The TIMELY Adaptive Resource Management Architecture

Vaduvur Bharghavan Kang-Won Lee Songwu Lu Sungwon Ha Jia-Ru Li Dane Dwyer
Coordinated Sciences Laboratory
University of Illinois at Urbana-Champaign
Urbana, Il - 61801
Email: {bharghav, kwlee, slu, s-ha, juru, dwyer}@crhc.uiuc.edu

Abstract

Mobile computing environments that seek to support communication-intensive applications need to provide sustained end-to-end networking resources to static and mobile flows in the presence of scarce and variable wireless bandwidth, bursty wireless channel error, and user mobility. In order to achieve this goal, we present the TIMELY adaptive resource management architecture and algorithms for resource reservation, advance reservation, and resource adaptation in mobile computing environments. A key feature of our approach is the coordination of adaptation between the different layers of the network in order to solve the problems introduced by scarce and dynamic network resources.

1 Introduction

Mobile computing is becoming increasingly popular because of the availability of indoor and outdoor wireless packet networks such as Wavelan, Rangelan, RAM and CDPD. In order to effectively support communication-intensive applications such as Web browsing and multimedia conferencing, future mobile computing environments will need to provide sustained levels of the scarce, dynamic, and shared wireless/wireline network resources to applications and end users.

In the domain of wireline networks, several techniques have been proposed for providing quality of service to applications. The network satisfies the throughput and end-to-end delay requirements of applications by using a variety of mechanisms for resource reservation and packet scheduling. Transport protocols such as TCP and RTP enable end hosts to adapt to the available network resources and accordingly adjust the offered load to the network. Mobile computing environments require similar mechanisms, though they additionally need to address the following unique issues: First, wireless channels are prone to bursty and location-dependent error. Second, contention for the wireless channel is location-dependent. Third, mobile users may move from lightly loaded cells to heavily loaded cells. As a result of the first two reasons, the *wireless channel resources are highly dynamic*. As a result of the third reason, *resource contracts that are made in one cell may not be valid when the user moves to another cell*. Besides, user mobility between cells may cause packet flows to be rerouted, also contributing to changes in the resource availability in the backbone and wireless networks.

Since the dynamics of the resources in mobile computing environments is much more severe than in wireline environments, we perceive the need to provide effective mechanisms for adaptation at multiple layers of the network protocol stack. In particular, each layer must seek to minimize the adaptation that higher layers need to perform, but at the same time, expose the variations in the available resources that enable higher layers to adapt intelligently. This approach is in stark contrast to the traditional TCP/IP standard, wherein each layer independently monitors and adapts to resource changes in the network. The coordination of adaptation among multiple layers of the protocol stack is a key feature of our approach, and is discussed in this paper. While our solutions are specifically targeted to mobile computing environments - where the dynamics of the network resources are more pronounced and more critical - they are also applicable to traditional wireline networks.

Over the past two and a half years, we have developed an adaptive resource management architecture for mobile computing environments, and built a testbed for experimenting with adaptive resource management algorithms at the TIMELY Research Group in the University of Illinois. This testbed has been operational for a year and a half. Our architecture spans the protocol stack, and provides mechanisms for adaptation within each protocol layer as well as coordination of adaptation across protocol layers. The following are the key components of our resource management architecture:

- A service model that captures the requirements of both static and mobile flows in a mobile computing environment.
- A wireless/wireline link scheduling algorithm that can provide rate and delay bounds for packet flows while accounting for location-dependent and bursty wireless channel error.
- A resource reservation protocol that supports both application-initiated resource reservation and network-initiated resource adaptation.
- A revenue model for resource usage and a resource adaptation algorithm that seeks to maximize the network revenue while satisfying the QoS requirements of flows.
- A transport layer protocol that enables applications to specify adaptation-related hints for each packet flow to the network, and performs end-to-end adaptation in coordination with the network and link layers.
- The cooperative adaptation of the key resource management algorithms for reservation, adaptation, and predictive advance reservation.

The purpose of this paper is not to present a single new algorithm. Rather, it is to present a resource management architecture, and describe how different resource management algorithms for adaptation, reservation and advance reservation interact in an integrated services mobile computing environment.

The rest of the paper is organized as follows. Section 2 describes the network, service, and revenue models assumed in this paper. Section 3 presents the adaptive resource management architecture. Section 4 describes the the revenue-based rate adaptation algorithm. Section 5 describes the protocol for resource reservation, adaptation, and predictive advance reservation. Section 6 describes the interaction of adaptation between protocol layers. Section 7 illustrates the concepts presented in previous section using some examples from our testbed, and Section 8 concludes the paper.

2 Network Model

Our mobile computing environment consists of a cellular packet network, with a wireline backbone and base stations which provide network access to mobile hosts via wireless links. We assume that routing and flow management algorithms exist for stationary and mobile hosts.

In the cellular network, two cells are called 'neighbors' if it is possible for a mobile user to execute a handoff between them. A user (or host) who has remained in the same cell for longer than a threshold period of time, T_{th}, is called a 'static' user (or host). A user who has performed a handoff within the threshold period of time is called a 'mobile' user. A unicast flow whose two end-points are both static is called a static flow. If either of its end-points is mobile, the flow is called a mobile flow. Thus, the same flow may be designated as being either static or mobile at different times. *Our goal is to maximize the resources allocated to static flows and minimize the variance in resources allocated to mobile flows.*

2.1 Service Model and QoS Bounds

In a related work, we have introduced the notion of *adaptive service* [20]. In adaptive service, the resource specification for a flow specifies the minimum and maximum bounds for each QoS parameter required by the flow. For example, the resource specification for rate is given by a $[b_{\min}, b_{\max}]$ bound. When the network admits a flow, it guarantees that the rate granted for the flow, b_g, will be at least $b_{\min}$[1]. The flow will pay the network a revenue that is a function of its granted rate b_g rather than the minimum guaranteed rate of $b_{\min}$. Thus, the network has the ability to adjust the granted rate of the flow within the range $[b_{\min}, b_{\max}]$ in order to optimize its own revenue. This model is also applicable for other QoS parameters, notably delay.

Specifying a range rather than a single value is crucial in being able to support efficient resource management in mobile computing environments. Applications in such an environment are expected to handle resource fluctuations within acceptable bounds. For the network, providing a bound offers the flexibility to handle mobility, channel error and dynamic resource variations. Note that guaranteeing a minimum resource reservation to an admitted flow provides a degree of separation between flows, while the ability to dynamically adjust the resource reservation within bounds enables the network to multiplex resources among flows in order to optimize its revenue and also to accommodate resource fluctuations.

Typically, the lower bound (e.g., $b_{\min}$ for rate and $d_{\max}$ for delay) is determined by the minimum requirements for the application, while the upper bound (e.g., $b_{\max}$ for rate and $d_{\min}$ for delay) is determined by how much the user is willing to pay for the service. A flow whose rate allocation equals its maximum requirement (e.g., $b_g = b_{\max}$) is said to be *saturated*.

2.1.1 Resource Model

In this paper, the QoS parameter of interest is rate. Since our focus is on rate, link bandwidth is the resource of interest. For each link, we divide the bandwidth B into three categories: (a) per-flow reserved bandwidth, $b_g(f)$, for each admitted ongoing flow f, (b) common reserved bandwidth, R_g, for advance reservations and handling exception conditions, and (c) aggregate bandwidth, B_b, reserved for best effort traffic. Note that our resource model can be easily extended to support multiple classes of service, by reserving a separate fraction of the bandwidth for each class. We only consider one class in this paper for simplicity.

For the purposes of resource adaptation (described in Section 4), we ignore best effort traffic. Thus, we only consider an effective link bandwidth, $(B - B_b)$, that is adaptively divided among ongoing flows and a common reserved bandwidth. The common reserved bandwidth also has a bound $[R_{\min}, R_{\max}]$, just like any other flow. $R_{\max}$ is a network parameter which bounds the bandwidth that can be set aside for future use, while $R_{\min}$ is a dynamically variable parameter, as defined in the next subsection.

2.1.2 Advance Reservation

In a mobile computing environment, a user may move between cells while continuing to send and receive packets in ongoing flows. In order to provide seamless mobility, the network needs to reduce transient packet loss during the handoff and also provide approximately the same QoS for the flow before and after the handoff. The first requirement can be handled by several possible techniques such as multicasting packets to the two cells or forwarding packets from the old to the new cell during and after handoff. The second requirement is much harder to accomplish, and is addressed in this paper.

