A Case for Customizable Resource Management in Networks

Peter Steenkiste Allan Fisher Hui Zhang

October 21, 1998
CMU-CS-98-167

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We envision the deployment of an electronic services market that will deliver a wide range of electronic services over networks. This market will allow applications to combine resources at endpoints with resources inside the network to deliver high-quality products to end-users. Electronic services will range from simple data delivery services to sophisticated value-added services such as video conferencing and data mining. We argue that the deployment of such a diverse set of services will be facilitated by the presence of an integrated set of customizable resource management mechanisms. In this paper, we introduce three concepts that respond to this need. First, the resources allocated to an application will be integrated in a virtual network that forms the basis of runtime resource management and quality of service optimization. Second, each resource is managed by a hierarchical resource manager that satisfies the combined priorities and constraints of the services and applications sharing the resource. Finally, since both network conditions and application requirements can change, application-specific adaptation is needed to optimize quality of service at runtime. The CMU Darwin project is developing a comprehensive set of customizable resource management mechanisms based on these concepts. We outline the Darwin resource management mechanisms and we describe an initial implementation.

This research was supported by the Advanced Research Projects Agency/ITO monitored by SPAWAR Systems Center under contract N66001-96-C-8528.

Keywords: resource management, hierarchical scheduling, active networks, congestion control, virtual networks, application-specific adaptation, network-aware applications.

1 An Electronic Services Market

While for several decades, computer networks have primarily been used to move data using programs such as ftp and telnet, they increasingly are being used for more complex tasks (e.g., information retrieval), or as tools in support of inter-personal communication (e.g., video conferencing). We expect this trend to continue, and to lead to the creation of a sophisticated electronic services market that can deliver a wide range of services over the network. These services will often integrate communication with computation and storage access. The ability to invoke value-added services in the same way that we now invoke more traditional "bitway" services will create exciting new opportunities for computer users: an application will be able to combine resources available at the end-points with a rich set of services delivered by the network to create a high-quality electronic end-product for its users. While applications may in some cases invoke only simple services, e.g. bitway services, in many cases the entire end-product may be delivered by the network, and the "application" on the end-point be just a user interface, e.g. a web browser.

The delivery of such value-added services requires appropriate resource management mechanisms that control who can use what resources inside the network. An abstract view is that for each service request from an application, the service provider will allocate an integrated set of resources—including communication, computing and storage resources—that satisfies the request, and the service provider will then use these resources during the application session to meet application needs. In addition to this idealized view, we also envision an approach in which a service provider does not actually own the resources, but instead manages resources already held or acquired by the application. We expect that both during the initial allocation phase and throughout the session, the provider will want to manage resource selection and use so that it can optimize the quality of the particular service being provided. In other words, it must be possible to tailor resource management to meet specific application and service requirements.

Besides appropriate resource management support, many other system components are needed before such complex services can be deployed. Examples include support for service composition, resource and service discovery, and electronic commerce. In particular, resource management, the focus of this paper, is a key building block in the implementation of an electronic services market. Each of the resource management mechanisms we present also has the important property that it is useful in combination with existing mechanism, even if the other components described are not present.

We first use a simple example to illustrate the envisioned services and to identify specific system requirements. We then present three concepts that cooperatively support such services: the virtual network, hierarchical scheduling, and customizable runtime resource management. Finally, we describe the initial implementation of Darwin, a prototype realization of these ideas, and we conclude with a discussion of related work and a summary.

2 Service requirements

While we are already experiencing the delivery of services over the Internet today, these services typically have fairly modest networking requirements compared with the services envisioned in this paper. We will use an example to illustrate the characteristics of the envisioned services and the resulting resource management requirements.

2.1 A simple example

At a high level we can characterize the envisioned electronic services market as providing complex, value-added network services. *Complex* means that the services will support multi-party applications that use a wide variety of data types that may impose challenging timing and synchronization constraints on the processing and transfer of the data. *Value-added* means that the services will in general not just perform

low-level data transfer and manipulation, but can present high-level functionality based on lower-level component services.

The prototypical application that would benefit from a dynamic electronic services market is a collaborative work environment combining traditional video conferencing with joint access by the participants to large amounts of archived data, real time data streams, and interaction with distributed computing tasks. A very simple version of this scenario is a group of scientists who have a (virtual) meeting to discuss the output of a simulation, and to compare it with archived results from experiments. A similar but more ambitious example is disaster control, e.g. planning the response to a forest fire. In this case, a group of specialists convene periodically to discuss status and actions. In their virtual meeting, they need access to live data feeds from the disaster area, various types of archived data that can be used to assess damage or compare with previous disasters, and the output of simulations that evaluate various control strategies. In both cases, the quality of delivered services must be very high to support collaboration effectively.

Even this simple virtual meeting example needs a diverse set of resources. It requires communication bandwidth to carry the video and audio between the participants. It requires computational resources to execute the simulation and additional communication resources to distribute the output of the simulation to the scientists. Depending on conditions, additional resources may be required. For example, video mixing resources may be needed for high-end video conferencing support. Similarly, if there is a mismatch between the video capabilities of the participants, video transcoders may have to be added. Finally, the simulation may need access to archived data that is available only in certain locations.

2.2 Service characteristics

We use the example to identify key service characteristics and the resource management features that are needed to support them efficiently.

A first service feature is that services need a collection of resources of diverse types. This means that it is advantageous to allocate resources in a coordinated fashion. For example, the simulation engines would ideally be placed close to participating scientists to minimize communication costs. Similarly, the simulation engines should be placed in locations that have sufficient communication bandwidth. While it is possible to allocate the resources one at a time in an uncoordinated fashion, this is likely to increase the cost and may degrade quality.

Second, providers will want to control the "quality of the service" they provide. For example, a provider could offer high quality video conferencing for board-level meetings, and a lower quality version for informal discussions. This requires resource management mechanisms that give different levels (or type, quality, etc.) of service to different users. Recent research efforts on support for integrated services networks [26, 34, 25] partially address this issue. However, these mechanisms focus on the quality of the service received by individual flows, rather than entire applications. Mechanism are needed that will map different levels of application-level QoS (e.g. minimal video frame rate) into appropriate per-flow QoS specifications (e.g. controlled load service). However, the specific low-level QoS mechanisms that are needed to maintain a fixed high-level QoS may depend on the load conditions in the network, i.e. the most appropriate mapping may change over time and may be different for different network segments. Current research has mostly focused on fixed mappings, e.g. MPEG-1 video translates into a reservation for 1.5 Mbps. Static mappings are likely to be inefficient, e.g. they may allocate too many resources or use reservations when best effort service suffices.

Third, the definition of high (or low) quality for complex services is service specific, so for maximum efficiency service providers will need sufficient control over resources to apply them in a way that optimizes the quality of the specific service being provided. This means that the resource management mechanisms should be customizable in a number of ways. First, providers will want to have control over what resources are allocated, as described above. Moreover, since network conditions change over time, providers will

want to be involved in how resource use is changed. For example, while some providers might be happy with a proportional redistribution of bandwidth, others might want to protect higher priority flows, or even change the number of flows. How quickly this adaptation can happen is also important for the quality of service perceived by the application. For example, applications such as long data transfers can live with large fluctuations in bandwidth, while others, such as visualization, will want to respond quickly, since in the time interval between the change and the adaptation, the user may receive poor service.

