Quality of Service and System Design

Kevin T. Kornegay†, Gang Qu‡, and Miodrag Potkonjak‡
† Department of Electrical Engineering, Cornell University, Ithaca, NY 14853-5401
‡ Computer Science Department, University of California, Los Angeles, CA 90095-1596

Abstract

Quality of Service (QoS) of the implementation of an application can be defined as a function of the properties of the application and its implementation as observed by the user and/or the environment. Typical application and implementation properties include latency, throughput, jitter, and the level of resolution. Many of the current and pending most popular applications, such as multimedia, wireless sensing and communications, security and PEBBs, have intrinsic relevant QoS components. Recently, quality of service attracted a great of deal of attention in a number of research and development communities, and in particular, in the network and multimedia literature.

However, until now synthesis and CAD research did not addressed how to design systems with quantitative QoS requirements. Our goal in this paper is to outline foundations and framework in which QoS system design trade-offs and optimization can be addressed. We first identify and state in synthesis-usable way two currently most popular approaches to Quality of Service treatment: Q-RAM and DS-curve (demand/service). We discuss advantages and limitations of the two approaches. Next, we show how these two approaches can be combined in a new more comprehensive QoS framework. We also explain and illustrate using examples interaction between QoS and synthesis and compilation tasks. We conclude by identifying and discussing the future directions related to synthesis of QoS-sensitive systems.

1 Introduction

Traditionally, the most successful domain of behavioral synthesis has been high-level synthesis of computationally intensive designs. These type of design follows the synchronous data flow model of computation. Many programs in applications such as digital signal processing, communications, and control theory applications can be expressed using this computational abstraction. This is in particular true from smaller applications, such as filtering, FFT, DCT, and cordic [14]. The success in this domain can be mainly attributed to a very clear, simple, and tractable abstraction: fully repetitive processing of fully periodic streams of input data. This restricted semantics has as a prime ramification has applicability of compile-time static scheduling and great amenability for optimization using transformations.

At the same time, processor design was a well understood process. A widely accepted benchmarks, such as SPEC, have been driving processor optimization. While this task is considerably more complex than DSP behavioral synthesis, the designers leveraged on quantitative optimization procedures based on implementation optimization of the set of benchmark programs.

However, the recent application and technology trends are making the traditional behavioral DSP synthesis and microprocessor design techniques obsolete in a sense that they are not adequate for specifying and optimization the pending most popular applications. These new applications have a new set of objectives and constraints. For example, in multimedia applications the parameters such as jitter, latency, synchronization and a need for multiresolution are considered of primary importance. All these parameters form Quality of Service (QoS) for a given application.

We can define Quality of Service (QoS) of the implementation of an application as a function of the properties of the application and its implementation as observed by the user and/or the environment. Many of the current and pending most popular applications, such as multimedia, wireless sensing and communications, security and PEBBs, have intrinsic relevant QoS components. Recently, quality of service attracted a great of deal of attention in a number of research and development communities, and in particular, in the network routing and multimedia delivery.

However, until now synthesis and CAD research did not addressed how to design systems with quantitative QoS requirements. Our goal in this paper is to outline foundations and framework in which QoS system design trade-offs and optimization can be addressed.

The rest of the paper is organized in the following way. We first identify and state in synthesis-usable way two currently most popular approaches to Quality of Service treatment: Q-RAM and DS-curve (demand/service). We discuss advantages and limitations of the two approaches. Next, we show how these two approaches can be combined in a new more comprehensive QoS framework. We also explain and illustrate using examples the interaction between QoS and synthesis and compilation tasks. We conclude by identifying and discussing the future directions related to synthesis of QoS-sensitive systems.

