estimate, how ever it may be obtained, for a new flow f be $B_{\mathrm{p}}(f)$. The CTP $p_{\min}(f)$, required to satisfy the new flow f's minimum bandwidth requirement $B_{\min}(f)$, is $p_{\min}(f) = B_{\min}(f)/B_{\mathrm{p}}(f)$. $p_{\min}(f) = 0$ for best-effort flows. Similarly, the CTP $p_{\max}(f)$, required to satisfy flow f's maximum bandwidth requirement, is $p_{\max}(f) = B_{\max}(f)/B_{\mathrm{p}}(f)$. 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 checks whether, for all flows g in the set F of previously registered flows, $1 - \sum_{g \in F} p_{\min}(g) \geqslant p_{\min}(f)$. 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 2.6.

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 $p_{\mathrm{a}}(g) = p_{\min}(g)$, $\forall g \in F$. The remaining channel time, $p_{\mathrm{rem}} = 1 - \sum_{g \in F} p_{\min}(g)$, is distributed among the flows $g \in F$ in *max–min fair* fashion. Our channel time allocation policy is thus called *max–min fair with minimum guarantee*. The maximum CTP requirement for each flow $g \in F$ in the max–min fair computation is set to $p_{\mathrm{newmax}}(g) = p_{\max}(g) - p_{\min}(g)$. This is because $p_{\min}(g)$ has already been allotted to it and it only needs $p_{\mathrm{newmax}}(g)$ more to fulfill its maximum CTP requirement. Thus, knowing p_{rem} and $p_{\mathrm{newmax}}(g)$ $\forall g \in F$, the max–min algorithm can now proceed. Details of the max–min fairness algorithm can be found in section 2.5.

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 $p_{\mathrm{mm}}(g)$. Now, $0 \leqslant p_{\mathrm{mm}}(g) \leqslant p_{\mathrm{newmax}}(g)$ and $\sum_{g \in F} p_{\mathrm{mm}}(g) = p_{\mathrm{rem}}$. Then, the total CTP allotted to each flow $g \in F$ is $p_{\mathrm{a}}(g) = p_{\min}(g) + p_{\mathrm{mm}}(g)$. Note that for best-effort flows, since $p_{\min}(g) = 0$, $p_{\mathrm{a}}(g) = p_{\mathrm{mm}}(g)$. In other words, channel time is allotted to best-effort flows only after all the higher priority real-time flows are all allotted at least their minimum share.

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) $p_{\mathrm{a}}(f)$. The socket descriptor is stored in the 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 2.7.)

Finally, for every flow $g \in F$, the allotted CTP $p_{\mathrm{a}}(g)$ 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.) 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. Although we implement the `reply` message as multiple unicast messages to individual RAs for reliability, it can also be implemented for efficiency as a subnet broadcast message, containing flow-id and $p_{\mathrm{a}}(g), \forall g \in F$. A flow f is rejected using a unicast `reply` with $p_{\mathrm{a}}(f) = 0$. Other existing flows' allotted CTPs are not affected.

The RA of every flow that receives a `reply` message, gratuitous or otherwise, from the BM sets its transmission rate respectively to $p_{\mathrm{a}}(g) \cdot B_{\mathrm{p}}(g)$ bits per second (bps). 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 $B_{\mathrm{p}}(g)$ 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 $B_{\mathrm{p}}(g)$ that $p_{\mathrm{a}}(g)$ was calculated by the BM. As a new $B_{\mathrm{p}}(g)$ is returned by the TBE periodically, a new rate is also used periodically. If the $B_{\mathrm{p}}(g)$ change is large since the last period, re-negotiation must occur, as explained below.

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 $p_{\mathrm{a}}(f)$ among the other flows using the max–min fair algorithm with minimum guarantees. 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.

Change in a flow's perception of total network bandwidth. The RA of every flow periodically obtains from the TBE the flow's current perceived total bandwidth. The TBE updates the RA with the mean of the perceived total network bandwidth 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 bandwidth value $NEWB_{\mathrm{p}}(f)$ differs significantly from $B_{\mathrm{p}}(f)$, 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 bandwidth $B_{\mathrm{p}}(f)$ to the newly obtained value $NEWB_{\mathrm{p}}(f)$. Note that the RA only sets $B_{\mathrm{p}}(f)$ to $NEWB_{\mathrm{p}}(f)$ and re-negotiates with the BM using this new value 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 $\delta = 15\%$ of $NEWB_{\mathrm{p}}(f)$ from $B_{\mathrm{p}}(f)$ as significant enough to warrant re-negotiation. We also measure how other perceived bandwidth 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 value of $B_{\mathrm{p}}(f)$. 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 3.) 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 $p_{\mathrm{a}}(f) = 0$, 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 $B_{\mathrm{p}}(f)$ changes 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 strategies to pick flows to cut-off in mid-operation are discussed briefly in section 3.1.1.

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 [11] 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.

Figure 5 is an MPEG-4 trace of an hour-long, 25 frames per second, medium-quality, clip of the movie "Silence of the Lambs". The trace was taken from [10] 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 5 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.

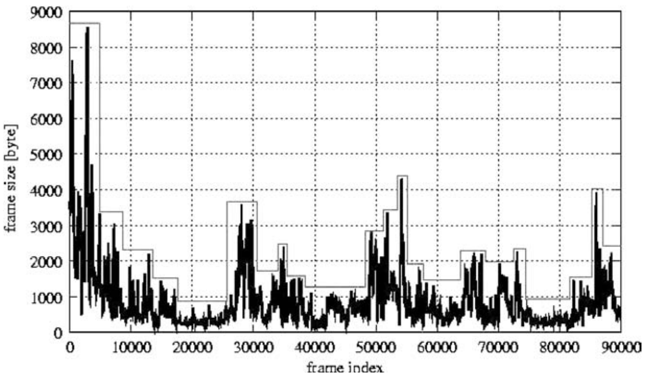


Figure 5. MPEG-4 trace of "Silence of the Lambs" clip with corresponding CBR components.

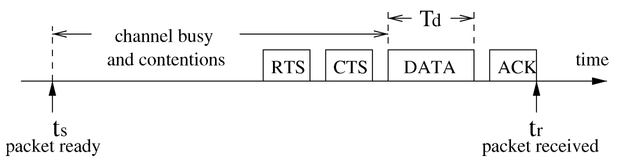


Figure 6. IEEE 802.11 unicast packet transmission sequence.