In summary, the service characteristics suggest that we need mechanisms for coordinated and customizable resource management not only in space (i.e. type and location of resources) but also in time. The latter means that resource allocation activities occur on a number of time scales (i.e. application start up time, round trip times, packet times) and these activities should be coordinated. For example, resource allocation at start up time should be able to choose between the use of reservations on one hand, or best effort service augmented with appropriate runtime adaptation on the other. Without appropriate support, service quality maybe poor and unpredictable, or peak allocation of resources may be needed. We are not aware of any existing system that provides systematic support for customization of network resource management over a broad range of time scales.

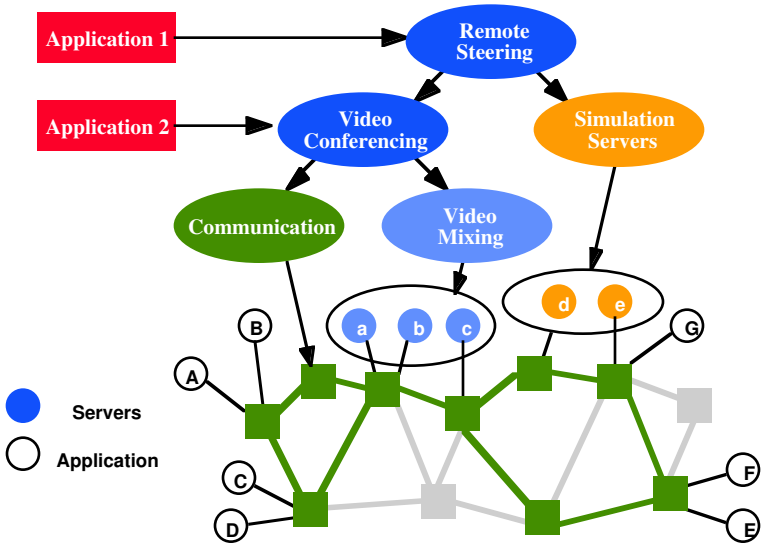

Figure 1: Example service

2.3 Hierarchical services

While services can certainly be delivered in a monolithic way, i.e. a single service provider implements the complete service on top of the bare hardware, it seems more likely that electronic services will be structured in a hierarchical fashion: value-added services are implemented in terms of lower-level services. The motivation is reuse of functionality and need for specialization. Specialized service providers will develop, and other providers will want to incorporate their services instead of trying to develop expertise in that particular area. Figure 1 illustrates one way in which the needs of our scientists can be met by a service hierarchy. One of the users issues a request to a service provider; the request specifies the location of the scientists, the quality of the video and the approximate computational and I/O requirements of the simulation. This particular service provider does not implement this functionality, but breaks up the request and submits separate request to service providers specializing in video conferencing and simulation. These

service providers satisfy the request and allocate resources, possibly using yet another layer of providers. The original provider coordinates the various players both at startup and at runtime. Runtime actions might, for example, involve reducing the quality of the video if the network load increases dramatically. Alternative, at particular points in the simulation it may be appropriate to increase the resolution of the simulation output at the expensive of the interactive video.

The nature of the service providers will differ at different levels in the service hierarchy. Low-level service providers will focus on managing resources, e.g. communication resources, storage resources including both archived data and storage for temporary data, and computing resources, which can be general-purpose (e.g. a workstation cluster) or special-purpose (e.g. a real-time MPEG encoder). Higher up in the hierarchy we will find increasingly more sophisticated providers that add value through the use of proprietary code and resource management mechanisms. Hierarchical services require support for hierarchical resource management. While there has been some work in this area, e.g. research on network link-sharing [14], research is needed in mechanisms that support a wide range of sharing policies and large, dynamic hierarchies.

2.4 Non-electronic services

The electronic services market we envision has much in common with "real-world," non-electronic services markets. In both cases, a wide diversity of services exists and many options are available for integrating and marketing services. For example, in the regular market, a homeowner who wants to add a room to his house may do it herself after purchasing materials and tools (i.e. use her own resources combined with low-level services), or could instead work with a general contractor (i.e. invoking a high-level service) who takes care of everything. We envision that users will have the same options in an electronic services market. A second example is that in both cases we expect the structure of the market to be hierarchical: contractors will often rely on subcontractors to perform some of the tasks. Finally, we expect that in both markets, many services will be offered as an integrated package and most details of the implementation will be hidden from users. For example, when ordering financial data (bricks), the user will in general not care where they were produced or how they were transported. She might care who was responsible (i.e. what company generated the data or produced the bricks), or to what specification the product is guaranteed to conform, since that has a dramatic impact on the quality of the final product.

3 Application/Service Customizable Resource Management

The goal of supporting a diverse collection of interdependent services imposes new requirements on network resource management. In this section, we identify three concepts that are important for resource management in networks with rich sets of value-added services.

3.1 Resource Management Dimensions

The resource management mechanisms needed for customizable value-added services have to meet requirements along three dimensions: resource management across space, time, and organizational boundaries:

- *space*: Services and applications have to allocate resources (links, switch capacity, compute resources, etc.) to meet their needs. Mechanisms are needed to identify and allocate these resources.
- *time*: Conditions in the network and user requirements change over time. Services and applications must be able to change the use and allocation of resources in response to these changes.
- *organizations*: Many organizations will share resources in the network. Mechanisms are needed to isolate or dynamically share resources in a controlled fashion across these organizations.

The need for customization cuts across these three dimensions: services and applications need to customize what resources they allocate, how they change (adapt) resource use over time, and how they share resources with other organizations. We call a network that supports such a high degree of customizability an *application-aware network*.

3.2 Concepts

To capture the requirements for customizable resource management for application-aware networks, we introduce the following three concepts:

Virtual network: The resources allocated to an application or service are grouped into a virtual network (or mesh) that forms the basis for resource management. Service providers will for example use the resources in their mesh to satisfy service requests. The mesh includes not only resources, but also appropriate control state in the switch, computing, and storage nodes that are part of the mesh. This state can be manipulated at runtime to customize or speed up tasks in a wide range of areas such as security and adaptation.

Hierarchical scheduling: Use of each resource is managed by a hierarchical scheduler that enforces the sharing relationships between organizations. These relationships are represented by a tree with nodes representing the resource slices that are allocated to providers and arcs representing contractor-subcontractor relationships. Hierarchical resource management should support a broad range of resource sharing policies between siblings, and resource management decisions in one subtree should not adversely affect other subtrees, in order to allow resource management to be done locally by individual providers or users.

Application/service specific runtime resource management: Resource management decisions are made continuously in response to changes in network conditions and application requirements, and these resource management adjustments will also have to be tailored to the particular service being delivered. A direct way of achieving this is to have application or service specific state and code inside the network that participates directly in resource management. This approach allows a fast response to changing conditions since decisions are made locally with easy access to relevant information on network conditions.

