Chapter 7

Related Work

The work presented in this dissertation draws on research in several different areas. The reserve model depends on theoretical results from real-time scheduling, and the design and implementation were influenced by the work in real-time system design as well as the requirements for multimedia applications. This chapter presents an overview of related work: most of the related work focuses on systems issues although a section on applications discusses work on tools and application adaptation techniques.

7.1 System Implementation

The increasing integration of computer and telecommunications technologies has focused attention on the real-time issues that arise in processing digital audio and video. Handling multimedia data streams requires an understanding of the timing requirements, encoding techniques, and data formats of the new media types [13,69,94,135,136] as well as programming interfaces [60,101] and new application design techniques [35].

7.1.1 Multimedia support

A great deal of recent work has focused on how software systems (including operating systems) can be designed to support multimedia applications. Many researchers and practitioners consider resource reservation desirable if not absolutely necessary for real-time multimedia operating systems [3,46,53,60,87,102,127]. Herrtwich [42] gives an argument for resource reservation and careful scheduling in these systems. Others prefer a best effort approach to OS design for multimedia applications [19,21,90]. Recent survey papers [116,132] and books [12,115] discuss work in this area.

One of the simplest ways to do resource allocation for multimedia applications is to dedicate an entire machine to a single multimedia application. Multimedia applications for single-user personal computers typically assume that only a single multimedia activity will exist at any one time. Or if several multimedia activities exist, the assumption is that they will be associated with a single application that can do cooperative scheduling of these activities. Adobe Premiere [95] is an example of such an application. If this assumption is violated, neither the applications nor the system will be in a position to assert anything about the behavior of multimedia applications. There is no admission control or any kind of overload protection in such systems.

Other systems such as OS/2 [60] put limits on the number of high-level activities, such as video streams being processed by the system, as a primitive form of admission control. This technique does not address the problem of resource allocation or admission control for arbitrary computations and media processing applications.

Jeffay *et al.* implemented an operating system designed for guaranteed real-time scheduling [53]. A video capture and playback application demonstrates that the analytical techniques can be successfully applied to real applications. In subsequent work, Jeffay has focused attention on transport mechanisms to detect and deal with variability in network behavior [54]. The reservation system described in this dissertation uses real-time scheduling analysis as does Jeffay. It focuses on enforcement whereas Jeffay's recent work focuses more on flexibility. The reservation system would benefit greatly from the increased flexibility of having adaptive mechanisms such as Jeffay's at higher levels.

Anderson *et al.* [3] argue for introducing more sophisticated timing and scheduling features into operating systems, and their DASH system design supports a reservation model based on linear bounded arrival processes (LBAP) [22,23]. They implemented their system design and were able to report some preliminary experiences with the system. They use earliest deadline scheduling for real-time traffic because it is optimal in the sense that if any algorithm can schedule a particular collection of tasks, the earliest deadline algorithm can do it. Admission control for this system is based on a time-line where new jobs are admitted only if they fit onto the time-line when the job request arrives [129]. The reservation system described in this dissertation uses an admission control algorithm based on a periodic scheduling framework with scheduling analysis rather than a timeline approach. One way for reserves to accommodate one-shot events would be to use a timeline based admission control policy and scheduling algorithm. The reservation system focuses more on enforcement of specified computation times.

Hyden [46] considered the problem of supporting QOS in operating systems in his thesis. He implemented a system that offers a virtual processor interface to applications. A video decoder application demonstrates how such an interface can be used by an application. In contrast, the reserve system provides flexibility in reservation binding for a more integrated view of resource usage reservation, measurement, and enforcement.

Coulson *et al.* [21] present a system design based on Chorus [103]. This system uses earliest deadline scheduling, but they do not provide any admission control and usage enforcement. QOS commitments can be revoked, and overload is permitted; commitments are degraded as a response to overload. The work focuses primarily on fast context switching and reducing protection domain crossings. The reserve system provides guaranteed resource reservation using an enforcement mechanism with QOS policy modules layered over the reservation abstraction.

Jones [56] describes some ideas on system design for multimedia applications which depend on value functions as a means of scheduling processes based on timing constraints, semantic importance, user preferences, and other information about application-level requirements. The reserve system divides the QOS provision problem into a reservation mechanism and QOS policy layers. Jones' work focuses more on the QOS policy and could benefit from mechanisms for guaranteed resource reservation.

Many multimedia cards and co-processor boards and boxes are actually embedded systems with their own processors and operating systems. These systems must manage resources for multiple real-time activities that must be multiplexed.

Hopper [44] described Pandora's Box which is a transputer system designed to be a multimedia peripheral for a traditional workstation. The Pandora system employs several transputers, each of which handles a particular function in the system such as compression, decompression, network traffic, and audio. A similar approach is being pursued in the context of the Desk Area Network [40] and related operating system efforts [9,81]. The reserve system addresses the sharing of devices and software resources by appropriate admission control policies, scheduling algorithms, and enforcement mechanisms. Multiplexing of resources results in more efficient resource utilization.