2.4. Total bandwidth estimation procedure

To determine $p_{\min}(f)$ and $p_{\max}(f)$, the RA of a flow f needs to have an estimate of the total bandwidth over the wireless link being used by the flow. To this end, we introduce a bandwidth measurement mechanism based on IEEE 802.11 DCF MAC layer, and demonstrate its robustness.

IEEE 802.11 relies on the DCF method to coordinate the transmission of packets based on CSMA/CA without any central control unit. The packet transmission sequence is illustrated in figure 6. 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.[4] 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.

Similar to [14], we measure the throughput of transmitting a packet as $TP = S/(t_r - t_s)$, where S is the size of the packet, t_s is the time-stamp that the packet is ready at the MAC layer, and t_r is the time-stamp that an ACK has been received. Note that the time interval $t_r - t_s$ includes the channel busy and contention time. We keep separate throughput estimates to different neighboring nodes because the channel conditions may be very different. We only keep an estimate for *active* links, since we do not have any packets to measure $t_r - t_s$ over inactive ones.

This MAC layer measurement mechanism captures the effect of contention on a flow's perceived channel bandwidth. If contention is high, $t_r - t_s$ will increase and the throughput TP will decrease. This mechanism also captures the effect of physical errors because if the RTS or DATA packets are affected by channel errors, they have to be re-transmitted, upto the re-transmission limit. This increases $t_r - t_s$ and correspondingly decreases the flow's perceived bandwidth. Since our MAC layer measurement of perceived bandwidth takes into account the effects of both contention and physical errors due to fading and interference on a flow, we can have the flow react suitably to these factors by monitoring the change in perceived bandwidth. It should be noted that the perceived bandwidth is measured only using *successful* MAC layer transmissions.

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

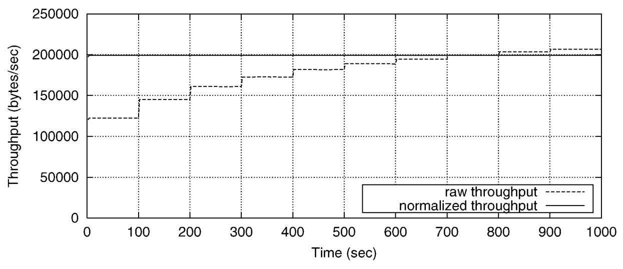


Figure 7. Raw throughput and normalized throughput at MAC layer.

It is clear that the measured throughput of a packet depends on the size of the packet. Larger packet has higher measured throughput because it 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. Before being used by a flow of a particular packet size, it must be scaled to that packet size. In figure 6, $T_d = S/BW_{ch}$ is the actual time for the channel to transmit the data packet, where BW_{ch} is the channel's bit-rate. Here we assume channel's bit-rate is a pre-defined value. The transmission times of two packets should differ only in their times to transmit the DATA packets. Therefore, we have:

$$(t_{r1} - t_{s1}) - \frac{S_1}{BW_{ch}} = (t_{r2} - t_{s2}) - \frac{S_2}{BW_{ch}} \tag{1}$$

$$= \frac{S_2}{TP_2} - \frac{S_2}{BW_{ch}}, \tag{2}$$

where S_1 is the actual data packet size, and S_2 is a pre-defined standard packet size. By equation (2), we can calculate the normalized throughput TP_2 for the standard size packet. To verify the validity of this equation, we simulated a group of mobile nodes within a single-hop ad hoc network using the *ns-2* network simulator [23]. 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 7 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.

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.

It should be noted that the bandwidth estimation mechanism in no way alters the IEEE 802.11 protocol. Our bandwidth estimation 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 applicable for arbitrarily large packet sizes or arbitrarily high bit-error rates. In such cases, the TBE 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.

2.5. *Max–min fairness*

Fairness is an important issue in designing our Bandwidth Manager. In this paper, we adopt a max–min fairness algorithm with minimum guarantee in allotting channel time to the flows. 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 [4], 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 in section 2.3, $p_a(f)$ 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 $p_{\text{newmax}}(f) = p_{\max}(f) - p_{\min}(f)$, 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 $p_{\text{mm}}(f)$. This is in addition to $p_{\min}(f)$ allotted before the max–min algorithm is even invoked.

```
Input. Channel time: $p\_rem$; set of requests: $p\_newmax[f]$
Output. Set of allocations: $p\_mm[f]$
proc Max–min($p\_rem, p\_newmax[f]$) $\equiv$
  $R := \{\}$; //set of satisfied flows
  $N := size\_of(p\_newmax[f])$;
  $p\_mm[f] := 0$;
  while ($true$) do
    $total\_satisfied = 0$;
    foreach $f \in R$ do
      $total\_satisfied += p\_mm[f]$;
    od
    $CA := (p\_rem - total\_satisfied)/(N - size\_of(R))$;
    $stop := true$;
    foreach $f \notin R$ do
      if ($p\_newmax[f] < CA$) then
        $R := R + \{f\}$;
        $p\_mm[f] := p\_newmax[f]$;
        $stop := false$;
      fi
    od
    if ($stop$) then
    foreach $f \notin R$ do
    $p\_mm[f] := CA$;
    od
    $break$;
    fi
  od
```

Figure 8. Max–min fair resource allocation algorithm.