3.3 Virtual Networks

The virtual network, sometimes also called a virtual mesh or supranet [13], is the core abstraction for resource management in application-aware networks. Virtual networks can belong to service providers or applications, i.e. service or applications meshes, and since we are interested in value-added services, a virtual network will in general include not only communication, but also computation and storage resources.

The three important operations on virtual networks are creation, support, and management. Virtual networks can be created in a number of ways. Large services meshes will typically be controlled by the owners of physical resources, or they will be created as a result of long-term agreements (contracts) between providers. Smaller meshes e.g. meshes created to support specific applications, will be created on the fly and may be fairly short-lived. At runtime, the virtual network is supported by the (hierarchical) scheduler: it makes sure that sufficient resources are used to support activities inside the virtual network. Finally, the network management system and service-specific code support the runtime management of resources inside the virtual network.

When we combine the notion of a virtual network with that of a service hierarchy, we get a hierarchy of virtual networks (Figure 2). At lower levels in the hierarchy, we find large service providers, whose primary

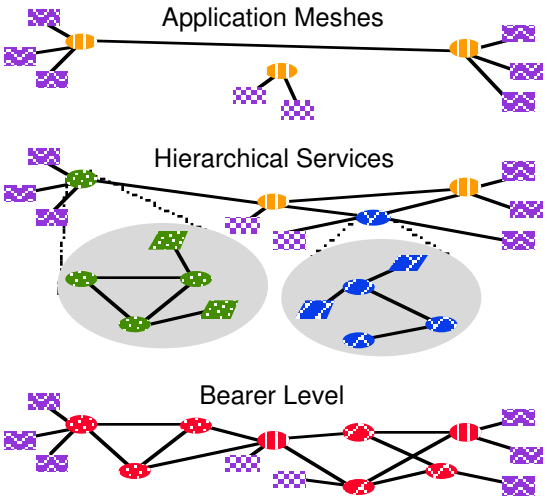

Figure 2: A hierarchy of virtual meshes

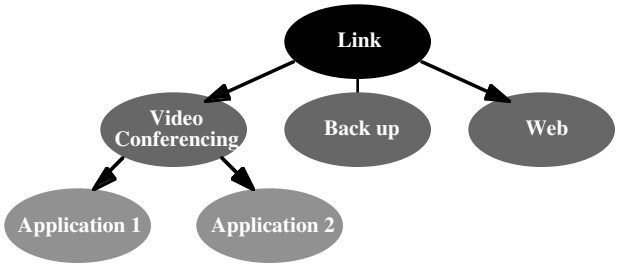

Figure 3: Resource tree for link

role is leasing resources to providers higher up in the hierarchy; these providers provide the core network infrastructure. Service providers higher in the hierarchy typically lease resources from several providers beneath them in the hierarchy, and add value by adding functionality. At the top of the hierarchy are the virtual networks that correspond to "integrator" service providers and individual applications that invoked them. Drawing a hierarchy of service meshes is difficult because it has too many dimensions. However, if we focus on a single resource (e.g. a link), the hierarchy becomes a resource tree. Figure 3 shows for example the resource tree for one of the links owned by the the Communication Services provider in Figure 1; note that it is "upside down" compared to service hierarchy.

While virtual networks and resource hierarchies can precisely characterize resource sharing policies, they also raise complexity and scalability problems. For example, backbone routers may have to manage huge trees. The key to scalability is flow aggregation, which corresponds to collapsing subtrees into nodes. Our expectation is that routers on customer premises and edge routers may manage trees that include application flows, but that core routers will largely operate on flow aggregates, along the lines of virtual private networks. This is illustrated in Figure 4. The figure shows an application mesh (highlighted in black) supporting a rich set of flows between endpoints C and A. Edge routers use the full resource tree to make scheduling decisions, but in the core of the network, the mesh resources are represented by a single node.

In the remainder of this paper we focus on resource management mechanisms, or, in terms of virtual networks, on mechanisms for the creation, support, and management of virtual networks.

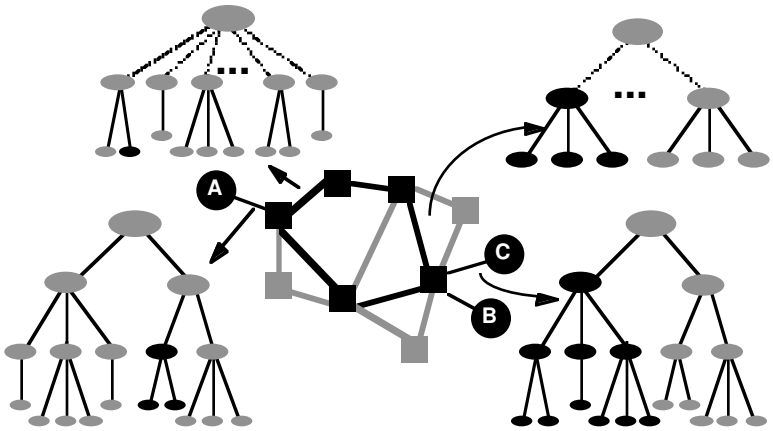

Figure 4: Flow aggregation inside a virtual network

3.4 Virtual network creation

The creation of the virtual network provides the first opportunity for resource management customization. We focus on application meshes since they are created most frequently.

When an application starts up, it will issue a request to the network listing its requirements. A request will typically include the end-points that are part of the application, the need for access to specific resources in the network, plus a description of desired communication, computation and storage functionality. The latter should be specified at as a high a level as possible, giving the network maximal flexibility in how it satisfies the request. For example, if possible, the request specifies the quality of the video stream instead of a specific bandwidth, giving the network the option of selecting the most appropriate encoding. A *service broker* will use the request to identify the required resources, using a cost or performance criterion to optimize network and/or application performance. Service brokers will often have some knowledge of the application domain so they can perform domain specific optimizations, e.g. select the best video encoding [9]. Applications may use their own service brokers to identify necessary resources. Research issues include defining languages to specify application requests, resources, and the implementation of functions in terms of resources, plus algorithms to map requests on resources. The key challenge is dealing with scale, both in terms of the size of requests and the number of options in available resources.

Once resources have been identified, a signaling protocol is used to allocate the resources, in effect creating the virtual mesh. Note that in a hierarchical resource management environment, a signaling protocol will often request resources from another service provider, indirectly invoking another signaling protocol. For example, it might end up invoking PNNI to set up an ATM connection or RSVP to set up resources inside an IP subnet. One of the key new features that must be supported by the Darwin signaling protocol is support for application delegates, both to set up the delegates on specific switch nodes, and to set up communication channels so delegates can interact. Another unusual feature is the need to set up non-communication resources.

The bottom of Figure 1 shows a mesh that could support the collaborative work environment application described in Section 2. The mesh consists of a virtual "slice" of each of the physical resources shown in black. The compute resources could either be special purpose devices or, more likely, generic resources that have been programmed to provide the specific service needed for this application.