While a stationary user cares only about maximizing the resource allocation along the current path of a flow, a mobile user cares more about minimizing the variation in resource allocation across handoffs. The primary requirement for seamless mobility is to reduce or eliminate relative change in QoS upon handoff. The absolute QoS value in itself, is a secondary concern so long as the QoS satisfies the minimum bound. Thus, in adaptive service, we provide only the minimum resource requirement for mobile flows (i.e.,$b_g = b_{\min}$). However, in order to support seamless mobility, we perform advance reservation of resources

[1] As described in [20], all 'guarantees' in our mobile computing environment are *channel-conditioned guarantees* as opposed to absolute guarantees. Thus, the guarantees are valid only if the wireless channel resources do not decrease below a threshold value [16].

in the next predicted cell(s) of the user and along the next predicted route of a mobile flow. The algorithm to predict the next cell of a mobile user is described in [4, 18].

Adaptive service supports two types of QoS specifications for dealing with handoffs: the first is guaranteed per-flow advance reservation, and the second is aggregate advance reservation. For per-flow advance reservation, once the next cell for a mobile flow f is predicted, advance reservation is achieved by explicitly initiating resource reservation along the next-predicted route of f. For aggregate advance reservation, once the next cell for a mobile flow f is predicted, advance reservation is achieved by setting $R_{\min} = \min(R_{\min} + b_{\min}(f), R_{\max})$ for each link along the next-predicted route of f. Advance reservation is cancelled at a link by setting $R_{\min} = \max(R_{\min} - b_{\min}(f), R_{MIN})$, where R_{MIN} is network parameter that provides a lower bound on the common reserved bandwidth. Per-flow advance reservation ensures that the resources reserved in advance for a flow cannot be reused by any other flow, while the resources reserved through aggregate advance reservation can potentially be reused by multiple flows.

2.2 Revenue Model

The notion of revenue, and the revenue model used for resource allocation, are fundamental to rate adaptation in our architecture. We have chosen a large and generally applicable class of revenue models for this paper.

In our revenue model, a flow does not pay the network any revenue unless its granted rate is at least equal to its lower bound. Once the flow is granted its minimum requirement, it pays an *admission fee* (A) for the granted rate. Above the minimum requirement, the flow pays a positive but *decreasing marginal revenue* for each extra unit of aggregate rate[2] that is granted by the network. Above the upper bound (i.e.,for saturated flows), the marginal revenue for each additional unit of rate is 0.

If the network drops an ongoing flow, it pays the user a *termination credit* (T). If the network readjusts the rate of an ongoing flow, it pays the user an *adaptation credit* (C_a) irrespective of whether the new rate is greater or less than the old rate. Thus, the network will cause adaptation upon increase in resource availability in some of its links only if the expected additional revenue generated from granting additional resources will exceed the aggregate adaptation credit granted to all the flows that are adapted. This is critical in reducing the frequency of adaptation when available resources fluctuate rapidly in a mobile computing environment. While our architecture allows for pre-emption of ongoing flows to accommodate higher priority incoming flows, in our testbed we set the termination credit of each flow to be sufficiently large that no ongoing flow is dropped in favor of either a newly arriving flow or rate adaptation for another flow.

We make the following observations based on the revenue model described above.

- In the absence of future adaptations, the rate adaptation algorithm that maximizes long term revenue for a link in isolation distributes the excess rate equally among all the unsaturated flows traversing through the link if the revenue functions for all the flows are identical.

- In absence of complete knowledge of future events such as handoff, link bandwidth change, new connection requests, etc., the network cannot maximize its long term revenue merely by maximizing its short term revenue.

- The admission fee and termination credit are set large enough in our environment that no ongoing flow is terminated or no incoming flow is refused connection only to accommodate increased rate of another ongoing flow (i.e.,a flow is rejected or terminated only if all ongoing flows have $b_g = b_{\min}$ and resources still need to be reclaimed). Thus, the revenue model does not explicitly affect admission control, but it does affect rate adaptation.

[2] Aggregate rate is the sum of the rate allocations over all the links that a flow traverses.

The revenue model of a flow is characterized by the 4-tuple $< A, T, C_a, F >$ - admission fee, termination credit, adaptation credit, and revenue function. In the rest of this paper, we will assume that all flows are governed by the same revenue model, and thus have the same values for the 4-tuple.

2.3 Wireless Channel Resource Model

Unlike wireline networks, wireless networks are prone to channel error and location-dependent contention. Since, the available resources of wireless channels are both scarce and dynamic, any 'guarantees' that are provided to flows in mobile computing environments are channel-conditioned. In a related work [16], we describe an algorithm for wireless fair scheduling that provides rate and delay channel-conditioned guarantees to flows based on estimated wireless channel resources. This algorithm is independent of the specific pattern of wireless channel errors, but needs a bound on the number of slot (or fixed size packet) errors that can happen over any time window of a given length. Our resource management architecture assumes the presence of a resource monitor for providing such a bound based on tracking transmission and reception of packets in the wireless network. Given this bound, our wireless scheduling algorithm can provide channel conditioned guarantees.

3 Resource Management Architecture

Figure 1.A shows our adaptive resource management architecture for mobile computing environments. Our architecture consists of four horizontal layers and a virtual vertical layer. The four horizontal layers are the scheduling/MAC layer [12, 16], the resource reservation and predictive advance reservation layer [4, 18], the resource adaptation layer [17], and the transport layer [10]. The virtual adaptation interaction layer provides for adaptation-related information to be passed across the protocol layers, and enables the different layers to adapt cooperatively. Coordinated adaptation across multiple layers is a key aspect of our resource management architecture. In this paper, the focus is on the middle two layers - resource reservation and resource adaptation. Figure 1.B shows the different resource management algorithms and

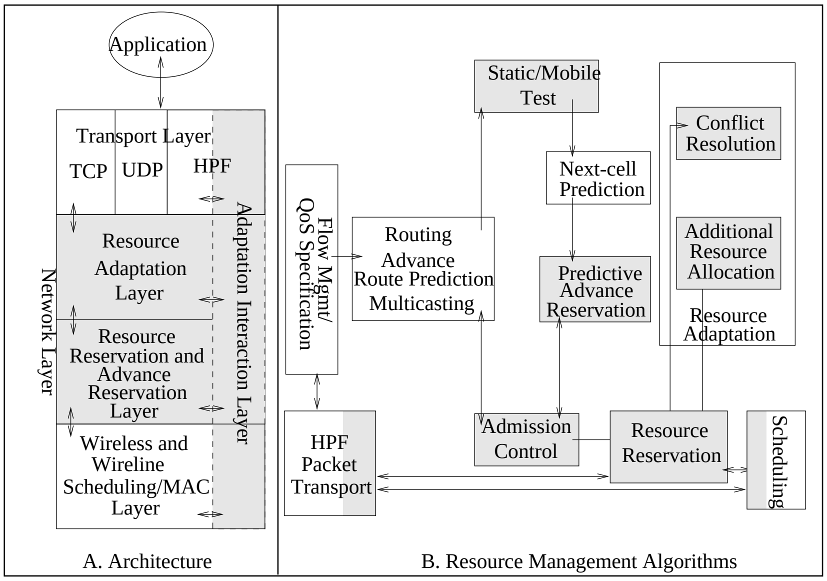

Figure 1: Resource Management Architecture and Algorithms. Figure 1.A shows the protocol stack. Figure 1.B shows the interaction among the resource management algorithms. The shaded areas in the figures highlight the parts of the architecture that are described in this paper.

their interaction. In order to provide adaptive service, various resource management algorithms interact in the following sequence of events.

1. The application notifies the network that it wishes to set up a flow between two end-points, and provides the flow specifications and **resource specifications**.
2. The network performs admission control along the computed route of the flow. When performing **admission control** for a flow, each link tests only whether it can satisfy the minimum requirements of the new flow without violating the *lower bounds* of ongoing flows ($\sum b_{\min}(f)$ for all flows f traversing the link) and the common reserved bandwidth ($R_{\min}$) of the link.
3. Since only the minimum requirements are considered above, an admitted flow may cause a **resource conflict**, in case some ongoing flows have been granted resources above their minimum requirements.
4. The network performs **advance reservation** for mobile flows.
5. The network performs **resource adaptation** among the adaptable static flows in order to resolve resource conflicts and also distribute additional resources.
6. Since the resources allocated to a flow can vary dynamically, applications use an adaptive transport protocol called **HPF** [10] to transmit packets. HPF enables applications to interleave multiple packet substreams with different priorities in a single stream, such that only the important substream is transmitted during sudden resource reductions. Also, the congestion control mechanisms in HPF use the resource adaptation information from the network layer to react effectively to dynamic resource changes.

There is a close interaction between the admission control, resource reservation, advance reservation, and resource adaptation algorithms. There is also a close interaction between the transport protocol, resource reservation, and packet scheduling algorithms.