The computation of the max–min allocation is as follows. Initially, the set of flows f, whose new maximum channel time requirement $p_{\text{newmax}}(f)$ has been satisfied, is empty: $\mathcal{R} = \emptyset$. Then, we compute the first-level allotment as $CA_0 = p_{\text{rem}}/N$, where N is the total number of flows. Now we include all flows f with $p_{\text{newmax}}(f) < CA_0$ in set $\mathcal{R}$, and allot each of them $p_{\text{mm}}(f) = p_{\text{newmax}}(f)$. Next, we compute $CA_1 = (p_{\text{rem}} - \sum_{f\in\mathcal{R}} p_{\text{newmax}}(f))/(N - \|\mathcal{R}\|)$. If for all flows $g \notin \mathcal{R}$, $p_{\text{newmax}}(g) \geqslant CA_1$, then we allot each of them $p_{\text{mm}}(g) = CA_1$ and stop. Otherwise, we include those flows g with $p_{\text{newmax}}(g) < CA_1$ in set $\mathcal{R}$, allot each of them $p_{\text{mm}}(g) = p_{\text{newmax}}(g)$, and re-compute the next level CA_2. When the algorithm terminates, the allocation $p_{\text{mm}}(f)$ for all the flows is max–min fair. The pseudo-code for the algorithm is shown in figure 8. It is clear that the computational complexity of this algorithm is $O(N^2)$. As mentioned earlier, after every flow f's $p_{\text{mm}}(f)$ has been determined using the max–min algorithm, the BM sets $p_a(f) = p_{\min}(f) + p_{\text{mm}}(f)$ and returns this value to flow f's RA.

2.6. *Alternate channel time allocation strategies*

Although we use the max–min fairness with minimum guarantee policy for bandwidth allocation in our implementation, a different fairness policy or even a biased, priority-based scheme could also be used.

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 $\sum_{g\in F} p_{\text{min}}(g) \rightarrow 100\%$. 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 [1]. The minimum requirements of the real-time flows, i.e., all $p_{\text{min}}(g) > 0$, 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 $p_{\text{mm}}(g)$, using just a max–min scheme. Both real-time as well as best-effort flows, i.e., all flows with $p_{\text{newmax}}(g) > 0$, 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.

Another alternate scheme involves pricing of channel time and enforcing priorities based on flow budgets. The max–min fair policy with minimum guarantee lends itself to an elegant two-tier pricing scheme. The guaranteed minimum CTP $p_{\text{min}}(g)$ is valued at a substantial price, whereas any channel time $p_{\text{mm}}(g)$ in excess of this is relatively very cheap. Under this two-tier pricing scheme, users would be inclined to request as little minimum guaranteed bandwidth as possible, in order to save cost. High minimum requirements are thus "punished" while high maximum requirements carry no penalty. The BM adjusts the price so as to trade-off *blocking probability* of the flows with its revenue. If the price is too high, too few flows can afford it and hence blocking probability is high. If the price is low, blocking probability is low, but revenue may suffer. Pricing for wireless networks has been studied previously [17,20,22,26], but our two-tier approach is especially suitable for our bandwidth allocation policy.

2.7. Policing

In our bandwidth management scheme, *policing* refers to the task of monitoring the users, to make sure that they conform to their allocated bandwidth. The bandwidth manager operates in two modes: normal and policing. When operating in *policing mode*, the bandwidth manager listens promiscuously to the network traffic, and checks whether a flow, identified by the source and destination addresses and port numbers in its packet headers, is sending out packets faster than its allotted rate. Additionally, it can also catch those flows who have not registered with the bandwidth manager. 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 $B_{\text{p}}(g)$, 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 physical errors or 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. It could also happen that the unreliable subnet broadcast `reply` message did not reach a particular RA, so a flow is continuing to transmit packets faster than its re-allotted rate.

3. Experimental results

We evaluate the performance of our Admission Control and Dynamic Bandwidth Management system using both a prototype testbed as well as 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.

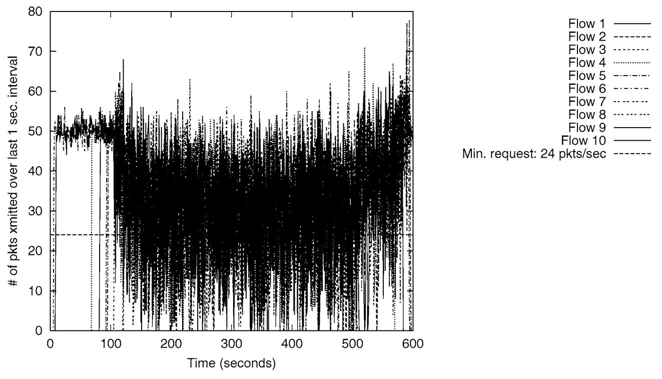
3.1. Simulation experiments

For experiments with large numbers of nodes ($\geqslant$5 nodes) and flows, we used the *ns-2* simulator. We compared the performance of an Admission Control and 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 170 m $\times$ 170 m network area and the transmission range of each node was 250 m. 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.

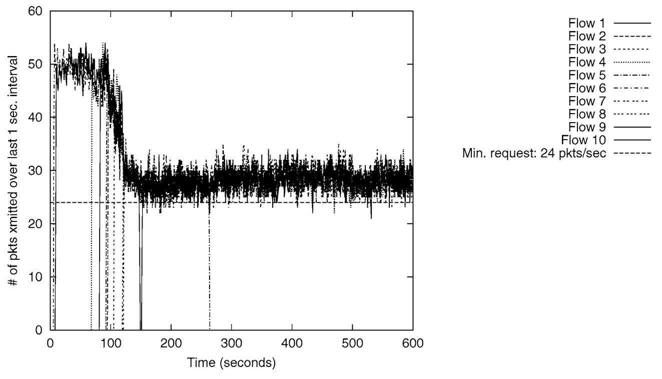
3.1.1. UDP throughput performance

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 section 2.3. 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 $\delta = 15\%$, by default.

Figure 9(a) is a plot of number of packets successfully transmitted over every 1 second interval for each of the 10



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 9. Comparative throughput performance of base and enhanced IEEE 802.11 for 10-flow scenario.

flows using the base IEEE 802.11 scheme. Figure 9(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 pay 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 10 is a 100-second snapshot from the combined plot of figures 9(a) and 9(b) that shows the comparative behavior of a single flow (flow 1).