An important criterion for the selection of the mesh is that there be a very high probability that sufficient mesh resources will remain available for the duration of the session. This is most easily achieved by reserving the resources, but this is an expensive solution, especially for bursty applications. It is more

attractive to have users share resources dynamically. Predictability can often still be achieved by carefully selecting a best effort or weak guaranteed service. For example, suppose a service provider offers a high and low quality video conferencing service, and the network infrastructure provides three service classes: two classes of differentiated best effort service and an IETF style guaranteed service. The provider could use the following policy: during peak hours, the high and low quality video streams would be mapped to the guaranteed and high quality best effort service classes, respectively, while the high and low quality best effort service classes might be sufficient during off-peak hours. Clearly, more refined strategies are possible: the selection can be more dynamic based on real-time information, and different classes can be picked for different flows and in different subnets. The provider can make such optimizations transparently to the user.

Mesh allocation involves not only identifying and allocating the resources needed to satisfy the application request, but also preparing the network to deal efficiently with a wide variety of possible runtime situations. For example, by establishing the appropriate state in the mesh, e.g. the delegates described below, the response to changes in traffic conditions could be implemented as local actions, e.g. selective packet dropping and the redistribution of bandwidth. The mesh creation phase can also be used to plan for application changes. For examples, when a mesh is created for a video conference, there may be advanced knowledge about the likely number and geographical distribution of participants. This knowledge could be used to preallocate resources in some regions, and to establish service delegates that can quickly redistribute these resources as participants enter and leave the conference. Finally, the mesh can also help in dealing with recovery after failures: the state associated with mesh can be used to redirect traffic around a failed link, or reestablish state in a failed switch node after it comes back up.

3.5 Hierarchical resource management

An important aspect of our vision is that services can be composed in a hierarchical fashion: applications invoke high-level service providers, which may in turn invoke services from more primitive or lower-level service providers. As described above, the set of resources controlled by each service provider can be seen as a virtual network. The network is virtual in the sense that the service provider does not control specific physical resources, but rather shares physical resources with other service providers. Consider the example shown in Figure 5. There are two bitway service providers (B1 and B2) that own physical link and router resources. In addition, there are two value added service providers (V1 and V2), each of which builds a virtual mesh that consists of link and router capacities of B1 and B2, and additional processing/storage nodes owned by itself.

Each service provider in the hierarchy supports multiple clients and may need to dedicate a fraction of its resources to each of its clients. Clients are either higher level service providers or applications, and they may in turn want to control resource sharing between their clients (for service providers) or across their flows (for applications). In other words, the service hierarchy results in a resource management hierarchy. For each physical resource, sharing and thus contention exist at multiple levels: at the physical resource level among multiple service providers, at the service provider level among lower level service providers or organizations, and at application level among individual flows. This hierarchical resource sharing relationship for a particular resource can be concisely represented by a resource tree. Figure 5 shows an example of a resource hierarchy for a specific link. The root node represents the physical link resource and leaf nodes represent the finest granularity of virtual resources, which can correspond either to individual traffic streams, such as the Seminar Video stream, or to traffic aggregates, such as Provider 2's best-effort traffic class. Interior nodes represent virtual resources that are managed by entities such as service providers, organizations and applications.

We now use this example to illustrate several important features of a resource hierarchy. First, the resource management policies of the entities sharing the physical resource can be very diverse. In the example, two service providers dynamically share the 155 Mbps link. Service Provider 2 supports the IETF Intserv

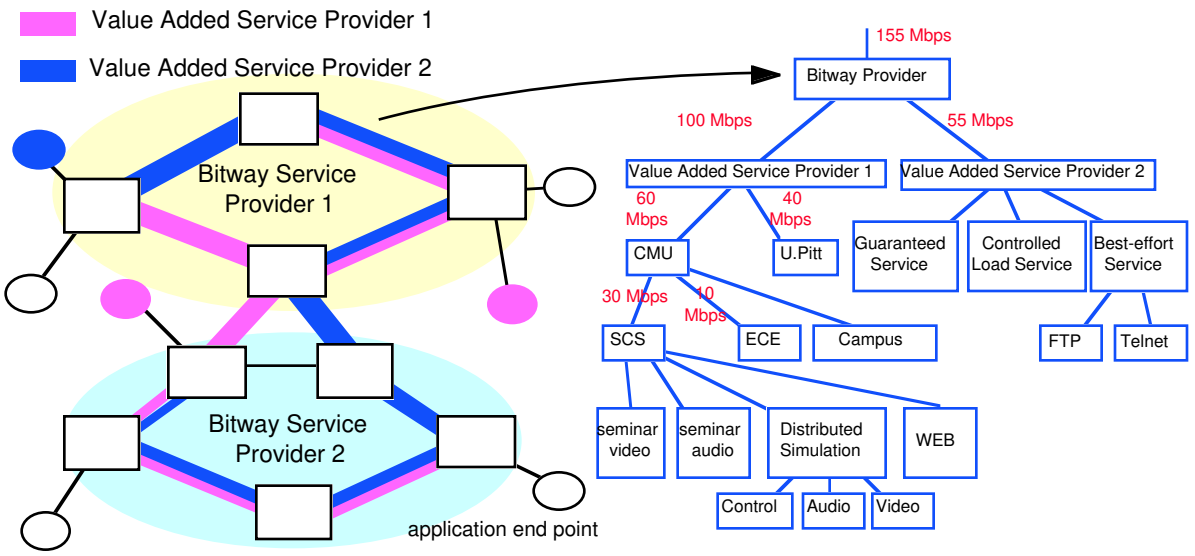

Figure 5: An Example Resource Management Hierarchy

QoS model (guaranteed and controlled load) for individual traffic streams, while the more sophisticated Service Provider 1 supports organization-based QoS, where the organizations and applications can specify the dynamic sharing relationship for their traffic streams. It is important that these diverse policies can coexist without negative interactions. For example, changes in how CMU distributes its bandwidth should not result in missed deadlines for the flows in the guaranteed service class of Provider 2.

Second, the hierarchy specifies both a minimum share of the resources for each entity plus sharing policies that defines how unused resources should be distributed over other entities. The minimum share of resources given to an entity can be viewed as a virtual resource that can be further allocated according to a policy specified by that entity. For example, Value Added Service Provider 1 further divides its virtual link resource to two organizations: CMU and the University of Pittsburgh. An entity will not only receive its minimum share of resource, but also get excess resources that are allocated to other entities but not currently used. This *dynamic* (rather than *static*) sharing relationship is also governed by the resource sharing hierarchy. For example, if the aggregate bandwidth of controlled load traffic served by VP2 is less than 10 Mbps during a certain period, the best-effort traffic served by VP2 will have the highest priority to use the extra resource.