The Mwave system [47] consists of a digital signal processor (DSP) card intended for use with a PC. The Mwave card handles various audio processing and telephony tasks, depending on the host processor for control functions. A programmer can develop applications that are divided between the host system and the Mwave processor. There is a programming environment that supports application development, and the operating system running on the Mwave processor provides some real-time support for scheduling and enforcement. In particular, the Mwave/OS performs enforcement functions, resetting the card when any of the real-time activities misses a deadline. The reserve abstraction provides more functionality for flexible binding of reserves, and the reserve system also has a more flexible policy for handling timing failures. In particular, real-time activities are not affected by the timing failure of an independent real-time or non-real-time activity.

7.1.2 QOS architectures

The reserve abstraction is designed to support higher-level architectures for managing quality of service specification and negotiation. Several QOS architectures have been proposed assuming that the mechanisms for operating system resource management, such as the reserve system, would be developed.

Nicolaou [89] described a QOS architecture suitable for programming distributed multimedia applications. His work describes an architecture that one might use to design and implement multimedia applications. While some systems issues like resource management and scheduling are identified as being important, the cited work does not address those problems. A prototype implementation based on the architecture demonstrates the feasibility of this approach, but since the prototype is implemented on UNIX, its performance suffers from a lack of real-time scheduling techniques in the operating system. The reserve model is designed to provide the kind of operating system support necessary for Nicolaou's QOS architecture.

Wolf *et al.* [137] defined a QOS architecture for a communication transport system, and they have an initial implementation of their system, called the Heidelberg Transport System. The reserve system focuses more on admission control, dealing with interaction between processes (such as client/server interactions) and usage enforcement.

The QOS Broker [84] provides an architecture for handling QOS negotiation among resource "buyers" and "sellers." Protocols are provided for carrying out the negotiation, and a version of the QOS Broker was implemented and used in the context of a telerobotics application. The Broker contains hooks for reserving resources in operating systems and networks when reservation mechanisms are available. The reserve system described in this dissertation provides the mechanism necessary for negotiating guaranteed service.

7.1.3 Networking

The idea of bandwidth reservation for network quality of service models has been explored by a number of researchers. The thesis work complements the work on networking by providing resource reservation in the end hosts as well as in the network. Thus the operating system resource reservation work enables predictable end-to-end performance for real-time programs.

Ferrari and Verma [31] describe a model for guaranteeing bandwidth in a wide-area network. Their analytical model provided a basis for subsequent work on network bandwidth reservation. In contrast, Clark *et al.* [18] describe another service model based on the idea of predictive service. In related work, L. Zhang *et al.* [140] gave a basic description of RSVP, a protocol for reserving resources across nodes in an internetwork. This thesis provides operating system support to implement these types of schemes. Reserves support guaranteed service for hard real-time applications as well as supporting predictive service for soft real-time applications that dynamically change their requirements on various system resources.

H. Zhang [141] describes a bandwidth reservation model and an implementation in an internetwork protocol. He was able to allocate and control network bandwidth in gateways, but did not address the allocation and control of resources like the processor in the more general operating system environment. The reserve system would make it possible to address resource reservation issues in general end systems, extending H. Zhang's work on resource reservation within the network and its routers.

Anderson *et al.* [4] describe a Session Reservation Protocol (SRP) which reserves resources along the route of a connection to ensure a particular bandwidth and delay for the connection. This protocol provides resource reservation at routers for predictable network performance. The reserve system would make it possible to reserve resources at end hosts as well as in the network routers.

7.2 Scheduling theory

Real-time scheduling theory has been an active area of research for years. Many important theoretical results date back to the 1950's and 1960's [20,49], and work on system design dates back to the same time period [71,73,82]. More recently, real-time systems research has focused on scheduling algorithms and techniques for building complex, distributed real-time systems [1,130,131].

The reserve system was designed using several results from real-time scheduling theory in its admission control policy, scheduling algorithm, and synchronization and communication primitives. The reserve model is based on the original rate monotonic scheduling analysis [67] as well as recent extensions [63]. The simple two-parameter reservation model is suitable for a wide range of applications, but applications that require different reservation semantics might need a model that takes advantage of other scheduling algorithms and analyses. For example, an application might require generalized rate monotonic analysis [39,64,107,109], aperiodic servers with different replenishment policies [111,112,120], earliest deadline first scheduling [27,49,67], sporadic task scheduling [51], or deadline monotonic scheduling [7].

The reservation system also depends of priority inheritance protocols for fixed priority scheduling [96,108], and similar inheritance protocols for earliest deadline first [8,16] would be required for a reservation model based on that policy.