Admission control determines if a new flow can be admitted without violating the minimum resource bounds of ongoing flows and the common reserved bandwidth. Resource reservation is then performed for the new flow. Advance reservation is only performed for mobile flows in the next-predicted cell(s) and along the next predicted route(s). Note that per-flow advance reservation is similar to regular resource reservation, while aggregate advance reservation increases the lower bound of the common reserved bandwidth. This leads to multiplexing of advance reservations, thereby improving utilization of the network. Resource adaptation is only performed on a subset of static flows, and increases the allocated resources of these flows. Note that while adaptation can cause resource conflict (item 3), it increases the utilization and hence the revenue of the network. One of the critical tasks of a mobile computing environment is to prevent frequent adaptation due to the dynamics of resources and mobility of flows, while still optimizing the utilization and revenue of the network. This is achieved in our approach because the network will never allocate additional rate to flows upon a resource increase unless the expected revenue gain due to the adaptation exceeds the aggregate adaptation credit that the network has to give to the flows it causes to adapt.

While the network initiates adaptation in order to maximize its own revenue, applications need to adapt to the variations in the resources granted to them by the network. When resource changes occur in the network, the application is notified. At the same time, a large number of packets may be in transit, thus the transport protocol needs to gracefully adapt to short term resource fluctuations in the network. In our architecture, the transport protocol provides graceful adaptation through two mechanisms: (a) it still performs window based flow/congestion control, but the window is set to reflect the granted delay-bandwidth product for the flow, and (b) the transport protocol tags packets with priority levels, so that intermediate schedulers can drop low priority packets while still transmitting higher priority packets during periods of sudden congestion or resource decrease. We have noticed from performance measurements [10, 12] that having coordinated window and rate based adaptation works effectively for adapting to longer term resource fluctuations, while having priority-based link level packet dropping during congestion handles short term resource fluctuations effectively.

In the next two sections, we deal with network-level adaptation, while in Section 6, we describe the coordination of adaptation across the protocol stack. There are six events that we need to consider for network-level resource management: (a) connection setup or unexpected handoff into a cell, (b) connection teardown, (c) advance reservation setup, (d) advance reservation teardown, (e) resource increase, and (f) resource decrease. The first four events occur due to mobility or connection setup, while the last two events typically occur due to resource fluctuations in a wireless network. Events (a) and (c) impact admission control and resource reservation, events (c) and (d) impact advance reservation, events (a) and (f) may cause conflict resolution (i.e.,rate adaptation for decreasing resources), while events (b) and (e) may cause adaptation (for increasing resources).

In the following sections, we will describe the rate adaptation algorithm, and the protocol for resource reservation and predictive advance reservation in detail.

4 Revenue-based Rate Adaptation

Since mobile users care more about minimizing the variance in QoS across handoffs, rate adaptation is performed only among static flows. The goal of rate adaptation is two-fold: (a) upon resource increase, allocate additional rate to a subset of ongoing static flows in order to maximize long term revenue gain due to the adaptation, and (b) upon resource decrease, reclaim rate from ongoing flows in order to minimize long term revenue loss due to the adaptation. In a mobile computing environment, resource changes occur frequently as a result of wireless channel dynamics as well as user mobility. However, each adaptation costs the network the adaptation credit C_a for each flow that is required to adapt. Thus, a critical task of the rate adaptation algorithm is to increase revenue due to improved network utilization but avoid repeated adaptation due to resource changes. The rate adaptation algorithm essentially involves answering two questions at any given time: (a) which flows are eligible to adapt, and (b) how rate adaption is performed among the adaptable flows.

4.1 Criteria for Selecting Adaptable Flows: Who Should Adapt

At any time, an *adaptable set* identifies the set of flows in the network which are eligible to adapt. If the network did not have to pay any cost for adaptation (i.e. $C_a = 0$), the adaptable set would consist of all the flows in the network. However, because $C_a > 0$, the adaptable set must consist of those flows from which the network expects a long term revenue gain by causing them to adapt. Unfortunately, it is not possible to determine the optimal adaptable set at any time without knowledge of future resource change and mobility events. We thus adopt a conservative heuristic to select the flows in the adaptable set. A static flow belongs to the adaptable set only if it satisfies one of the following three conditions: (a) the flow has not adapted for a threshold time T_a, (b) the last resource adaptation for the flow resulted in a decrease in its rate, or (c) the additional revenue generated by the last resource adaptation for the flow has paid off its adaptation credit C_a. The conservative heuristic seeks to ensure that the network will not lose revenue in the long term due to adaptations induced by resource increase. Furthermore, frequent adaptation of static flows due to mobility of other flows which share common links is prevented. This last point is very important since it guarantees that frequent mobility or local adaptation will not cause a domino effect of adaptation throughout the network.

4.2 Rate Computation Among Adaptable Flows: How to Adapt

Once we determine the adaptable set, the remaining question is when and how to adapt to resource changes in the network.

4.2.3.1. Resource increase and decrease events: Recall from Section 2.2, that each link has two types of resource allocations: a per-flow $b_g(f)$ for each admitted ongoing flow, and a common reserved fraction R_g.

When available bandwidth increases at a link, (a) if $R_g < R_{\min}$, then bandwidth is allocated to the common reserved bandwidth till $R_g = R_{\min}$; (b) if additional bandwidth is left over, then rate adaptation is performed among the flows in the adaptable set according to the algorithm specified in Section 4.2.3.2. Note that for the purposes of adaptation, the common reserved bandwidth is treated as just another flow with $[R_{\min}, R_{\max}]$ rate bounds, and is always present in the adaptable set.

When available bandwidth decreases at a link and this causes a resource conflict, (a) if $R_g > 0$, then bandwidth is first taken from the common reserved bandwidth till $R_g = 0$; (b) if still more bandwidth needs to be retrieved, then rate adaptation is performed among the flows in the adaptable set according to the algorithm specified in Section 4.2.3.2; (c) if still more bandwidth needs to be retrieved, then rate adaptation is performed among all ongoing flows; (d) if still more bandwidth needs to be retrieved, then flows are dropped one by one in decreasing order of $b_{\min}$.

4.2.3.2. Resource Adaptation Algorithm: The resource adaptation algorithm seeks to distribute additional resources among the flows in the adaptable set in order to maximize revenue increase, or reclaim lost resources from the flows in the adaptable set in order to minimize revenue decrease. Note that all the flows in the adaptable set already have been granted their lower bound $b_{\min}$. Thus, we only deal with excess rate in this algorithm, i.e.,each flow f in the adaptable set is considered to have bounds $[0, b_{\max} - b_{\min}]$ for the purpose of rate adaptation.

We use a weighted version of the max-min fair rate adaptation algorithm for rate adaptation [3, 8]. Each flow is assigned a weight that is equal to the number of hops that it traverses. The weighted max-min rate adaptation algorithm allocates rate among flows in the adaptable set in such a way that the minimum 'weighted rate' among the flows is maximized, and subject to this constraint the next lowest weighted rate is maximized, and so on. Let each flow f be granted a weight w_f and a rate r_f. As in max-min, the weighted max-min rate allocation computes the 'weighted fair share' at each link. Let link l be the bottleneck link for an unsaturated flow i; then $w_i.r_i = w_l^*$, where w_l^* is the weighted fair share for link l. Thus, the algorithm iteratively distributes the rate inversely proportional to the weight of a flow among competing unsaturated flows at the bottleneck links (links with the minimum weighted fair share in each iteration); bottleneck links and the flows that traverse through them are then removed from future iterations.

In a related work [19], we show that the weighted max-min rate adaptation algorithm works very well for maximizing the long term revenue. However, it is well known that max-min and weighted max-min algorithms require global state [8]. Distributed implementations of the max-min algorithm have been proposed in literature, but they have long convergence times and feasibility constraints [9]. In our testbed, we have implemented a maximum-leaf-spanning-tree (MLST) based distributed approximation of the weighted max-min algorithm which converges in $O(log\ D)$ time in the average case and maintains feasibility at all times. While the details of this implementation are beyond the scope of this paper and are described in [19], we will generically refer to the distributed entities of this implementation as *weighted max-min servers* in the next section.

5 Resource Reservation and Advance Reservation

The resource reservation protocol in our architecture needs to support three types of reservations: (a) user-initiated reservation during connection setup, (b) advance reservation for mobile flows, and (c) network-initiated adaptation. In this section, we describe our reservation protocol, and its interaction with resource adaptation.