Finally, the hierarchy allows the co-existence of competitive and cooperative sharing. For example, the Distributed Simulation application may have many audio streams traversing the same link. However, it is unlikely that more than a few audio streams are active simultaneously. Rather than making a separate reservation for each audio stream, the Distributed Simulation application makes one reservation and lets all its audio streams *cooperatively* share this virtual resource. Similarly, rather than making a reservation for each traffic stream, Provider 2 lets all controlled load traffic freely share the same virtual resource, exploiting statistical multiplexing. In contrast, since Providers 1 and 2 are *competitively* sharing the link, each provider is guaranteed to get its share (100 Mbs and 55 Mbs respectively) of the link if it needs it to support its traffic, independent from the requirements of, or the actions by, the other provider.

While this hierarchical resource management framework is also useful for today's networks, for example, to support hierarchical link sharing service [14], it is much more important in the context of value-added services networks in which resource owners, service providers, and applications will all have their own

entity-specific resource management policies. The ability to customize resource management policies at all sharing levels is one of the key requirements and distinctive features for value-added services networks.

In contrast, today's networks provide limited support for customizable management of network resources. The network allocates resources either on a per packet basis (datagram network) or on a per traffic stream basis (Virtual Circuit Network). RSVP introduces the concept of reservation style, with which traffic streams from different senders can share one reservation. However, this capability is limited in several respects. First, resources can be shared only by traffic streams directed to the same multicast address. Even if a single video conferencing application uses video, audio, and whiteboard tools, there can be no dynamic resource sharing by these tools if they use different multicast addresses, as is case for the MBONE tools. The current model of RSVP and IntServ also provides no support for hierarchical virtualization of resources.

3.6 Application-specific adaptation

Achieving high quality of service is not a goal that can be met at startup time alone; it requires attention on an ongoing basis. The reason is that both network conditions and application requirements change, and thus the "optimal" allocation of resources changes constantly. Runtime resource management should be customizable for the same reasons that virtual network creation is customizable: the best way to adapt to changes in resource availability is service specific, and service providers will want to consider user preferences when optimizing service quality. For example, if the bandwidth available to a video stream drops, the service provider may simply reduce the frame rate for "basic" customers, or may switch to a higher differentiated service class for "deluxe" customers. Moreover, runtime resource management decisions have to be made on several time scales, including very fine time scales. The faster a provider can respond, the less noticeable and the less objectionable the impact is likely to be for users.

A direct way of achieving responsive, customizable runtime resource management is to have *application and service presence* in the network that is directly involved in or affects resource management decisions. Application or service presence can take many forms, ranging from a set of parameters that encode application priorities, to general programs that are invoked when certain application-defined events are triggered. The most appropriate format will have to balance the degree of customization and sophistication with the complexity and overhead of the required runtime environment. We will refer to such state or code as *delegates* since they represent the interests of the application or service provider in the network. Having delegates inside the network (as opposed to the endpoints) should improve responsiveness, since the resource management code has ready access to relevant information on network conditions and can quickly implement the decision, avoiding delays of potentially several roundtrip times. Delegates are a focused application of active networking [30].

Delegates are logically part of the virtual network and they are established when the virtual network is created. This involves checking the safety properties of the delegates, allocating local resources, and arranging for appropriate access to the switch state and network events. The set up phase also has a security component that determines 1) what runtime resource management actions delegates are allowed to take, and 2) what pool of resources they are allowed to manage. Delegates will typically be allowed to manage the resources of the virtual network they are associated with; if hierarchical scheduling is used, this means that they are allowed to modify a subtree of the local resource tree. One of the reasons for having such an extensive set up phase is that it reduces the need for runtime checking, resulting in more efficient customized runtime resource management.

Both the usefulness and implementation cost of delegates depend critically on the type of resource management decisions they can make. Ideal candidates are operations that benefit from customization and fast response, but that do not require resource intensive processing. The simplest class of operations is to change the resource tree used by the scheduler in response to an increase or decrease in bandwidth, e.g. the delegate can change the distribution of bandwidth across flows or flow aggregates, or it can change the

service class that used for a set of flows. Customization is needed to implement service-specific cost or quality goals, e.g. a minimum frame rate has to be maintained, or to implement organizational policies, e.g. certain classes of video get dropped if there is congestion. Other possible operations include changing the routing inside the virtual network to balance traffic load or selectively dropping packets to relieve congestion. Finally, delegates can be used to coordinate distributed events. For example, delegates could provide explicit feedback to applications if some application-specified condition is met, thus allowing the application to respond more quickly and based on more accurate information than if it had to rely exclusively on implicit feedback (packet drops).

While delegates may make it possible to do more effective resource management, this comes at a cost: delegates can potentially have significant computational requirements, and their execution could in fact reduce overall network performance if they are not supported correctly. Routers support activities on a range of time scales, and complexity and performance can be controlled by choosing the appropriate time scale for each activity. The most frequent operation is packet forwarding, and performance can be optimized by keeping the data forwarding path simple. Instead, we expect application or service specific delegates to execute in the control plane of the switch nodes, i.e. they operate on a coarser time scale. While there will of course still be stringent limits on the resource use of delegates, this should be easier to manage. Note that it is likely that people will develop delegates that are computationally expensive or executed frequently, or even delegates that perform data processing tasks such as video transcoding or encryption. However, these tasks would generally require different locus of execution, e.g. they could be executed on a workstation farm or on compute nodes inside the router that are specifically designed for this purpose.

While application presence in the network and explicit interactions may allow more effective adaptation, there are also several risks associated with these mechanisms. These risks include safety and security concerns caused by application presence, reduced network performance, and, maybe most importantly, a potentially significant increase in complexity inside the network, which may cause increased failures and longer recovery times. Dealing with these problems is a major challenge, and the benefits of these mechanisms ultimately must be weighed against their risks and costs.

4 The CMU Darwin Project

The CMU Darwin project has implemented a prototype system that incorporates the resource management mechanisms described in the previous section. We give an overview of the Darwin architecture and briefly describe its components. More details can be found elsewhere [8].

4.1 System Architecture

Figure 6 shows how the components in the Darwin system work together to manage network resources. Applications running on end-points can submit requests for service to a service provider which includes a service broker (Xena). The service broker identifies the resources needed to satisfy the request, and passes this information to a signaling protocol, Beagle, which allocates the resources. For each resource, Beagle interacts with a local resource manager to acquire and set up the resource. The local resource manager modifies local state, such as that of the packet classifier and scheduler shown in the figure, so that the new application will receive the appropriate level of service. The signaling protocol can also set up *delegates*, code segments that can customize resource management for newly allocated resources. Throughout this process, appropriate resource authorizations must be made. Service brokers have to know what resource pools they are allowed to use, and the signaling protocol and local resource managers must be able to validate the resource allocation request and set up appropriate billing or charging. Other scenarios, using the same architecture, are possible. For example, sophisticated applications can use a built-in service broker