2 Related Work

Most of the research on quality of service is in the networking community and especially distributed multimedia systems. There have been several proposals and prototype implementations of end-to-end transport protocols for delivering QoS guarantees, such as ST-II [18], RSVP [19], and the Tenet Real-Time Protocol Suite [3]. For example, RSVP provides a mechanism for reserving resources along the path from a source host to a destination host so that subsequent data packets are guaranteed to have certain bandwidth available and meet certain delay bounds. Many QoS architectures have been proposed: TINA[8], the QoS broker[12], Globus[20], Darwin[6] and a survey can be found in [2].

How to measure the QoS has been the one of the fundamental problems. The quality of the complex real-time, distributed multimedia services should be application specific, user dependent and thus it is hard to find an explicit one-fit-all definition for QoS. In multimedia applications, at the highest level, QoS can be interpreted as an image quality or a sound quality, and at lower levels, it can be measured by bits per second or transit delay. In [1], QoS is defined as a combination of the basic quality metrics for the network layer: delay, jitter, bandwidth, and reliability. Lawrence [10] discusses the metrics based on the QoS attributes of timeliness, precision, and accuracy that can be used for system specification, instrumentation, and evaluation.

Cruz [7, 17] introduced the arrival curve and service curve in the context of packet-switch networks. From these curves, one can view QoS in terms of backlog, transmission delay and throughput. The problem of satisfying services guarantees becomes a scheduling problem to meet the backlog and latency constraints. Rajkumar et al. [15] present an analytical approach for satisfying multiple QoS dimensions in a resource-constraint environment (see below for detail) and provide optimal and near-optimal resource allocation schemes for two special cases. Later on, [16] shows this problem is NP-hard and gives a polynomial algorithm which yields a solution within fixed short distance from the optimal solution as well as an approach by formulating the problem as a mixed integer programming problem. Recently, Qu and Potkonjak [13] considered the low power system design with guaranteed QoS. They treated energy as a special resource and showed how the QoS can be maximized with fixed energy consumption by using variable supply voltages.

3 The QoS-based Resource Allocation Model

Rajkumar et al. [15] proposed the QoS-based Resource Allocation Model (Q-RAM) to analyze two problems: (i) Satisfying simultaneous requirements along multiple QoS dimensions such as timeliness, security, data quality, dependability, and reliable packet delivery, and (ii) Allowing applications have access to multiple resources such as CPU, disk bandwidth, network bandwidth, memory simultaneously.

Q-RAM considers a system of n applications and m resources each with a finite capacity, as well as a set of QoS requirements. An application, for each QoS dimension, has a minimum resource requirements, and it achieves a certain utility with the allocated resources. The objective is to make resource allocations to each application such that every application satisfies its QoS requirements on all dimensions and the total system utility (which is a weighted sum of each application's utility) is maximized. Following is a detailed description of Q-RAM:

- Resources: The system has m sharable resources $\{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_m\}, m \geq 1$, each resource has a finite capacity. We abuse the notation a little bit by denoting $\mathbf{R}_i$ the capacity for resource $\mathbf{R}_i$.
- Applications: There are n applications $\{\tau_1, \tau_2, \cdots, \tau_n\}$, $n \geq 1$ to be executed on a system with the above resources.
- QoS dimensions: Each application τ_i needs to satisfy requirements along d QoS dimensions. $\{Q_1, Q_2, \cdots, Q_d\}$, $d \geq 1$. τ_i has minimal resource requirements on QoS dimension Q_k denoted by $R_i^{min_k} = \{R_{i,1}^{min_k}, R_{i,2}^{min_k}, \cdots, R_{i,m}^{min_k}\}$ where $R_{i,j}^{min_k} \geq 0, 0 \leq j \leq m$.
- Application utility: The utility U_i of an application τ_i is the value that is accrued by the system when τ_i is allocated $\mathbf{R}^i = (R_{i,1}, \cdots, R_{i,m})$. Each application τ_i has a relative importance specified by a weight w_i.
- System utility: The total system utility with a resource allocation scheme $(\mathbf{R}^1, \mathbf{R}^2, \cdots, \mathbf{R}^n)$ is $U = \sum_{i=1}^{n} w_i U_i(\mathbf{R}^i)$.