Figure 2 shows the architecture of the reservation layer. The key entities that participate in the resource

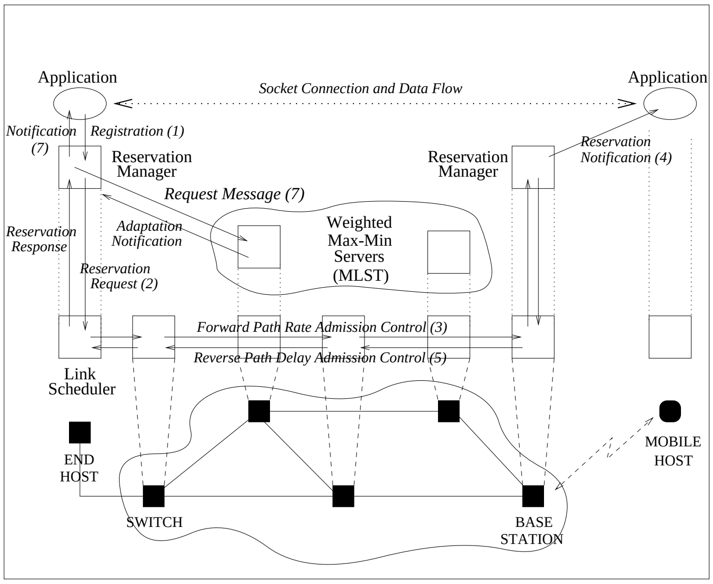

Figure 2: Resource Reservation Protocol. The numbers in the figure correspond to the sequence of steps involved in setting up a new connection reservation request.

reservation protocol are the following: (a) application, (b) resource reservation manager (RM), (c) weighted max-min servers (WMS), and (d) link scheduler at each network switch. The application client and server reside at the two end hosts between which a packet flow is set up. The packet flow may be unidirectional or bidirectional. Each end host has a resource reservation manager which performs the resource reservation and adaptation on behalf of the application. A collection of weighted max-min servers reside in a small subset of the network nodes. Each network switch has a reservation layer component and a scheduler/MAC layer. For the wireless links, the base station undertakes to perform the resource reservation on behalf of the mobile hosts in its cell. Each base station thus has a resource reservation manager. Resource allocation within the cell is done through a wireless packet scheduling algorithm, described in [16].

The resource reservation protocol works as follows. Applications set up TCP, UDP, or HPF [10] connections using standard socket mechanisms. Once a connection is setup, applications may transmit packets using best-effort service till either end-point seeks to make a resource reservation request.

When an application wishes to set up a resource reservation for an ongoing connection (duplex or simplex in either direction), it registers the connection with its local RM, and requests a reservation to be made through a *Reservation Request* message to the RM, specifying its flow specification (*Fspec*) and the resource specification (*Rspec*). In our testbed implementation, Rspec consists of bounds for rate and delay, and the type of advance reservation for mobile flows. Upon reception of a *Reservation Request* message, the resource reservation manager sends a reservation request along the path of the flow.

Resource reservation is a round-trip process, with rate reservation being performed on the forward path, and delay and buffer reservation being done on the reverse path. When an intermediate node sees a reservation request that has been satisfied by all upstream nodes thus far, it performs the local rate admisssion test. Consider a link l, with a bandwidth B, common reserved bandwidth bounds $[R_{\min}, R_{\max}]$, a set of flows C traversing through l, and aggregate bandwidth reserved for best effort traffic B_b. When a request for a new flow f^* comes in, the admission test is successful if $\sum_{i \in C} b_{\min}(i) + b_{\min}(f^*) + R_{\min} \leq B - B_b$, and failed otherwise. If the rate admission test is passed, the link *tentatively admits* the flow f^*, and adds it to C for the purpose of future admission tests. The request message is forwarded downstream with a flag that indicates success or failure thus far, and also the route of the flow.

During the reverse path of the round trip, each link performs the delay and buffer admission tests (which are beyond the scope of this paper). If either test fails, then flow is taken out of the tentatively reserved list and added to the best effort list. If both tests succeed, the flow is added to the reserved list of flows at the link. Irrespective of whether the reservation succeeded or failed, the reservation process takes a single round trip. All reservations time out after a threshold period if not renewed. Thus, for each reserved flow, the RM periodically refreshes its resource reservations. For this reason, if the rate admission succeeds but the delay admission fails, the remote host will learn this fact after the threshold time expires.

Upon the completion of a successful reservation request, the RM forwards reservation request packet of the newly admitted flow (which includes the path of the flow) to the nearest WMS, which then initiates resource adaptation if required.

Note that there are five kinds of resource requests: (a) *new reservation request*, just described above, (b) *refresh request*, which is the periodic refreshing of resources for ongoing flows, (c) *advance reservation request*, for mobile flows along the next-predicted route, (d) *reroute request*, which occurs upon a handoff, and (d) *adaptation request*, for static flows which have been chosen for adaptation. While (a), (d), and (e) cause new resources to be committed to a flow, (b) resets the expiry timer of currently committed resources, and (c) modifies the lower bound of the common reserved bandwidth.

5.1 Mobility and Predictive Advance Reservation

For mobile flows, the RM at the base station handles the resource reservation requests. A mobile flow is explicitly disabled for adaptation. Thus, only after a mobile host has stayed in the same cell for the threshold time T_{th} is any flow from or to the host considered static, and may then be eligible for adaptation according to the rules described in Section 4.

For a mobile flow, the next-cell prediction algorithm predicts the next cell [4, 18]. Predictive advance reservation is initiated by sending the base station of the next cell an advance reservation request with the *Rspec* of the mobile flow. Upon reception of an advance reservation request for a flow, a base station computes a predicted route for the flow and initiates an advance reservation request along the new route. For per-flow advance reservation, the *advance reservation request* is similar to the *new reservation request*. For aggregate advance reservation, an advance reservation request for a flow f does not explicitly reserve any resources at a link. It only updates the lower bound of the common reserved bandwidth $R_{\min} = \min(R_{\min} + b_{\min}(f), R_{\max})$. At the first network switch where the current route and the next-predicted route merge, the reservation flag of the *advance reservation request* is set to 0, so that switches downstream ignore the request. Note that since aggregate advance reservation does not timeout, it needs to be explicitly cancelled. Cancelling an aggregate advance reservation for flow f at a link updates the common reserved bandwidth $R_{\min} = \max(R_{\min} - b_{\min}(f), R_{MIN})$.

Mobility is handled as follows: a handoff need not be accompanied with an explicit teardown of reservations along the current path because they time out automatically. When a mobile flow hands off, the base station of the new cell initiates a *reroute reservation request* along the new route. The case of handoff with a successful per-flow advance reservation is straight forward. Thus, we only present handoff with aggregate advance reservation. If next cell prediction was successful, i.e.,if the flow did hand off into the next predicted cell, then if the aggregate advance reservation was successful for the flow, its resources are transfered from the common reserved bandwidth to the per-flow reservations. Note that this can lead to resource conflicts, since according to the resource adaptation algorithm in Section 4, aggregate advance reserved resources may be reused by ongoing flows to improve network utilization. If the advance reservation for this flow had failed, the flow may be rejected, though it may still be admitted if there are enough resources available in the common reserved bandwidth. This implies that since the common reserved bandwidth is multiplexed among several mobile flows, success of aggregate advance reservation does not guarantee seamless handoff and failure of advance reservation does not guarantee handoff dropping. Thus aggregate advance reservation is enhanced best effort rather than guarantee of seamless mobility, while per-flow advance reservation guarantees seamless mobility. Of course, an ongoing flow will never be rejected so long as it can pass the admission control test even if it involves reclaiming the bandwidth from other flows that have been allocated more than their minimum requirements.

5.2 Interaction of Resource Adaptation and Resource Reservation

As mentioned in Section 4.2, rate adaptation is achieved via a network of WMSs which perform an approximation of the weighted max-min fair algorithm on the set of adaptable flows. When a new reservation request or reroute request is successfully completed, the RM that initiated the request forwards the request packet to the nearest WMS. When an application deregisters a connection with its RM (or when a base station detects a handoff of a mobile host out of its cell), the RM sends a *connection termination notification* to the nearest WMS. Thus, upon a handoff of a mobile flow, the network of WMSs receives a connection termination notification from the old base station and a reroute request from the new base station (assuming that resource reservation was successful along the new route). Because of the asynchronous nature of these two messages, we associate a sequence number with each flow, that is incremented every time a handoff takes place. This ensures that the WMS does not get into the wrong state if an old connection termination notification reaches later than a newer reroute request.

Once the WMS has not received a reroute request or connection termination notification for a flow for a threshold T_{th} period of time, the flow is labelled as static. It is then eligible to adapt according to the algorithm in Section 4. For static flows, the network of WMSs computes the adaptable set and performs resource adaptation among the flows that belong to the adaptable set. If a resource adaptation causes a change in $b_g(f)$ for a flow f, the corresponding RM receives an *adaptation notification* with the flow id and newly granted rate of the f. Upon reception of this notification, the RM initiates a *adaptation request* specifying b_g for the flow, and also notifies the application about the newly granted rate.

Rate adaptation may also be triggered by resource decrease on a link. Each link is assumed to have a *resource monitor* that monitors the usage of link capacity and available resources. When link capacity changes by more than a threshold value due to wireless channel error or location-dependent contention, the resource monitor notifies the nearest WMS, which then initiates adaptation if required. The design of the resource monitor is beyond the scope of this work.

In summary, the main features of the resource reservation protocol are as follows:

- Changes in resource allocation may be *initiated either by end hosts or by the network*. This gives the network the flexibility to change resource allocations within bounds in order to either adapt to the dynamic network conditions or to maximize its own long term revenue.