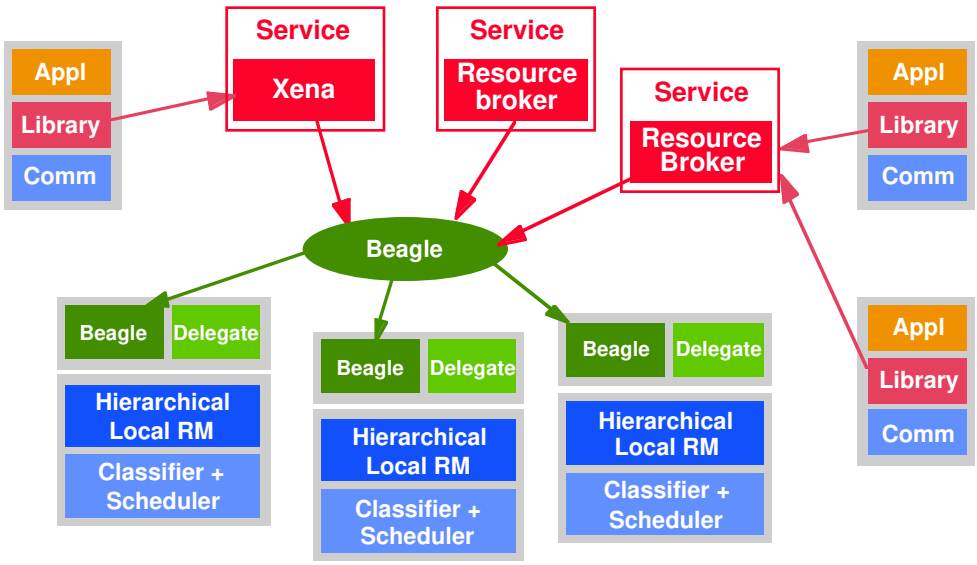

Figure 6: Darwin system architecture

to identify resources and can then interact directly with Beagle to allocate the resources.

This architecture is driven in part by our goal of supporting customizable services in a heterogeneous network environment, and by a need to support multiple administrative domains. Service brokers interact directly with applications and can incorporate a lot of knowledge about the application domain. One example of a service broker could be a service provider specializing in video conferencing: it can translate a high level request (high-quality conference for 5 participants) into a list of required resources. A radically different example of a service broker could be a network resource manager for a university; the manager understands the relative priorities of requests and also enforces a set of policies that determine when additional resources can be leased. Local resource managers, on the other hand, concern themselves only with the management of a specific communication, computation or storage resource. We expect that different types of local resource managers will be deployed in different administrative domains. The signaling protocol has to bridge the gap: it receives high-level, application-independent and implementation-independent instructions from the service brokers, and translates them into specific instructions for each local resource manager.

Figure 7 shows how the different entities on switch nodes interact. For performance reasons, it is important to distinguish between the data forwarding and control functions on the switch nodes. The focus of the data path, which includes classification, route lookup, and scheduling, is on high packet throughput, so this component must be kept simple. Control plane activities include signaling, routing (not shown), and customized runtime management actions. These activities happen on a much coarser time scale, and, although resources will always be limited, there is more room for customization and intelligent decision making. The control interface, i.e. the interface exported by the local resource manager to delegates and Beagle, specifies the range of resource management operations that delegates and Beagle can perform. As such, it defines the scope of customization of resource management that is possible at startup (through Beagle) and at runtime (through the delegates).

4.2 System components

The Darwin architecture is similar in many ways to traditional resource management structures. For example, the resource management mechanisms for the Internet defined by the IETF in the last few years rely on

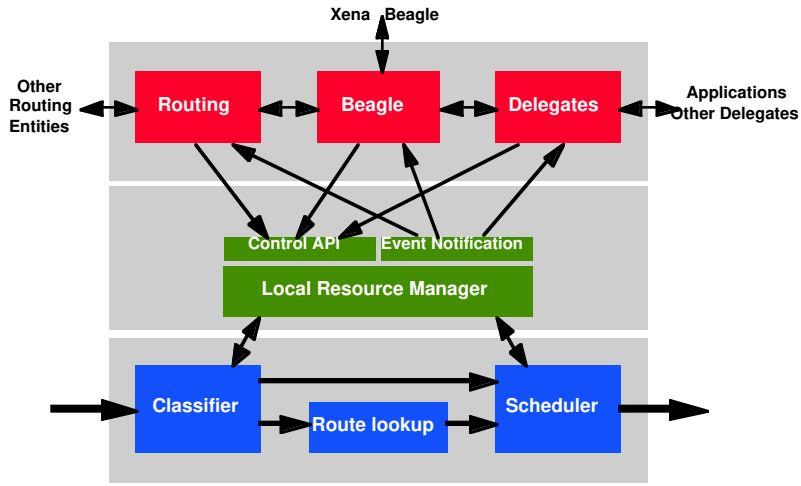

Figure 7: Darwin node architecture

QOS routing [32, 12] (service brokers), RSVP [36] (signaling similar to Beagle), and local resource managers that set up packet classifiers and schedulers. The more recent proposals for differentiated service [25] require similar entities. The systems differ in the specific responsibilities of the entities. We briefly elaborate on the Darwin system components and explain how they address the requirements listed in Section 2.

The first component is the hierarchical fair service curve (HFSC) scheduler [28] that manages link bandwidth and that can implement a broad range of sharing policies, including fair sharing and guaranteed services. Moreover, it supports the hierarchical resource management that is needed for hierarchical service deployment, and it allows controlled sharing of resources between sibling nodes without violating guarantees inside the subtrees, so subtrees can be managed independently.

A second component is a service broker, called Xena, which translates an application request into a set of resources and lower-level service invocations. Xena is essentially a multi-resource router: it tries to satisfy the request while minimizing a cost function or maximizing an application-provided quality function. The latter allows applications to customize the quality of the service. The current implementation of Xena recognizes not only generic resource types (e.g. bandwidth, CPU cycles), but also specialized resources related to video streams, e.g. different encodings and transcoder requirements, allowing Xena to make domain-specific optimizations for that class of applications [9]. For example, it can transparently insert transcoders to deal with mismatches in encoding. These optimizations are possible because brokers receive high-level requests and can use domain knowledge. Xena currently uses an integer linear programming formulation [4] for its optimizations.

The Beagle signaling protocol is responsible for carrying out the allocation of the chosen resources and for maintaining the state associated with virtual networks [10]. The distinguishing feature of Beagle is that it allocates resources for complete virtual networks, instead of individual flows. Other properties include support for hierarchical resource management, the installation of application and service delegates, and the provision of the communication channels they need. Work is in progress on providing appropriate authorization and security mechanisms.

Delegates are implemented as Java applets that execute in the Kaffe [33] runtime environment, which runs on the router. This environment provides maximal flexibility for experimentation, albeit at some performance penalty. The current control interface allows delegates to modify the resource tree associated with the virtual networks they manage, e.g. they can change bandwidth distributions, define new flows, or specify

that a particular flow should be dropped. Since the packet classifier uses not only the traditional IP header fields, but also an application flow id, this allows delegates to implement a broad range of customized resource management policies. For example, if congestion builds up, a delegate can specify that specific flows should get guaranteed bandwidth, or it can selectively drop low priority flows. Much more work is needed to define the full control interface and to evaluate benefits and costs of the delegate model.