The objective of the Q-RAM is to make resource allocations to each application such that the total system utility is maximized under the constraint that every application satisfies all QoS dimensions. Stated formally, we need to determine $\{R_{i,j}, 1 \leq i \leq n, 1 \leq j \leq m\}$ such that $R_{i,j} \geq \sum_{k=1}^{d} R_{i,j}^{min_k}$ and U is maximized over all possible allocations.

Figure 1 shows the utility functions of two applications τ_1 and τ_2. There is a single resource with a total amount of 10 units. The minimal resource requirements for τ_1 and τ_2 are 3 and 2 units respectively. We want to allocate the 10 units of resource to the two application and maximize the total utility. For example, if we split the resources in halves, the total utility is $U = U_1(5) + U_2(5) = 5 + 2 = 7$. Observing that τ_2 yields the same amount of utility from 3 units of resource to 5 units, while τ_1 will gain about 1.5 more

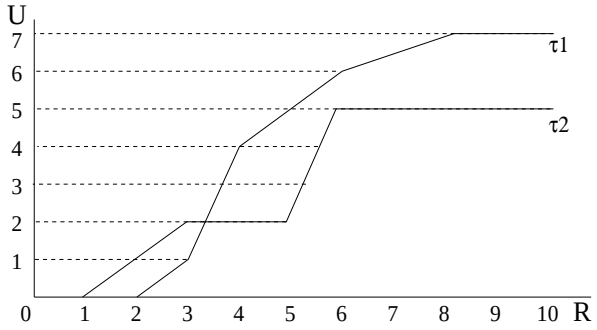

Figure 1: Utility functions of two applications.

units of utility if it receives 7 units of resource instead of 5. This results in a total utility $U = U_1(7) + U_2(3) = 6.5 + 2 = 8.5$. However, since U_2 has a sharp increment in the region between 5 units and 6 units resources, allocating 6 units resource to τ_2 and the remaining 4 units to τ_1 gives a total utility of $U = 4 + 5 = 9$. It is clear that the system utility can be greatly improved by careful resource allocation schemes.

Rajkumar et al. [16] showed that this Q-RAM problem of finding the optimal resource allocation to satisfy multiple QoS dimensions is NP-hard by an induction to the inexact 0-1 knapsack problem. For the case of single resource and single QoS dimension or multiple independent QoS dimensions (i.e., changes along any QoS dimension have no impact on the other QoS dimensions.), they provided optimal resource allocation schemes [15] based on the Kuhn-Tucker theorem. The problem of apportioning multiple resources to satisfy a single QoS dimension is solved efficiently as a mixed integer programming problem [16].

4 Demand/Service-Curve Model

Cruz [7] introduced the notion of service curves and proposed their usage as a general framework for service characterization, mainly in the domain of packet-switched networks. One key feature of charactectizing service for a connection using a service curve is that the QoS guarantees can be expressed as simple functions of the service curve and the traffic burstiness constraint of the connection. Another feature of the service curve specification is that is gives greater flexibility to a server in allocating its resources to meet diverse delay and throughput requirements. Later on, [17] proposed the service curve-based earliest deadline first scheduling policy, which provides guarantees to a virtual circuits in packet-switch networks.

We now explain this framework in the general domain of QoS instead of the packet-switch networks. Assuming that time is divided into slots, and the system receives (and buffers) computation demand from several applications and services up to c (capacity of the system) units of computation per slot in a "cut-through" manner (i.e., a computation demand may be serviced in the same slot as it arrives.). The amount of computation demand from all the applications which are not yet accomplished (stored in the system) at the end of a slot is called the *backlog* at the end of this slot. The *latency* (called virtual delay in Cruz's work) relative to the slot t is the difference between t and the first slot when the system finishes the computation requests arrive before slot t.