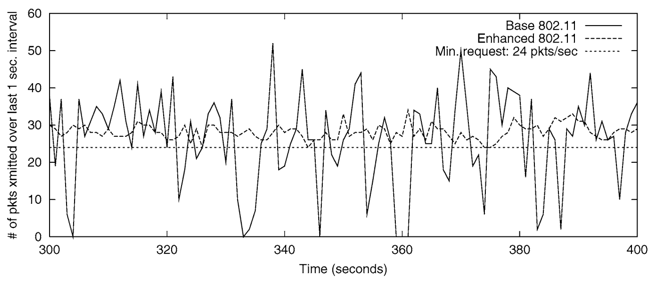
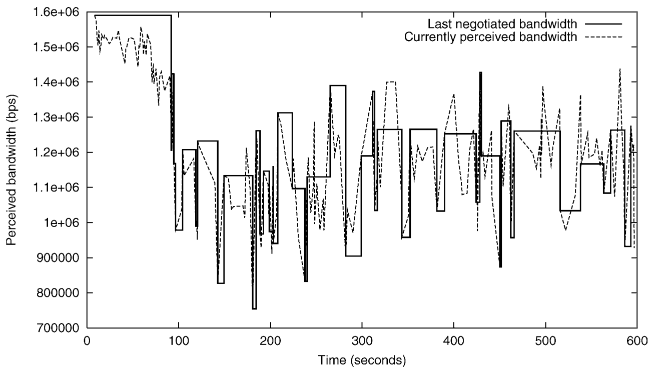
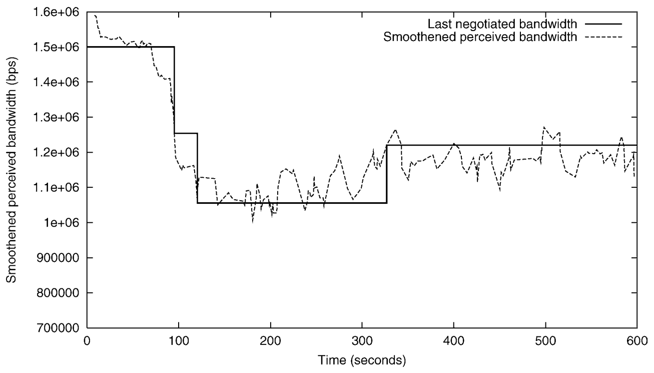


Figure 10. Comparative behavior of a single flow over base 802.11 versus enhanced 802.11.



(a) Without smoothing.



(b) With smoothing.

Figure 11. Perceived bandwidth and re-negotiations corresponding to its variation.

Figure 11(a) shows the variation of perceived bandwidth for one of the flows in the above experiment as measured by its TBE at the MAC layer. The superimposed stepwise curve shows the bandwidth last used for re-negotiation in the above experiment. Recall that $\delta = 15\%$. We also experimented with smoothed perceived bandwidth estimates, which reduced the overhead of re-negotiation frequency. Figure 11(b) is a plot of a running average of the measured perceived bandwidth with exponential decay, which is used for smoothing of the estimate. The smoothed estimate falls as contention increases and rises when the two flows are dropped and contention decreases. Other methods to reduce re-negotiation overhead are described in the next subsection.

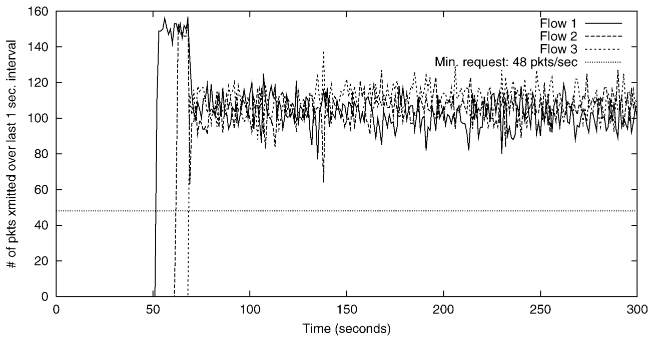
In section 1, we mentioned that improving fairness is essential for providing minimum throughput guarantees to wireless multimedia applications. The key factor enabling our scheme to provide minimum bandwidth requirement guarantees with a high probability, is its improved fairness. No flow takes up excess 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 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 3.2.2, 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 $B_{\mathrm{p}}(f)$ 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.

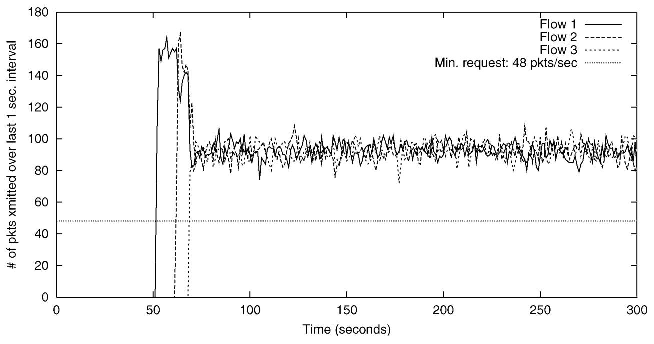
3.1.2. Overhead for UDP experiments

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 dura-



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 12. Comparative throughput performance of base and enhanced IEEE 802.11 for 3-flow scenario with identical bandwidth requirements.

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

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 12(a) and 12(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 smooth-

Table 2
Effect of $B_p(f)$ inter-update period on performance and overhead.

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 3
Effect of various $B_p(f)$ variation tolerance levels δ 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

ing. 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 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.

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 2 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 $\delta = 15\%$ 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 $\delta = 15\%$ change in perceived bandwidth. If we define significant change as, say, a $\delta = 25\%$ 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 3 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 2 and 3 both show that for a small price in terms of performance, we can obtain large gains in overhead reduction.

3.1.3. Additional UDP performance results

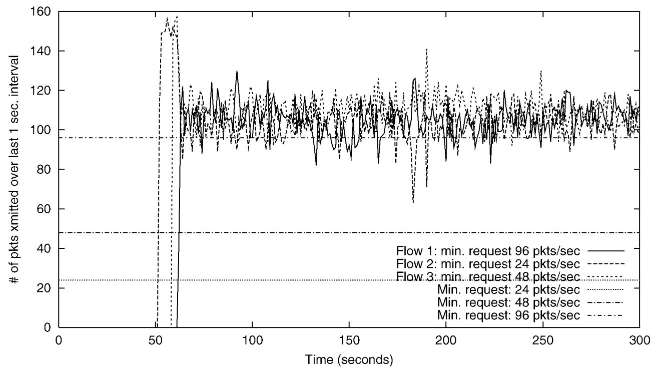
In this section, we present results for two additional scenarios: (a) when the flows have different minimum bandwidth requirements and (b) when the arrival time of the flows is staggered. We use the 6-node, 3-flow scenario used in the previous section, with the default perceived bandwidth tolerance of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

Figure 13 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 802.11, we can make coarse guarantees with our scheme.