All four components have been implemented and execute in an integrated fashion [8]. The initial development and test environment was a controlled testbed consisting of three routers along with twelve endpoints that can execute traffic generators and applications. The routers are PCs running NetBSD or FreeBSD. Darwin has recently been ported to the DARPA Cairn wide-area network testbed and a port to a vendor platform is planned.

5 Impact on Core Network Infrastructure

In this section we look at the changes required in the network infrastructure to support application-aware networks and describe some possible deployment scenarios.

While the mechanisms described above are quite different from these in use today, we believe that introducing them is quite feasible, since most of the required changes are outside of the core network infrastructure. As we indicated above, it is useful to categorize the activities according to the time scale on which they occur. On the finest time scale, we have hierarchical scheduling. It has be built into the "core network" to provide the bearer services, on which higher level services can be built in a hierarchical way. This is however the main change, and in fact, simpler measures (such as provisioning of CBR circuits at some switches or use of a differentiated services infrastructure [5]) can support high quality service. The other mechanisms are on coarser time scales and will be easier to introduce.

Given the cost of the installed networking infrastructure, new features are always introduced incrementally. Application-awareness can be introduced incrementally by gradually introducing application-aware switch nodes. The nodes would be connected using the existing networks, which represent the "bearer" service layer of Figure 2. While having a large number of application-aware switch nodes would increase flexibility, we expect that most applications will require only a small set of such nodes, and most switches and routers that handle application traffic will provide only basic bearer services. The basic bearer services should provide some mechanism to provide different level of service to different users, but this can be at a coarse level, e.g. for flow aggregates. For example, edge routers could use the mechanisms described in this paper to implement customized definitions of QoS while core routers would only operate on flow aggregates, as described in Section 3.3.

Our motivation based on an electronic services market leads us to suggest that these or similar mechanisms can be used to support such a market on the Internet. Even at a smaller scale, dynamic availability of efficient, assured services and service creation through composition will be useful, and this environment may be more appropriate for initial testing and deployment. An example would be a university campus or a commercial or government organization that has multiple sites connected by leased lines. The resource management mechanisms described above could be used to manage traffic belonging to different departments or classes of applications (e.g. with different priorities or security requirements). Similarly, resources on the network could be packaged into value-added services accessible to end-points.

6 Relationship to Other Research

There has recently been a lot of work in defining and implementing support for multiple classes of QoS in networks. This has resulted in specific proposals both for ATM [1] and for IP [11]. The "Integrated Services" IETF proposal, for example, defines a controlled load (weak guarantees) [34] and guaranteed

traffic class (strong guarantees) [26] in addition to the traditional best effort class. A more recent proposal for differentiated services [5] places more emphasis on flow aggregation. The Darwin project is looking at a higher level notion of QoS than these efforts: it looks at collections of streams instead of individual streams, and tries to define QoS in an application-oriented way rather than having a small set of predefined classes. Note that these two research thrusts complement each other: Darwin expects a bearer layer, which is likely to support a small number of simple bearer services.

Another related research effort in the IP community is link sharing [14]. It addresses the problem of how organizations can share network resources in a preset way, while allowing the flexibility of distributing unused bandwidth to other users. Clearly, the hierarchical scheduling addresses in part the link sharing problem [3, 28].

Another active area of research is QoS routing. Given an application request for a connection with a certain QoS properties, QoS routing algorithms try to identify a path that meets the requirements while also optimizing some cost function. QoS routing is basically a simple version of the virtual mesh allocation problem in Darwin. Note that QoS routing is in general NP-complete [32], although it has been shown to be tractable for certain types of networks [24, 23]. The mesh allocation problem is significantly more complex, so we expect to make use of heuristics that have good properties for the common case.

Signaling protocols have been widely studied and deployed. These protocols typically allow the user to request one of a small number of service types, e.g. establishing a point-to-point connection or adding a receiver to a multicast connection. A recent example is the ATM signaling protocol standardized by the ITU and ATM forum [2]. RSVP [36] is an example of a signaling protocol that came out of the data networking community. It establishes multicast "flows", which are weaker than connections, and allows the creation of filters that merge the reservations of multiple flows.

Other efforts have defined and implemented signaling frameworks that give end-points access to the internals of the network [21]. Examples of primitive operations are updating a routing table or reserving link resources. These can be combined to set up a connection, i.e. signaling frameworks could be viewed as an assembly language that can be used to implement higher level constructs. Similar to assembly languages, they provide maximum flexibility, but little guidance or support for program development.

The signaling protocol in Darwin will strike a balance between these two extremes. The protocol will be more structured than a framework, but will have to be flexible enough to support application delegates and their requirements. Support for application-awareness and security are two areas of emphasis for the Darwin signaling protocol.

The idea of "Active networks" has recently attracted a lot of attention. The idea is that instead of having packets be passive entities that are carried around, packets can be active and change the behavior of the network [30]. The term "active networks" refers to a broad range of networks. One extreme is the idea that every packet header is a program that will be executed by the switches. In a weaker form, programs are restricted to special packets and active networks provide a mechanism to manage the network more easily, e.g. active packets can be used to upgrade software over the network.

The Darwin project touches on the active networking area in two ways. First, application delegates are an example of active packets, but represent a very "mild case" of active networks: delegates operate in the control plane, and their actions are restricted to the management of traffic and resources supporting communication. A second area where Darwin overlaps with active networks is the integration of computation and communication inside the network. Since active packets contain programs, they will have certain computational requirements. When allocating a virtual mesh, Darwin will try to satisfy both computational and communication requirements, and it may indeed make tradeoffs across these different resource types. Virtual mesh allocation addresses one of the resource allocation problems associated with active networks.

A final area of related work is adaptation to network conditions. In the Internet, there is no exchange of congestion information between entities, so end-points have to adapt based on implicit feedback, e.g. TCP interprets dropped packets as a sign of congestion [19]. Recently, several applications have been developed

that adapt to the bandwidth and latency variations of the underlying network. One class of examples are Internet-based video and audio tools [17, 18, 15, 6, 31]. Another class of examples consists of distributed computations modifying the granularity of the computation in response to network status [27, 29]. The adaptation policies used in these applications are ad hoc and application specific.

Interpreting implicit feedback from the network is difficult [7], and some recent network control algorithms, such as ATM flow control algorithms [1, 20], provide explicit information to each end-system. There have been some attempts at making this information available to applications, resulting in explicit, albeit generic, feedback to applications. Recent work on renegotiated service [22, 16, 35] attempts to combine reservation and feedback control algorithms, resulting in the explicit exchange of state information in both directions between applications and the network. In Darwin we plan to design mechanisms that allow systematic explicit state exchange among applications, transport layer, and network layer, giving applications access to more accurate and timely information about the network. Darwin focuses on multi-party applications, where the interpretation of and response to feedback is more complicated. Moreover, we hope that application presence will make it possible to generate feedback that is tailored to the application, and easier to interpret.