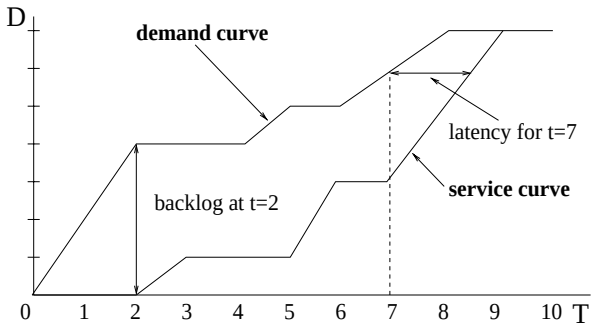

Figure 2: The demand and service curves for an application.

Several QoS dimensions have explicit interpretations in the demand and service curves. Figure 2 shows such curves for an application. The backlog at time $t = 2$ is the vertical distance between the two curves and the latency for time $t = 7$ is the horizontal distance. Figure 3 illustrates the

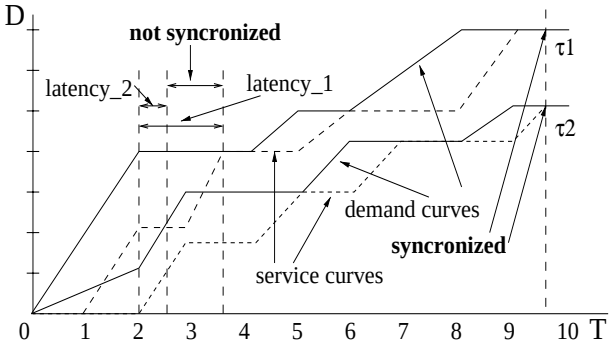

Figure 3: The synchronization of two applications.

synchronization of two applications τ_1 and τ_2 (for example, video and audio signals). The two solid curves are the computation demand curves and the dotted ones represent the service curves. The system keeps on switching between τ_1 and τ_2, serving one at a time (consequently, at most one service curve increases), to maintain the synchronization. We say that τ_1 and τ_2 are *synchronized* at the end of slot t if τ_1 and τ_2 have the same latency relative to the slot t. For example, τ_1 and τ_2 are synchronized at time $t \approx 9.5$ because the system has serviced both computation requests. On the other hand, at slot 2, τ_1's demand cannot be satisfied until around 3.75 which causes a 1.75-slot latency. Similarly, τ_2 has a roughly 0.5-slot latency. We see that τ_1 is behind τ_2 and they

are not synchronized. Synchronization can be quantitatively measured by the difference of the corresponding latencies where a small amount implies a good QoS. Obviously, the synchronization of τ_1 and τ_2 at slot 2 can be improved by serving τ_1 more before starting τ_2.

Now the connection between scheduling and QoS guarantees becomes clear. Sariowan et al. [17] proposed the service curve-based earliest deadline first (SCED) scheduling policy and showed that if the SCED policy results in a scheduler which does not violate the latency constraints, then the QoS guarantees can be achieved. The SCED policy says: each demand of computation is assigned a deadline and applications are serviced earliest deadline first in a work-conserving manner. Specifically, in a given slot up to c (the capacity of the system) units of computation with the smallest deadlines are selected to be served.

5 A Combined Model

Both the Q-RAM and Demand/Service curve model try to answer the problems of QoS guarantees. They capture different aspects of the problem and take different approaches. Each enjoys its advantage and suffers its limitations as well.

The Q-RAM, as an abstract QoS model, is highlighted by the observation that different resource allocation schemes result in different levels of QoS and thus transform the QoS guarantee problem to the well-studied resource allocation problem. In almost all current literature of Q-RAM, one basic assumption is that the system has more than enough resource to meet the requirements of all QoS dimensions from all applications (i.e., system is underloaded) and the discussion is limited at how to improve the QoS. Little work has been recognized for overloaded systems under the Q-RAM model. Another limitation for Q-RAM is the smoothness requirement on the utility functions. The QoS requirement and the utility function in the Q-RAM have little restrictions to capture the universal nature of the QoS. However, this also brings difficulties in analysis. For example, in Rajkumar et al.'s work, most general results are based on the assumption that the utility functions are twice continuously differentiable and concave [15]. On the other hand, in practice these functions are always oversimplified. Q-RAM has already found its connections with the resource allocation problem and integer programming problem, and because of the abstract nature of the model, this is a very promising approach once some utility functions and QoS dimensions are built on top of the Q-RAM.