- Soft state is used to time out resource reservations. Thus, mobile hosts that handoff or get disconnected automatically time out their resource reservations rather than having to explicitly do so. The resource reservation manager at the end host or the base station takes care of the periodic refreshing of resources. As far as applications are concerned, they need to explicitly register and deregister resource reservations with their reservation managers.

- Data transmission and resource reservation are decoupled. An application can send data packets in a best effort manner without resource reservation, and then attempt to reserve resources when it needs to.

- Reservation is a single round trip process, whether the reservation is a success or not.

6 Coordination of Adaptation across Layers

The previous two sections described how the network reserves and dynamically readjusts the resource allocation to flows in the two middle layers of Figure 1.a. In this section, we briefly provide an overview of how the network layer algorithms interact with the algorithms for end-to-end adaptation at the transport layer, and the algorithms for scheduling at the link layer.

6.1 Transport Layer

In order to promote interaction between the application and the network, and support complex packet flows where different packets have diverse QoS requirements (such as multimedia MPEG streams, where I-frames have higher priority than P or B frames, or multiplexed text and image HTTP requests in a single transport connection), we have developed the HPF transport layer protocol for supporting heterogeneous packet flows [10]. The architecture of HPF allows for multiple sub-streams with different priorities, reliability and QoS requirements to be interleaved into a single packet flow that provides sequencing and synchronization between the different substreams. HPF has two main mechanisms for performing end-to-end adaptation: (a) it uses a robust window-based congestion-control algorithm on top of the adaptive resource management algorithms in the network, but uses the granted delay-bandwidth product for reserved flows as an estimator for setting the window size, and (b) it enables applications to provide hints to the network switches about which packets in a flow of multi-priority interleaved streams are high priority, so that intelligent network switches will not drop high priority packets at the expense of low priority packets from the same flow. The first mechanism is useful to provide end-to-end adaptation to long term resource variations in the network, while the second mechanism is useful to deal with short term resource fluctuations in the network. While the details of HPF are beyond the scope of this paper, it is important to note that HPF uses information from the network (e.g. the delay-bandwidth product for a reserved flow) to dynamically adapt its sending rate, and also provides information to the network (e.g. the priority of packets in a flow) so that the network can drop lower priority packets during congestion.

6.2 Link Layer

While our network and transport layer protocols can co-exist with a variety of different scheduling algorithms, we have built two types of schedulers: a hierarchical weighted round robin scheduler with prioritized packet dropping for wireline links, and a wireless packet scheduler with prioritized packet dropping in wireless links.

Each backbone network switch implements a 3-level hierarchical scheduling algorithm [12]. The highest priority *control* level serves signalling packets (such as reservation requests and notifications) in FIFO order. The second priority *reservation* level serves admitted flows according to a weighted round robin (WRR) service discipline. The lowest priority *best effort* level serves unreserved flows according to a round robin (RR) service discipline. When a connection is initially set up and a flow of data packets transmit ted over the connection, each intermediate switch adds the flow to its best effort level. If admission control is successful, the flow is transferred from the best effort level to the reservation level. Likewise, when reservations time out on a flow in the reservation level, it is transferred to the best effort level. State management for a flow is done using soft state.

The wireless packet scheduler (WPS) is a practical implementation of the IWFQ wireless fair scheduling algorithm described in [16]. Both the wireless and wireline schedulers provide priority dropping, which is the link layer support expected by HPF. Each scheduler maintains individual queues for each flow. Within each queue, the scheduler queues packets in order, but has pointers for each priority level which link packets within the priority level. When a new packet arrives, the scheduler tests if queueing the new packet would exceed the buffer bound reserved for the flow. If so, then the packet cannot be queued without having some packet dropped. If the incoming packet has the lowest priority among all queued packets, it is dropped. Otherwise, the first queued packet with the lowest priority is dropped and the incoming packet is queued in sequence.

For the scheduler described above, we make the following observations: (a) priority levels have no effect across flows (i.e.,a lower priority packet of flow 1 is not dropped to accommodate a higher priority packet of flow 2), (b) the scheduling order is not altered within a flow, only the packet dropping is affected (i.e.,within a flow, scheduling is FIFO, but already queued lower priority packets may be dropped in favor of incoming higher priority packets during congestion), and (c) for a flow, the number of packets dropped during congestion does not change; however, the trade off is between the number of lower priority packets dropped versus the number of higher priority packets dropped. Thus, providing priority-based link level packet dropping enables us to deal with short term resources fluctuations by selectively dropping low priority packets, while long-term resource fluctuations are handled more effectively by end-to-end congestion control.

In our testbed, the link layer scheduler provides the mechanisms to implement the resource reservation policy of the network, as well as the prioritized packet dropping policy of the transport. The network layer resource management algorithms interact with both the link layer (to monitor the network resources and to specify the resource allocations to flows) and transport layer (to specify the delay bandwidth product for a flow). The transport layer interacts with both the network (to perform long term resource adaptation) and the link layer (to perform short-term resource adaptation). Thus, the coordination of the transport, network, and link layer protocols to perform adaptation to the dynamics of the network resources enables our environment to react very quickly and gracefully to sudden resource changes in the network. Our experience with the testbed has shown that providing such adaptation can dramatically improve the user experience in a mobile computing environment.

7 Examples

We have built an instantiation of our adaptive resource management architecture in a laboratory testbed, and deployed it for about one and a half years. In this section, we illustrate the the concepts of coordinated multi-layer adaptation using some simple scenarios from our testbed.

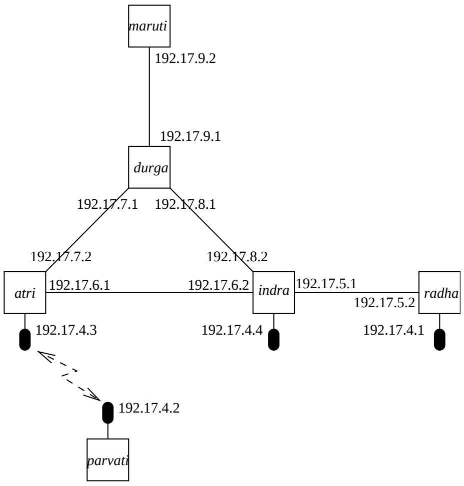

Figure 3: Experimental Testbed Configuration

Figure 3 shows the component of the testbed that is used for our illustrations in this section. There are six hosts and eight subnets, as shown. All our hosts with the exception of *parvati* are static hosts, Gateway P6-200s running Linux 2.0.31. *parvati* is a TI P-120 notebook also running Linux 2.0.31. There are three wireless cells, with hosts *atri*, *indra*, and *radha* acting as the base stations for these cells. The wired backbone consists of point-to-point 10 Mbps Ethernet links (throttled back to 3.3 Mbps in each direction to reduce collisions due to two-way traffic), while the wireless network consists of overlapping 2 Mbps 2.4 GHz WaveLAN cells.

While the link schedulers have been implemented as a part of the Linux kernel, RM and WMS servers are user-level programs. RM runs on all the end hosts in our testbed and WMS runs on *durga*. Each end host shapes the traffic of the flows according to the rate granted to it - either using an explicit traffic shaper (for use with TCP and UDP traffic), or with HPF. HPF is implemented on the end hosts, and is used to show the importance of coordinated adaptation across multiple layers using a network video player example. A more detailed description of the full testbed is found in [19].

We present four illustrative examples to show the different aspects of the resource management architecture. The first example shows how revenue-based adaptation works, and distinguishes it from max-min fair adaptation. The second example shows the impact of advance reservation in supporting seamless mobility. The third example shows that unnecessary adaptations are avoided in our environment. The fourth example compares HPF-based end-to-end adaptation with TCP-based end-to-end adaptation, thereby demonstrating the effect of coordinated adaptation across multiple layers.

For all the examples, the maximum link bandwidth is assumed to be 100 units for the backbone links, where each 'unit' of bandwidth corresponds to 33 Kbps of raw bandwidth. We chose a revenue function, $F(f,l) = 0.01(5b_l r_f - r_f^2)$ where r_f is the excess rate allocated to flow f and b_l is the link capacity over link l. Adaptation credit is set to 200 units and the threshold timer for paying off the adaptation credit, T_a, is scaled down to 5 seconds from a standard value of 5 minutes to induce adaptation more often.

The parameters we measure are the allocated rate for each flow in terms of the bandwidth units, and plots of packet sequence versus time for each flow using `tcpdump` at the receiver. The resource change related events we consider are the following: (a) connection/reservation request, (b) connection/reservation termination, (c) advance reservation, and (d) handoff. The offered load to the network consists of three types of flows: (a) TCP flows with calibrated offered load, (b) HPF flows with callibrated offered load, and (c) an adaptive network video player called VCR from OGI [7].