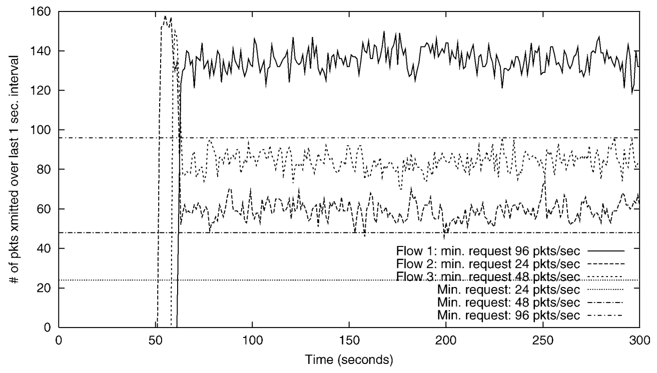
While in all our previous scenarios, all participating flows started at around the same time, figure 14 shows the throughput performance of the enhanced IEEE 802.11 scheme when the start times are staggered. All simulation parameters are identical to those in section 3.1.2, except the staggered start 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 3.1.2. This plot is similar to figure 1 from [5] and figure 11 from [2], which were for a base-station 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.

3.1.4. 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. To this end, 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.

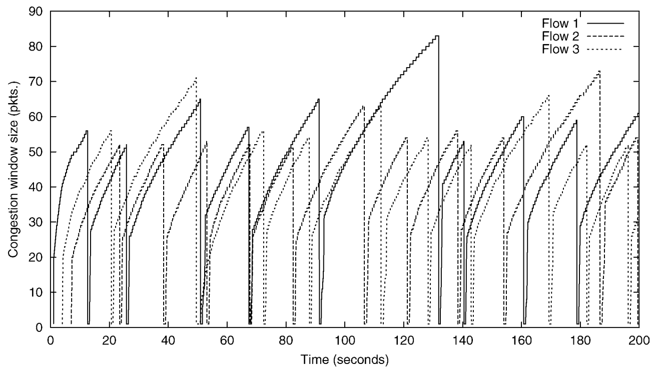


(a) Base IEEE 802.11.

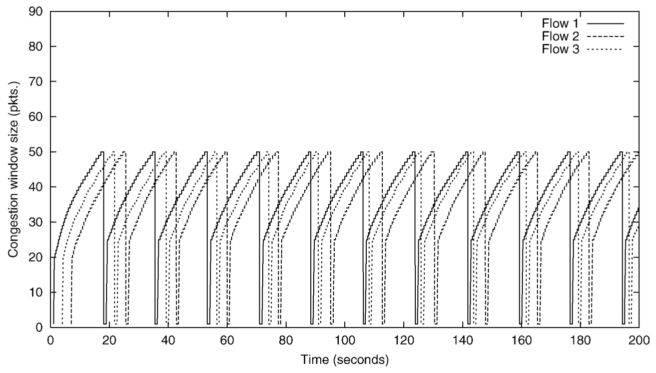


(b) Enhanced IEEE 802.11.

Figure 13. Comparative throughput performance of base and enhanced IEEE 802.11 for 3-flow scenario with different minimum bandwidth requirements.



(a) Base IEEE 802.11.



(b) Enhanced IEEE 802.11.

Figure 15. TCP congestion window behavior when interface queue size is smaller than congestion window limit.

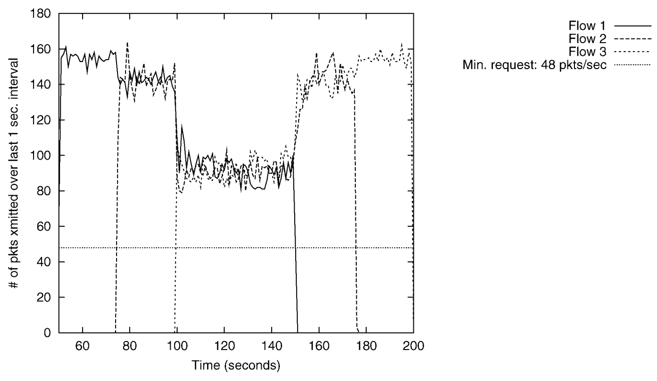


Figure 14. Enhanced IEEE 802.11 performance for 3-flow scenario with staggered start times.

TCP traffic is best-effort and elastic, so $p_{\min}(f)$ is set to zero and $p_{\max}(f)$ to 100%. As mentioned in section 2.3, different $B_{\mathrm{p}}(f)$ values derived from the same normalized bandwidth estimate are used for data and acknowledgements, due to their different packet 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, queue-based 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 15(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 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 15(a) shows that the congestion window sizes of the three flows follow the same "saw-tooth" pattern as in figure 15(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.

Table 4
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	6.70	9.40	565	45065
Enhanced IEEE 802.11	2.12	2.39	33	35698

Table 5
Performance and throughput loss comparison using TCP with interface queue size larger 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

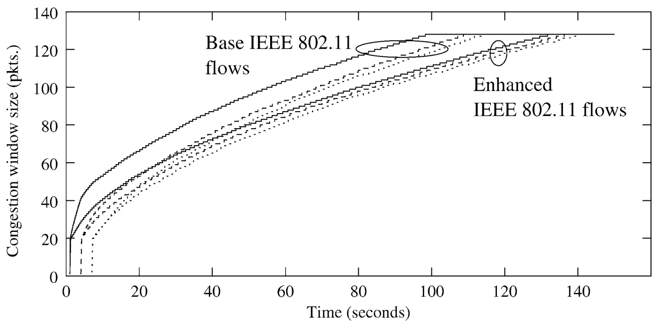


Figure 16. TCP congestion window behavior when interface queue size is larger than congestion window limit.

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 4. 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 3.1.2. We also experimented with less conservative $B_{\mathrm{p}}(f)$ 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 $B_{\mathrm{p}}(f)$ values can be used to trade-off performance (as measured by the *FM* and *JM*) and throughput loss, as with the UDP experiments.

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 16 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 5 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 16, 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.