The Demand/Service curve model comes from, but not limited to, the packet-switch networks. In this sense, it has very specific measures for the QoS guarantees, like backlog, transmission delay, throughput, synchronization and so on. One advantage of DS model over Q-RAM is that it treats time differently. In the original Q-RAM, time plays the same role as other resources, therefore, it is hard to describe the properties of real-time applications whose QoS dimensions depend on time. The demand curves reflect the burstiness constraints on arriving applications and the service curves provide the guaranteed service. Interestingly, similar to the Q-RAM approach, the QoS guarantee problem in DS curve model is interpreted as another well-studied problem: the scheduling problem. Besides its other advantages, many results have very clear intuition from the demand/service curves.

It is possible to have a combined model that takes advantages of both models. One straightforward example is as follows: a system with finite resources can make profit by serving multiple applications, each application tells the system its computation demand curve and agrees with the system on the utility function (the system's profit) based on the QoS provided by the system. The system wants to determine which application will get service and at which level of QoS to maximize its profit. This problem does not fit either of the Q-RAM and DS curve model, however, there is an approach based on both models. From the agreed utility functions, the Q-RAM provides us the optimal/suboptimal resource allocation schemes. Then for each application, with its assigned resources from the previous step, the DS curve model determines whether there exist schedulers that satisfy all the QoS requirements. If no such scheduler, we can take a suboptimal solution form Q-RAM and test the schedulability in the DS curve model. This procedure can be continued till we get a strategy for the system.

6 Energy Minimization with Guaranteed QoS

Low power consumption is one of the most important criteria for the design of systems and devices that carry out the distributed multimedia applications which needs a guaranteed QoS. [13] proposed the problem of Resource Allocation and Energy Minimization with guaranteed QoS (RAEM_QoS) and its dual problem, which targets at maximizing QoS with fixed amount of energy. The key idea is to exploit the advantages provided by the variable voltage design methodology [9] to choose the voltage optimally for the purpose of minimizing energy consumption with a guaranteed amount of QoS. In particular, the problems for single application system with a special type of QoS function was solved optimally and a Partition and Linear Approximation (PLA) heuristic for general QoS functions has been proposed. For system with multiple applications a dynamic programming proce-

dure was presented based on the concept of quantization of QoS and resources [13].

Consider two independent applications τ_1, τ_2 and their QoS *vs.* execution time functions (similar to the utility curves in Figure 1) at the reference supply voltage v_{ref} shown in Table 1, we want to distribute the execution time T between τ_1 and τ_2, and determine the voltage profiles so that both applications can be finished by time T with their respective QoS requirements and the total energy consumption is minimized.

Application	τ_1	τ_2
Deadline	T	T
QoS requirement	U_0^1	U_0^2
QoS *vs.* execution time at v_{ref}	$U_{ref}^1(t)$	$U_{ref}^2(t)$

Table 1: Characteristics of applications τ_1 and τ_2.

The PLA heuristic [13] solves for single application with parameters as shown in Table 1, how to get the required amount of QoS by given deadlines and minimize energy consumption. Suppose that in the best strategy for two applications, τ_1 is completed in time t_1 with a (variable) voltage $v_1(t)$ and τ_2 consumes time $t_2 (\leq T - t_1)$ with voltage $v_2(t)$. We construct two (independent) single application problems:

(I) Find the voltage scheme $v(t)$ to finish application τ_1 by time t_1 with QoS requirement U_0^1 and minimize the energy consumption.