Example 1: We compare two algorithms for rate adaptation: (a) the revenue-based rate adaptation algorithm proposed in Section 4, and (b) a max-min fair allocation which is a special case of the weighted max-min allocation where all flows have the same weight. Since the max-min fair allocation has been proposed in literature for rate adaptation [9], we compare the two algorithms for a simple case when different flows traverse over different number of links. We establish three static flows: $F1$ along the path *radha* $\rightarrow$ *indra* $\rightarrow$ *durga* $\rightarrow$ *maruti*, $F2$ along the path *atri* $\rightarrow$ *durga* $\rightarrow$ *maruti*, and $F3$ along the path *atri* $\rightarrow$ *durga*. The Rspec for rate for all the flows is $[30, 100]$. $F1$ and $F2$ send 20 MB of data, while $F3$ sends 10 MB of data. All flows are TCP. Table 1 shows the sequence of resource change events, rate allocation and network revenue generated over time using the revenue-based rate adaptation algorithm of Section 4. Table 2 shows the sequence of resource change events, rate allocation and network revenue generated over time using the max-min rate adaptation algorithm. Figure 4 compares the sequence number versus time plots for for the three flows using the revenue-based rate adaptation and max-min adaptation respectively. It turns out in this example, that the revenue-based rate adaptation algorithm generates the optimal long-term revenue.

time	$t=0$	$t=15$	$t=33$	$t=73$	$t=85$	$t=103$	data size
events	F1 $\uparrow$	F2 $\uparrow$	F3 $\uparrow$	F3 $\downarrow$	F1 $\downarrow$	F2 $\downarrow$	
$F1$	100	46	57	46	-	-	$20MB$
$F2$	-	54	43	54	100	-	$20MB$
$F3$	-	-	57	-	-	-	$10MB$
revenue	0	8935	15478	33051	39194	48066	

Table 1: Example 1.a Events and results using the revenue-based rate adaptation algorithm of Section 4. $\uparrow$ denotes a new connection, $\downarrow$ denotes a connection/reservation termination, the numbers in each row corresponding to a flow denote the rate allocated to the flow in bandwidth units, and '-' indicates no resource allocation. Network revenue is calculated according to the function specified at the start of this section.

Example 1 illustrates that when different flows traverse different number of links, the revenue-based rate adaptation algorithm favors shorter flows for adaptation. Note that this has no bearing on the minimum reservations. However, by adapting shorter flows more frequently and with higher priority, the revenue-based rate adaptation algorithm reduces the domino effect of adaptation of longer flows (since they cause

time	$t = 0$	$t = 15$	$t = 30$	$t = 73$	$t = 81$	$t = 97$	data size
events	F1 ↑	F2 ↑	F3 ↑	F3 ↓	F1 ↓	F2 ↓	
$F1$	100	50	50	50	-	-	$20MB$
$F2$	-	50	50	50	100	-	$20MB$
$F3$	-	-	50	-	-	-	$10MB$
$revenue$	0	8326	14646	38526	39160	40178	

Table 2: Example 1.b Events and results using the max-min fair rate adaptation algorithm.

resource change in more links). Thus, longer flows adapt less often, and areas of mobility do not cause adaptation throughout the network.

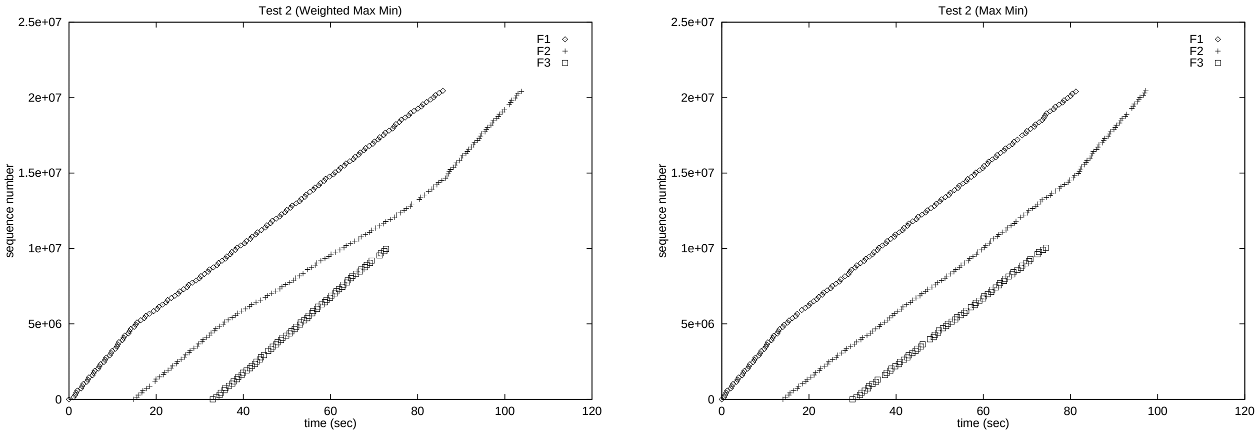

Figure 4: Plot of packet sequence versus time for the revenue-based rate adaptation algorithm and the max-min rate adaptation algorithm. Note that the revenue-based approach favors shorter flows.

Example 2: This example illustrates the interaction between advance reservation, reservation, and adaptation. In particular, we show that per-flow advance reservation ensures seamless mobility.

We establish three flows: static flow $F1$ along the path *atri* $\rightarrow$ *durga* $\rightarrow$ *maruti*, static flow $F3$ along the path *atri* $\rightarrow$ *durga*, and a mobile flow $F2$. The Rspec for rate for $F1$ is $[45, 100]$, for $F2$ is $[20, 100]$, and for $F3$ is $[40, 100]$.

Initially, $F2$ is in the cell of *radha*. $F2$ is set up along the route *parvati* $\rightarrow$ *radha* $\rightarrow$ *indra* $\rightarrow$ *durga*. At time $t = 58$, $F2$ hands off to the cell of *atri*. The new route is *parvati* $\rightarrow$ *atri* $\rightarrow$ *indra* $\rightarrow$ *durga* after handoff. At time $t = 28$, $F2$ performs advance reservation in *atri*. Immediately thereafter, we cause an artificial momentary resource decrease in the link *indra* $\rightarrow$ *durga*. While the advance reservation by itself would not have caused an adaptation (since it only updates the $R_{\min}$ of the link), the resource decrease forces an adaptation due to an apparent resource conflict. The rate allocation of $F1$ is now reduced to 80. Since the advance reservation for flow $F2$ precedes the arrival of flow $F3$, flow $F3$ fails the admission test when it arrives at time $t = 42$. Therefore, flow $F3$ is treated as a best-effort flow by the network. It is the application's task to periodically try to re-establish the reservation (or request for lower reservations), upon failure of the resource reservation request. In this example, flow $F3$ does not try to re-establish the resource reservation, and hence remains a best-effort flow for its lifetime.

Due to the advance reservation, when flow $F2$ hands off to the cell of *atri* at time $t = 58$, it still maintains the same rate $b_{\min}$. From the user perspective, the handoff for flow $F2$ is seamless. Table 3 presents the sequence of events described above. Figure 5.a shows the plot of packet sequence over time for this example.

Now consider the same scenario except that $F2$ does not make an advance reservation in *atri*. We

time	$t=0$	$t=15$	$t=28$	$t=42$	$t=58$	$t=68$	$t=93$	data size
events	F1 $\uparrow$	F2 $\uparrow$	F2 $\Uparrow$	F3 $\uparrow$	F2 $\rightarrow$	F1 $\downarrow$	F2 $\downarrow$	
$F1$	100	100	80	80	80	-	-	$25MB$
$F2$	-	20	20	20	20	20	-	$25MB$
$F3$	-	-	-	BE	BE	BE	BE	$10MB$

Table 3: Example 2 events and results for the per-flow advance reservation case. $\Uparrow$ denotes advance reservation, $\rightarrow$ denotes handoff, and BE denotes best effort. Note that handoff is accompanied by cancellation of advance reservation in the new cell and transfer of these resources to the handing off connection.

time	$t=0$	$t=15$	$t=30$	$t=45$	$t=88$	$t=118$	$t=129$	data size
events	F1 $\uparrow$	F2 $\uparrow$	F3 $\uparrow$	F2 $\rightarrow$	F3 $\downarrow$	F1 $\downarrow$	F2 $\downarrow$	
$F1$	100	100	50	50	100	-	-	$25MB$
$F2$	-	20	20	BE	BE	BE	-	$25MB$
$F3$	-	-	50	50	-	-	-	$10MB$

Table 4: Example 2 events and results for the no advance reservation case.

assume that the reserved fraction is 0 in this scenario to show the worst case. When $F2$ hands off to the cell of *atri* at time $t = 45$, it fails the admission test and is treated as a best-effort flow. Therefore, it experiences significant rate loss after handoff. After flow $F3$ terminates, flow $F1$ essentially captures the channel and starves flow $F2$. Table 4 shows the sequence of events and rate allocations, while Figure 5.b shows the plot of packet sequence numbers versus time for this scenario.