7 Conclusion

We envision a market place of electronic services that allows the construction of value-added services by composition and rapid marshaling of resources provided by other services. We argue that flexible resource management mechanisms that are customizable by applications and service providers are important for such a electronic services market place. In particular, we present three concepts that capture the resource management requirements:

- a virtual network consisting of all the resources and states that are provisioned to a service provider or an application; the creation of the virtual network provides an opportunity to optimize resource use and the quality of the delivered service, using application and service specific optimization criteria.
- a hierarchical scheduler allows each physical resource to be managed as a set of virtual resources, controlled by different entities; the scheduler has to be able to satisfy the customized resource management policies of different entities (resource owner, service providers, applications) simultaneously,
- runtime resource management is customized through the use of delegates, application or service specific code or state in the network; delegates allow resource allocation decisions to be made locally, without the need for slower global protocols.

These concepts have been implemented in the CMU Darwin system. Within Darwin, virtual networks are created using Xena, a a service broker that designs an efficient mesh in response to application requests, and Beagle, a signaling system that maintains the low-level state required to implement and maintain the mesh. The HFSC scheduler implements the virtual network abstraction and manages hierarchical resource sharing relationships between the virtual networks. Finally, virtual network resources are managed by Java-based delegates that operate in the control plane and manage data plane resource through a control interface.

References

[1] ATM Forum Traffic Management Specification Version 4.0, October 1995. ATM Forum/95-0013R8.

[2] ATM User-Network Interface Specification. Version 4.0, 1996. ATM Forum document.

[3] J.C.R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, October 1997. Also in SIGCOMM'96.

[4] Michel Berkelaar. lp_solve: a Mixed Integer Linear Program solver. ftp://ftp.es.ele.tue.nl/pub/lp_solve/, September 1997.

[5] An Architecture for Differentiated Services, April 1998. IETF Internet draft draft-ietf-diffserv-arch-02.txt, Work in progress.

[6] Jean-Chrysostome Bolot and Andres Vega-Garcia. Control mechanisms for packet audio in the internet. In *IEEE INFOCOM'96*, volume 1, pages 232–239, San Francisco, CA, March 1996. IEEE.

[7] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the SIGCOMM '94 Symposium on Communications Architectures and Protocols*, pages 24–35, University College, London, London, UK, October 1994. ACM.

[8] Prashant Chandra, Allan Fisher, Corey Kosak, T.S. Eugene Ng, Peter Steenkiste, Eduardo Takahashi, and Hui Zhang. Darwin: Resource Management for Value-Added Customizable Network Services. In *Sixth International Conference on Network Protocols*, Austin, October 1998. IEEE.

[9] Prashant Chandra, Allan Fisher, Corey Kosak, and Peter Steenkiste. Network Support for Application-Oriented Quality of Service. In *Proceedings Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 187–195, Napa, May 1998. IEEE.

[10] Prashant Chandra, Allan Fisher, and Peter Steenkiste. Beagle: A resource allocation protocol for an application-aware internet. Technical Report CMU-CS-98-150, Carnegie Mellon University, August 1998.

[11] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of ACM SIGCOMM'92*, pages 14–26, Baltimore, Maryland, August 1992.

[12] Eric Crawley, Raj Nair, Bala Rajagopalan, and Hal Sandick. A Framework for QoS-based Routing in the Internet, August 1998. IETF RFC 2386.

[13] L. Delgrossi and D. Ferrari. A virtual network service for integrated-services internetworks. In *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 307–311, St. Louis, May 1997.

[14] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[15] R. Frederick. Network video (nv), 1993. Software available via ftp://ftp.parc.xerox.com/net-research.

[16] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic. In *Proceedings of SIGCOMM'95*, pages 219–230, Boston, MA, September 1995.

[17] V. Jacobson and S. McCanne. Visual audio tool (vat), 1993. Software available via ftp://ftp.ee.lbl.gov/conferencing/vat.

[18] V. Jacobson and S. McCanne. Vic, 1995. Software available via ftp://ftp.ee.lbl.gov/conferencing/vic.

[19] Van Jacobson. Congestion Avoidance and Control. In *Proceedings of the SIGCOMM '88 Symposium on Communications Architectures and Protocols*, pages 314–329. ACM, August 1988.

[20] H.T. Kung and Robert Morris. Credit-based flow control for ATM networks. *IEEE Network Magazine*, 9(2):40–48, March/April 1995.

[21] A. Lazar, Koon-Seng Lim, and F. Marconcini. Realizing a foundation for programmability of atm networks with the binding architecture. *IEEE Journal on Selected Areas in Communication*, 14(7):1214–1227, September 1996.

[22] Kam Lee. *Hint-Assisted Traffic Control in Heterogeneous Networks*. PhD thesis, Department of Computer and Electrical Engineering, Carnegie Mellon University, October 1997.

[23] Qingming Ma and Peter Steenkiste. On path selection for traffic with bandwidth guarantees. In *Fifth IEEE International Conference on Network Protocols*, pages 191–202, Atlanta, October 1997. IEEE.

[24] Qingming Ma and Peter Steenkiste. Quality of service routing for traffic with performance guarantees. In *IFIP International Workshop on Quality of Service*, pages 115–126, New York, May 1997. IFIP.

[25] K. Nichols, L. Zhang, and V. Jacobson. A Two-bit Differentiated Services Architecture for the Internet, November 1997. Internet draft, draft-nichols-diff-svc-arch-00.txt, Work in progress.

[26] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service, September 1997. IETF RFC 2212.

[27] Bruce Siegell and Peter Steenkiste. Automatic selection of load balancing parameters using compile-time and run-time information. *Concurrency - Practice and Experience*, 9(3):275–317, 1996.

[28] Ion Stoica, Hui Zhang, and T. S. Eugene Ng. A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Service. In *Proceedings of the SIGCOMM '97 Symposium on Communications Architectures and Protocols*, pages 249–262, Cannes, September 1997. ACM.

[29] Hongsuda Tangmunarunkit and Peter Steenkiste. Network-aware distributed computing: A case study. In *Second Workshop on Runtime Systems for Parallel Programming (RTSPP)*, page Proceedings to be published by Springer, Orlando, March 1998. IEEE. Held in conjunction with IPPS '98.

[30] David Tennenhouse and David Wetherall. Towards and active network architecture. *Computer Communication Review*, 26(2):5–18, April 1996.

[31] Hideyuki Tokuda, Yoshito Tobe, Stephen Chou, and Jose Moura. Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network. In *Proceedings of the SIGCOMM '92 Symposium on Communications Architectures and Protocols*, pages 88–98, Baltimore, August 1992. ACM.

[32] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.

[33] Tim Wilkinson. KAFFE - A virtual machine to run Java code. http://www.kaffe.org/.

[34] J. Wroclawski. Specification of the Controlled-Load Network Element Service, September 1997. IETF RFC 2211.

[35] H. Zhang and E. Knightly. A New Approach to Support Delay-Sensitive VBR Video in Packet-Switched Networks. In *Proceedings of the 5th International Workshop on Network and Operating System Support For Digital Audio and Video*, pages 275–296, Durham, New Hampshire, April 1995. Also to appear in Multimedia System Journal.

[36] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Communications Magazine*, 31(9):8–18, September 1993.