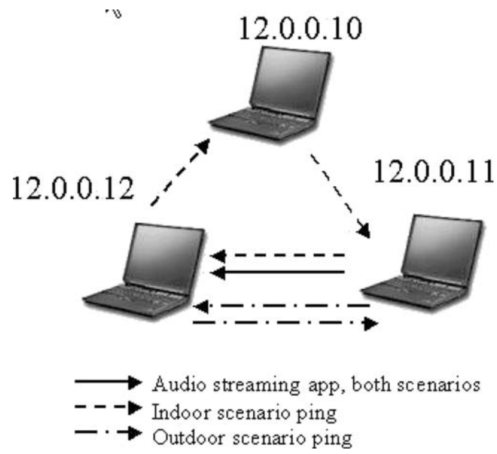
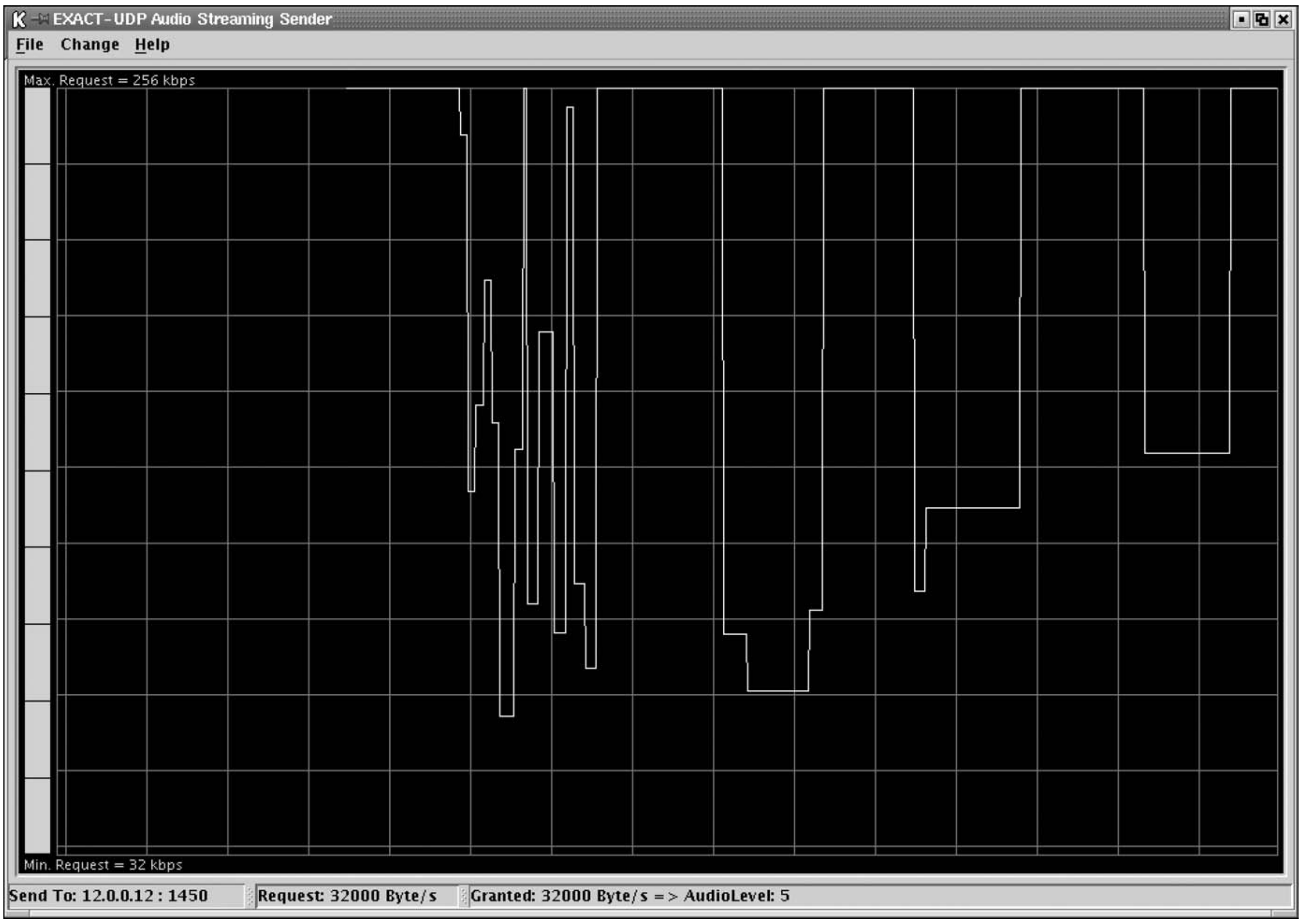


Figure 17. Single-hop ad hoc network testbed.

3.2. 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 active stations. Our testbed (see figure 17) 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. 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.



Send To: 12.0.0.12 : 1450 | Request: 32000 Byte/s | Granted: 32000 Byte/s => AudioLevel: 5

Figure 18. Indoor testbed experiment plot.