Example 3: This example illustrates the fact that as long as the rate allocated on a bottleneck link does not change, the handoff of a mobile host does not trigger resource adaptation among reserved flows. We establish three flows: a static flow $F1$ along the path *radha* $\rightarrow$ *indra* $\rightarrow$ *durga* $\rightarrow$ *maruti*, a static flow $F3$ along the path *indra* $\rightarrow$ *atri* $\rightarrow$ *durga*, and a mobile flow $F2$. The Rspec for $F1$ and $F3$ are $[40, 100]$ each, and the Rspec for $F3$ is $[20, 100]$.

Initially, *parvati* is in the cell of *radha*. $F2$ is set up along the path *parvati* $\rightarrow$ *radha* $\rightarrow$ *indra* $\rightarrow$ *durga* $\rightarrow$ *maruti*. At time $t = 45$, *parvati* hands off to the cell of *indra*. The new route for $F2$ is *parvati* $\rightarrow$ *indra* $\rightarrow$ *durga* $\rightarrow$ *maruti*. At time $t = 61$, *parvati* hands off to the cell of *atri*. The new route for $F2$ is *parvati* $\rightarrow$ *atri* $\rightarrow$ *indra* $\rightarrow$ *durga* $\rightarrow$ *maruti*. In this example, advance reservation does not make a difference since there are available resources in the next cell upon a handoff.

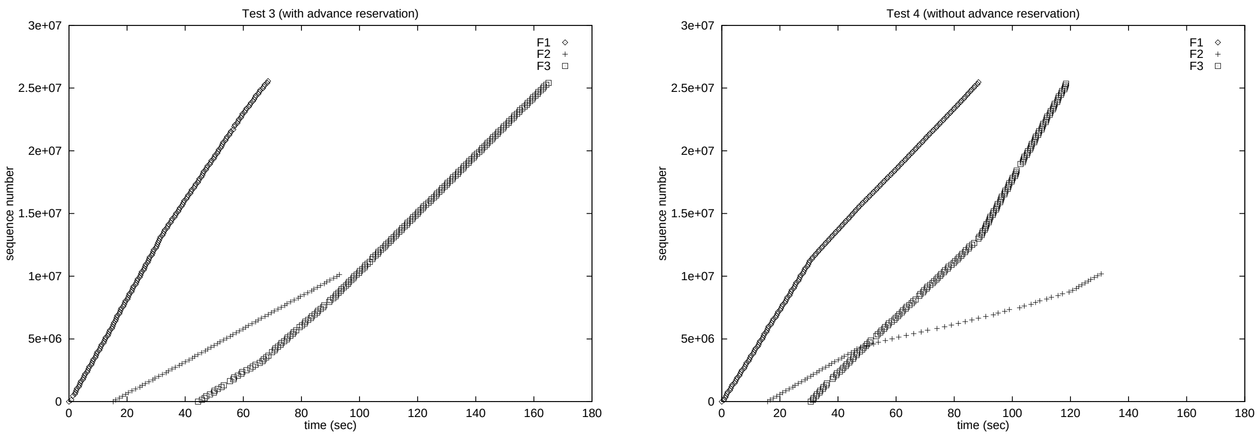

Figure 5: Plot of packet sequence versus time for TCP connections using the experimental setup of Example 2.

Consider that the adaptation credit for $F3$ is large. While *parvati* roams around in the three cells, and frequently hands off between *atri* and *indra*, the granted rate for the flows does not change: the rate for $F1$ will not change because $F1$ and $F2$ share a bottleneck link; the rate for $F3$ will not change until it has 'paid off' its adaptation credit. Table 5 and Figure 6 show the rate allocation and packet sequence versus time for this example.

time	$t = 0$	$t = 15$	$t = 30$	$t = 45$	$t = 61$	$t = 88$	$t = 95$	$t = 100$	data size
events	F1 $\uparrow$	F2 $\uparrow$	F3 $\uparrow$	F2 $\rightarrow$	F2 $\rightarrow$	F1 $\downarrow$	F3 $\downarrow$	F2 $\downarrow$	
$F1$	100	80	80	80	80	-	-	-	$25MB$
$F2$	-	20	20	20	20	20	-	-	$25MB$
$F3$	-	-	100	100	80	80	100	-	$10MB$

Table 5: Example 3 Events and Results.

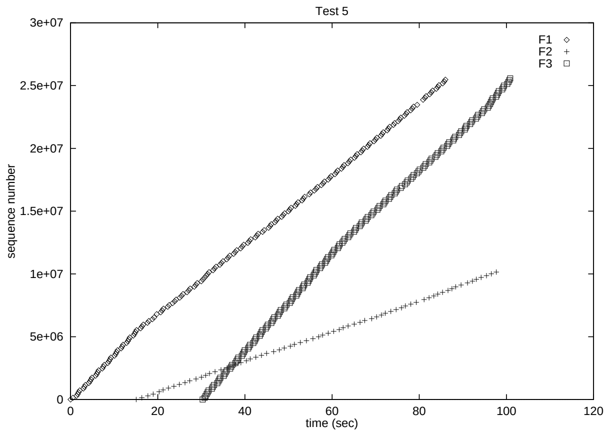

Figure 6: Example 3

Example 4: In this example, we illustrate the impact of end-to-end adaptation and multi-priority flows in HPF on a multimedia application in a mobile computing environment. We use a network MPEG player called VCR [7] as our multimedia application, and compare it over three transports: TCP, UDP, and HPF. For VCR over HPF, we label the I frames of the MPEG stream as high priority, and the P and B frames as low priority. We establish three flows in this example: an unreserved static VCR flow $F1$ along the path *durga* $\rightarrow$ *indra* $\rightarrow$ *radha*, a reserved mobile VCR flow $F2$ along the path *durga* $\rightarrow$ *indra* $\rightarrow$ *radha* $\rightarrow$ *pavarti*, and a reserved TCP flow $F3$ along the path *maruti* $\rightarrow$ *durga* $\rightarrow$ *indra*. The Rspecs of $F2$ and $F3$ are [10, 100] and [50, 100], respectively.

The test scenario is as follows: First, the unreserved VCR flow $F1$ starts. After 15 seconds, the reserved VCR flow $F2$ starts. We instrumented the traffic such that the transmission rate of a single VCR flow takes more than a half of the link bandwidth, but cannot saturate the link. Hence, upon the start of $F2$, the performance of $F1$ degrades. After 40 seconds, flow $F3$ starts. Since $F3$ has reserved a large share of the link bandwidth, it causes a severe degradation in the performance of both $F1$ and $F2$. In particular, the rate reservation for $F2$ is insufficient to maintain its nominal frame rate. Since $F2$ and $F3$ together saturate the link, $F1$ gets starved during the congestion period. After 70 seconds, $F2$ hands off to the cell of *atri*. Shortly thereafter, $F3$ stops.

Figure 7 shows the performance of VCR over HPF for $F1$ and $F2$, while Figure 8 shows the performance of VCR over TCP for $F1$ and $F2$.

For the unreserved flow $F1$, if we run VCR over UDP, $F1$ never recovers from the starvation, and basically stops midway through the program. If we run VCR over TCP or HPF, $F1$ eventually recovers,

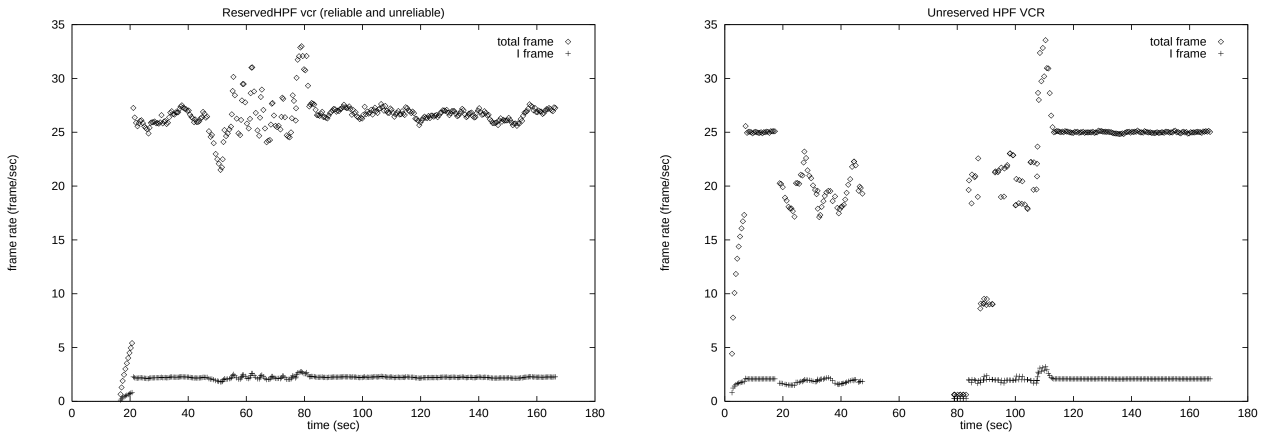

Figure 7: VCR over HPF: plot of frame rate versus time for the reserved VCR flow $F2$ and the unreserved VCR flow $F1$.