The two-parameter periodic scheduling framework of the original rate monotonic scheduling analysis divides the capacity of the processor among multiple tasks. Other scheduling techniques such as processor sharing and fair share scheduling aim to provide a similar kind of proportional sharing of the processor. The primary difference is that the two-parameter periodic scheduling framework specifies a granularity of sharing by specifying a period. Processor sharing and fair share scheduling seek to support sharing at a very fine granularity.

A processor sharing technique [50] intended to be accurate enough to accommodate the timing constraints of arbitrary multimedia applications would require a very small quantum. This would imply a high scheduling overhead. Such a system would also require some method for controlling the effects of synchronization and communication between processes.

Fair share schedulers [41,58,138] provide for resource allocation like processor sharing, although at a coarser granularity. Such schedulers ensure that users who pay more for compute time get better service than others who pay less do. They record usage measurements to try to match usage with target allocation levels over the long term.

More recently, work on fair share scheduling and proportional share scheduling has addressed the integration of network scheduling and end-system scheduling [118,133,134]. Instead of focusing on ensuring that a certain amount of computation time will be available by a deadline, this work focuses on ensuring that a certain proportion of the processor is available to a compute-bound task in any interval.

7.3 Applications

Two classes of applications turn out to be very important to the resource reservation work. Adaptive applications are important because dynamic real-time applications must be very sensitive to the relationship between their (changing) resource requirements and their levels of resource reservations. In addition, design tools, performance monitors, and dynamic resource allocation tools are necessary for the design and on-line monitoring and tuning of resource reservations for dynamic real-time applications.

7.3.1 Adaptive applications

Recently there has been a focus on how systems and applications can adapt their computations to the resources they find available to them. For example, video compression algorithms are designed to allow for tradeoffs in resource requirements in various ways. Software techniques are also being developed, primarily in the area of mobile computing, for application awareness of resource requirements and adaptation based on system resources available.

The MPEG compression algorithms support several ways to trade off between bandwidth, computational resources, and image quality [17,61]. The MPEG-1 q-factor trades image quality for less bandwidth and computational resources, and the MPEG-2 hierarchical encoding scheme supports incremental improvements in image quality for additional bandwidth and computation time [10,17]. Other methods such as subband encoding [121], Hyden's method [46], and other hierarchical encoding schemes [2,17] support this tradeoff as well.

Recent work in mobile computing explores how applications can be sensitive to the resources that are available to them in terms of network bandwidth, processor power, and screen resolution among others [32,91,106]. These applications discover the resources that are available to them and then use this information to guide their own computations. For example, a video player residing on a high-end workstation might request from a video server a full color (24 bits per pixel), full motion (30 frames per second), relatively high resolution (640x480 pixels) video stream. The same video player on a low-end personal computer might request only 256 bits of color, 15 frames per second, and a resolution of 160x120 pixels. Supporting a scenario like this requires that all of the involved system components are aware of the resources they have available, the resources they need to do their work, and the level at which all of the other components in the end-to-end activity can perform.

7.3.2 Tools

The reserve system provides a mechanism for tools that monitor and control resource usage. This section discusses current tools for monitoring functions and for controlling resource usage.

Different tools intended for monitoring various aspects of system performance work at different levels. Some are intended for program design while others are intended for system-level debugging and on-line resource monitoring.

Performance analysis tools such as gprof [29,37] and PCA [28] use PC sampling techniques to characterize program runtime behavior and object code to determine the static structure. This method gives a great deal of insight into the behavior of individual programs, but there is no notion of tuning task sets as a whole. Also, the method of exercising control of the programs being analyzed is through the programs themselves, either changing their structure or modifying their parameters. The tool does not exercise this control directly.

System monitoring tools typically separate the functions of capturing performance data, analyzing the data, and having an effect on the system that was measured. The Advanced Real-Time Monitor (ARM) [123], which was originally designed for the ARTS Kernel [124] and more recently updated for RT-Mach [125], takes this approach. ARM uses a kernel mechanism to capture scheduling events and then sends those events over the network to an ARM application running on a different machine. ARM then displays a scheduling history based on the events, and this history can be viewed, analyzed, and saved for future use.

Tools like xload [74] provide a very simple view of the cumulative processor load on a workstation. The xload application does this on-line, but it leaves out some interesting information like a breakdown of which processes are consuming what percent of the load. It also lacks a control element to help the user have an effect on the load through the tool.

Tools like the Memory Sizer on the Macintosh [6] offer control over a system resource, but the resource information is very simple, and one cannot set the memory size of a program while it is running. The reserve system allows for much more sophisticated control of system resource allocation. With the help of a QOS manager, a tool such as rmon can graphically display resource usage information and interact with the console user to control resource allocation.