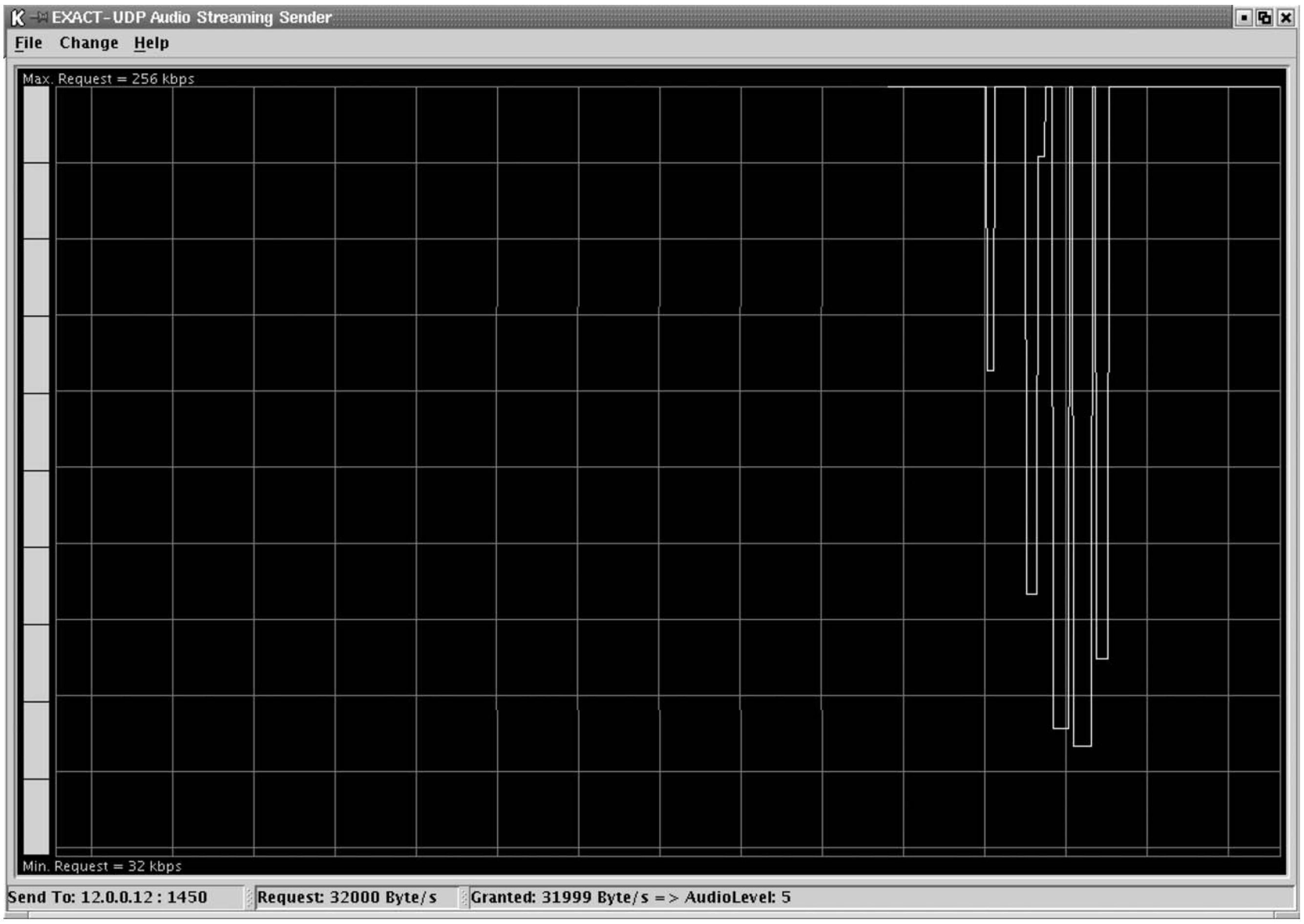
3.2.1. Throughput performance

We conducted two throughput experiments, one *indoors* and one *outdoors*. In each case, we started some unmanaged ping sessions, as shown in figure 17, 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 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 18 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 $\delta = 15\%$ 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



Send To: 12.0.0.12 : 1450 | Request: 32000 Byte/s | Granted: 31999 Byte/s => AudioLevel: 5

Figure 19. Outdoor testbed experiment plot.

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

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. 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.

3.2.2. 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.

4. Related work

In this section we discuss two areas of related work: (a) centralized channel allocation, and (b) distributed fair scheduling in single-hop and multi-hop wireless networks.

In wireless network environment, past research has focused on flow scheduling at the *access-point* to achieve certain fairness criteria between flows competing for the wireless channel [5,7,15]. Bianchi et al. [5] proposed the "utility fair" criteria in bandwidth allocation, where each user's bandwidth is allocated in such a way that their individual utility is equalized. It assumes that the central manager at the base station has exact knowledge of the asymptotic utility curves of all the applications, which might be difficult to obtain. The flows in our scheme can specify a simple linear utility curve using just two points. In our scheme, the BM guarantees a minimum bandwidth for each flow, and allots the rest of the channel capacity in a max–min fashion to each flow up to its maximum request. We believe our approach is simple yet effective in a smart-room where random users walk up to the room and share the wireless channel. Another difference is that we use a distributed peer-to-peer transmission (details in section 2.1), rather than an access-point model, in allocating the channel resources.

Another wireless network channel allocation scheme is the effort-limited fair scheduling by Eckhardt and Steenkiste [9]. 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, when a flow's minimum requirement cannot be satisfied, the flow is simply rejected. This creates incentive for the users to set a minimum channel time requirement as small as possible to reduce the possibility of being denied access to the channel.

In [25], the authors propose an admission control scheme for a peer-to-peer, single-hop, ad hoc wireless network model similar to the one we have used. 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. Moreover, we perform dynamic bandwidth re-negotiation over the course of the connection, in addition to admission control at flow startup.

Another area of related work is the distributed weighted fair scheduling (DWFS) schemes in single-hop and multi-hop wireless networks [3,13,18,19,24]. As mentioned before, our bandwidth management scheme is required to assist the DWFS scheme when it is available. At the same time, as shown in our experiments, our scheme also works well without any underlying DWFS schemes. This is a very important feature because today's IEEE 802.11 network interface card only implements the standard DCF MAC protocol without any DWFS extensions. Therefore, our bandwidth management scheme, of which we already have a working prototype, is highly deployable in today's smart-rooms.

5. Conclusion

In this paper, we presented an Admission Control scheme to determine what fraction of channel time each flow in a single-hop ad hoc wireless network receives. To this end, we mapped the bandwidth requirement at the application/middleware layer to a channel time proportion (CTP) requirement at the MAC layer. We presented an application/middleware layer rate control mechanism to ensure that flows conform to their respective CTPs. Since one-time admission control is not sufficient to handle the changes in network and flow characteristics, we also presented a Dynamic Bandwidth Management system that adapts the flows' respective CTPs during the course of their operation. The adaptation can be a response to change in the network environment or change in a particular flow's traffic characteristics. The simplicity and robustness of our system enables the incorporation of elegant pricing and security features into it. We have developed a prototype implementation of the system and we have used this implementation in a testbed, in addition to extensive simulations, to demonstrate the feasibility and utility of our scheme.

References

[1] G. Ahn, A. Campbell, A. Veres and L. Sun, Swan: Service differentiation in stateless wireless ad hoc networks, in: *Proceedings of IEEE InfoCom*, New York (June 2002).