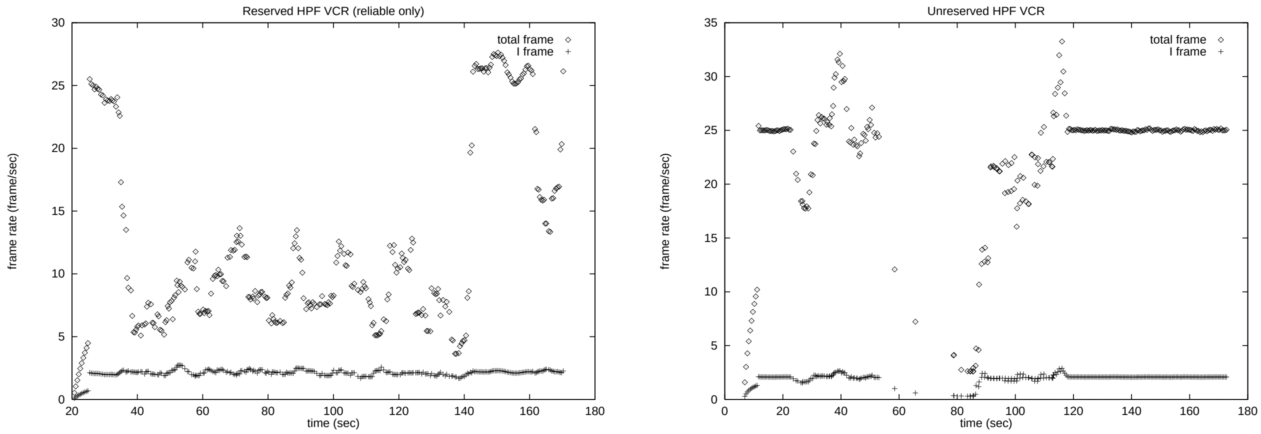

Figure 8: VCR over TCP: plot of frame rate versus time for the reserved VCR flow $F2$ and the unreserved VCR flow $F1$.

but takes several seconds to get to full speed even after $F3$ leaves - this is because of the high `RTO` value of $F1$ during the period of congestion. For HPF, the recovery is slightly faster than TCP because loss of P and B frames during the recovery phase does not slow down HPF.

For the reserved flow $F2$, VCR over HPF shows a very significant improvement compared to VCR over TCP. This improvement can be attributed to the two key adaptation mechanisms of VCR working in concert with the network layer. In fact, we expected an even smoother graph for VCR over HPF for the reserved flow when congestion is introduced. In theory, the design of the end-to-end adaptation in HPF should shield $F2$ from variations we observed during the congestion period. A careful study showed that our current HPF implementation reacts very quickly to short term resources fluctuations when the window size for the HPF connection is small - which is the case in this example. Ongoing work seeks to solve this problem and provide a more robust solution.

8 Conclusion

Future mobile computing environments will need to support communication-intensive applications such as web browsing and multimedia conferencing. In order to support such applications over scarce, shared, and dynamically varying networking resources, effective algorithms for resource management algorithms are crucial.

In this paper, we present the TIMELY adaptive resource management architecture for mobile computing environments. This architecture has four layers - link, reservation, adaptation, and transport - all of which perform resource adaptation in a coordinated manner. The goal of each layer is to minimize the amount of adaptation that higher layers need to perform, but at the same time, expose the resource variations to the higher layers to enable them to adapt intelligently.

Over the last year and a half, we have deployed a testbed that provides a full instantiation of our adaptive resource management architecture in a packet cellular mobile computing environment. Our experience with the testbed, both for artificially generated workloads and for standard applications such as multimedia, file transfer, and web browsers, has demonstrated that coordinated adaptation to the dynamics of the network resources at multiple layers is the key to supporting communication-intensive applications in mobile computing environments.

References

[1] A. Acampora and M. Naghshineh, "An architecture and methodology for mobile-executed handoff in cellular ATM networks," *IEEE Journal on Selected Areas in Communication*, vol. 12, no. 8, October 1994.

[2] P. Bahl, I. Chlamtac and A. Farago, "Optimizing resource utilization in wireless multimedia networks," *Proceedings of IEEE ICC*, Montreal, Canada, June 1997.

[3] D. Bertsekas and R. Gallagher, *Data Networks*, Prentice Hall, Engelwood Cliffs, NJ, 1989.

[4] V. Bharghavan and M.P. Jayanth, "Profile-based next-cell prediction in indoor wireless LANs," *Proceedings of IEEE Singapore International Conference on Networking*, Singapore, April 1997.

[5] R. Braden, D. Clark and S. Shenker, "Integrated services in an internet architecture," *RFC-1633*, June 1994.

[6] A.T. Campbell, G. Coulson, and D. Hutchison, "A quality of service architecture," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 2, April 1994.

[7] S. Cen, C. Pu, R. Staehli, C. Cowan and J. Walpole, "A distributed real-time MPEG video audio player," *Proceedings of NOSSDAV*, Durham, North Carolina, April 1995.

[8] A. Charny, D. Clark and R. Jain, "Congestion Control with Explicit Rate Indication," *Proceedings of ICC'95*, Seattle, Washington, June 1995.

[9] A. Charny, K. K. Ramakrishna and A. Lauck, "Time scale analysis and scalability issues for explicit rate allocation in ATM networks," *IEEE Transactions on Networking*, vol. 4, no. 4, August 1996.

[10] D. Dwyer, S. Ha, J. Li and V. Bharghavan, "An adaptive transport protocol for multimedia communications," *To appear in the International Conference on Multimedia Computing Systems*, Durham, North Carolina, August 1998.

[11] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in building wireless network," *Proceedings of ACM SIGCOMM*, Stanford, California, August 1996.

[12] S. Ha, K-W. Lee, and V. Bharghavan, "Performance Evaluation of Scheduling and Resource Reservation Algorithms in an Integrated Packet Services Network Environment," *To appear in the International Symposium on Computers and Communications*, Athens, Greece, June 1998.

[13] S. Jamin, P. Danzig, S. Shenker and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," *Proceedings of ACM SIGCOMM*, Cambridge, Massachussetts, 1995.

[14] K. Lee, "Adaptive network support for multimedia," *Proceedings of ACM MOBICOM*, Berkeley, California, November 1995.

[15] D. Levine, I. F. Akyildiz, and M. Naghshineh, "A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, February 1997.

[16] S. Lu, V. Bharghavan, and R. Srikant, "Fair Scheduling in Wireless Packet Networks," *Proceedings of ACM SIGCOMM*, Cannes, France, August 1997.

[17] S. Lu and V. Bharghavan, "Adaptive Resource Management Algorithms for Indoor Mobile Computing Environments," *Proceedings of ACM SIGCOMM*, Stanford, California, August 1996.

[18] S. Lu, R. Srikant and V. Bharghavan, "Advance resource reservation algorithms for indoor wireless LANs," *Proceedings of IEEE GLOBECOM*, London, October 1996.

[19] S. Lu and V. Bharghavan, "Rate Adaptation in Cellular Packet Networks," *Internal Technical Report*, TIMELY Research Group, University of Illinois, `http://timely.crhc.uiuc.edu`, June 1997.

[20] S. Lu, K-W. Lee and V. Bharghavan, "Adaptive Service in Mobile Computing Environments," *Proceedings of IFIP IWQoS*, New York, New York, May 1997.

[21] C. Noblet, B. Jafarian and A.H. Aghvami, "Quality of service and overload concepts in wireless multimedia networks based on cellular ATM," *Proceedings of IEEE International Conference on Communications*, Montreal, Canada, June 1997.

[22] G. T. Nguyen, R. H. Katz, B. Noble and M. Satyanarayan, "A Trace-based Approach for Modeling Wireless Channel Behavior," *Proceedings of Winter Simulation Conference*, Coronado, California, December 1996.

[23] S. Shenker, C. Partridge and R. Guerin, "Specification of guaranteed quality of service," *draft-ieft-interv-guaranteed-svc-07.txt*, IETF Draft, February 1997.

[24] S. Singh, "Quality of service guarantees in mobile computing," *Journal on Computer Communications*, vol. 19, 1996.

[25] W. R. Stevens. *TCP/IP Illustrated, Volume 1 and Volume2.* Addison-Wesley, 1994.

[26] A. K. Talukdar, B. R. Badrinath and A. Acharya, "On accommodating mobile hosts in an integrated services packet network,"*Proceedings of IEEE INFOCOM*, Kobe, Japan, April 1997.

[27] J. Wroclawski, "Specification of the controlled-load network element service," *draft-ietf-intserv-ctrl-load-svc-05.txt*, IETF Draft, May 1997.

[28] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP: A new Resource reSerVation Protocol," *IEEE Network*, September 1993.