[2] 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 5(4) (1998) 32–44.

[3] B. Bensaou, Y. Wang and C. Ko, Fair medium access in 802.11 based wireless ad hoc networks, in: *Proceedings of IEEE MobiHoc*, Boston, MA (August 2000).

[4] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed., chapter 6 (Prentice-Hall, 1992).

[5] G. Bianchi, A. Campbell and R. Liao, On utility-fair adaptive services in wireless networks, in: *Proceedings of IEEE/IFIP IWQoS*, Napa, CA (May 1998).

[6] F. Cali, M. Conti and E. Gregori, Dynamic tuning of IEEE 802.11 protocol to achieve a theoretical throughput limit, IEEE/ACM Transactions on Networking 8(6) (2000) 785–799.

[7] T. Chen, P. Krzyzanowski, M. Lyu, C. Sreenan and J. Trotter, A summary of Qos support in SWAN, in: *IEEE/IFIP IWQoS 1998*, Napa, CA (May 1998).

[8] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph and R. Katz, An architecture for a secure service discovery service, in: *Proceedings of ACM MobiCom*, Seattle, WA (August 1999).

[9] D. Eckhardt and P. Steenkiste, Effort-limited fair (ELF) scheduling for wireless networks, in: *Proceedings of IEEE InfoCom*, Tel Aviv, Israel (March 2000).

[10] F. Fitzek and M. Reisslen, Mpeg-4 and h.263 video traces for network performance evaluation, IEEE Network Magazine 15(6) (2001) 40–54, `http://www-tkn.ee.tu-berlin.de/research/trace/stat.html`

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

[12] E. Guttman, C. Perkins, J. Veizades and M. Day, Service location protocol, version 2, RFC 2608 (June 1999).

[13] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi and E. Knightly, Distributed multi-hop scheduling and medium access with delay and throughput constraints, in: *Proceedings of ACM MobiCom*, Rome, Italy (July 2001).

[14] M. Kazantzidis, M. Gerla and S. Lee, Permissible throughput network feedback for adaptive multimedia in AODV Manets, in: *Proceedings of IEEE ICC*, Helsinki, Finland (June 2001).

[15] J. Kim and M. Krunz, Bandwidth allocation in wireless networks with guaranteed packet-loss performance, IEEE/ACM Transactions on Networking 8(3) (2000) 337–349.

[16] A. Kuznetsov, Linux traffic control (tc), `http://www.sparre.dk/pub/linux/tc`

[17] R. Liao, R. Wouhaybi and A. Campbell, Incentive engineering in wireless LAN based access networks, in: *Proceedings of IEEE ICNP*, Paris, France (November 2002).

[18] H. Luo, S. Lu and V. Bharghavan, A new model for packet scheduling in multihop wireless networks, in: *Proceedings of IEEE MobiCom*, Boston, MA (August 2000).

[19] H. Luo, P. Medvedev, J. Cheng and S. Lu, A self-coordinating approach to distributed fair queuing in ad hoc wireless networks, in: *Proceedings of IEEE InfoCom*, Anchorage, AK (April 2001).

[20] P. Marbach and R. Berry, Downlink resource allocation and pricing for wireless networks, in: *Proceedings of IEEE InfoCom*, New York (June 2002).

[21] M. Mirhakkak, N. Schult and D. Thomson, Dynamic bandwidth management and adaptive applications for a variable bandwidth wireless environment, IEEE JSAC 19(10) (2001) 1984–1997.

[22] Y. Qiu and P. Marbach, Bandwidth allocation in ad-hoc networks: A price-based approach, in: *Proceedings of IEEE InfoCom*, San Francisco, CA (March–April 2003).

[23] The network simulator ns-2, `http://www.isi.edu/nsnam/ns/`, updated October 2001.

[24] N. Vaidya, P. Bahl and S. Gupta, Distributed fair scheduling in a wireless LAN, in: *Proceedings of ACM MobiCom*, Boston, MA (August 2000).

[25] S. Valaee and B. Li, Distributed call admission control in wireless ad hoc networks, in: *Proceedings of IEEE VTC*, Vancouver, Canada (September 2002).

[26] Y. Xue, B. Li and K. Nahrstedt, Price-based resource allocation in wireless ad hoc networks, in: *Proceedings of IEEE IWQoS*, Monterey, CA (June 2003).

Samarth Shah received his B.E. degree in computer science and engineering from the University of Madras, India, in 1998. He is currently a Ph.D. candidate in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His interests include quality of service (QoS) in wireless networks.
E-mail: shshah@cs.uiuc.edu

Kai Chen received his B.Eng. degree in computer science from Tsinghua University, Beijing, China, in 1995, and M.S. degree in computer science from the University of Delaware, Newark, Delaware, USA, in 1998. Currently he is a Ph.D. candidate in the Department of Computer Science at the University of Illinois at Urbana-Champaign. From 1998 to 2000, he worked as a research programmer at the National Center for Supercomputing Applications (NCSA) at Urbana, IL, USA. His research interests include mobile ad hoc networks, transport layer issues in mobile networks, quality of service, incentive engineering, and pervasive computing.
E-mail: kaichen@cs.uiuc.edu

Klara Nahrstedt is an Associate Professor at the University of Illinois at Urbana-Champaign, Computer Science Department. Her research interests are directed towards multimedia middleware systems, Quality of Service (QoS), QoS routing, QoS-aware resource management in distributed multimedia systems, and multimedia security. She is the coauthor of the widely used multimedia book *Multimedia: Computing, Communications and Applications* published by Prentice Hall, the recipient of the Early NSF Career Award, the Junior Xerox Award, and the IEEE Communication Society Leonard Abraham Award for Research Achievements. She is the editor-in-chief of ACM/Springer Multimedia Systems Journal, and the Ralph and Catherine Fisher Associate Professor. Klara Nahrstedt received her B.A. in mathematics from Humboldt University, Berlin, in 1984, and M.Sc. degree in numerical analysis from the same university in 1985. She was a research scientist in the Institute for Informatik in Berlin until 1990. In 1995 she received her Ph.D. from the University of Pennsylvania in the Department of Computer and Information Science.
E-mail: klara@cs.uiuc.edu