(II) Find the voltage scheme $v(t)$ to finish application τ_2 by time t_2 with QoS requirement U_0^2 and minimize the energy consumption.

Since any part of the optimal strategy is also optimal to the corresponding sub-problem, we know that for (I), an optimal way is to finish exactly at time t_1 with the voltage scheme $v(t) = v_1(t)$ and a similar solution exists for problem (II). We observe that for a strategy to the two applications problem to be optimal, both parts of the strategy should be an optimal to the single application system. Based on this necessary condition, we propose in Figure 4 the quantization approach.

In procedure Quant, we introduce discreteness into the continuous time domain by quantizing the execution time in step (2). Suppose the first application gets the first i quants, i.e., it reserves the CPU time $[0, t_i]$, then we are able to determine a "best" strategy by passing $\tau_1, U_0^1, t_i, U_{ref}^1(t)$ as parameters to the PLA procedure. After we get the "best" strategy (that we can get from PLA) S_1 which finished τ_1 at time t, we can give the rest of the execution time to τ_2 and use PLA again to determine strategy S_2 from τ_2. The sum of energy consumption by S_1 and S_2 is the total energy consumption. The for loop in step (3) is an exhaustive search

Input: Two applications as shown in Table 1.
Output: Execution time T_1, T_2 and the voltage schemes $v_1(t), v_2(t)$ for the two applications with $T_1 + T_2 \leq T$ such that the QoS requirements U_0^1, U_0^2 are satisfied and the total energy consumption is minimized.

Procedure Quant:
(1) $E_{total} = E_1 = E_2 = \infty$;
(2) Quantize $[0, T] : 0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$;
(3) for each $i = 1, 2, \cdots n$
{ PLA ($\tau_1, U_0^1, t_i, U_{ref}^1(t)$): find strategy S_1 that finish τ_1 at time $t \in [0, t_i]$ with energy consumption E_1;
PLA ($\tau_2, U_0^2, T - t, U_{ref}^2(t)$): find strategy S_2 that finish τ_2 at time $t' \in [0, T - t]$ with energy consumption E_2;
if ($E_1 + E_2 < E_{total}$)
update the best strategy $S = (S_1, S_2)$;
}
(4) Report the strategy $S = (S_1, S_2)$;

Figure 4: Quant heuristic for the two applications problem.

over the quants and at the end the one that achieves least total energy consumption is reported.

For a set of k applications, we can apply the dynamic programming technique and with careful quantization on both the QoS and resources, solutions to the RAEM_QoS can be arbitrarily close the optimal.

7 QoS and System Design: Future Directions

In this section we briefly outline several directions related to synthesis and optimization of efficient implementation of QoS-sensitive applications.

The first line of research is to find a proper computational models and abstractions for QoS systems. In the previous two section we presented in synthesis friendly Way two most popular models, Q-RAM and DS curve, which aim to address QoS issue. It is important to note that actually there are also numerous other approaches which try to address a variety of QoS components.

For example, Mok and Baruah [4, 5, 11] have been popularing the multiframe model for real-time tasks. Traditionally, the real-time systems task model follows a worst case execution time bound for every task. Therefore, the traditional real-time system task model may be often overly pessimistic if the worst case execution time of a task is much longer than the average. Mok and Chen [11] introduced the multiframe real time task model which allows the execution time of a part of a periodic task to vary from one instance to another by specifying the execution time of a task in terms of a sequence of modular numbers. It has been show that this model is well suited for taking scheduling advantages of burstiness of MPEG streams [4]. Therefore, this model provides, at least in some cases, an alternative to the Cruz DS-curve model.

We believe that in practice, QoS statistical models will prove exceptionally useful. Statistical model have a potential to provide simultaneously compact representation of QoS metrics and tasks characteristics as well as precise enough

data which will facilitate analysis and synthesis of QoS applications.

The currently only addressed synthesis and compilation task in QoS systems is scheduling. It is well know that scheduling has a relatively low impact on the overall system performances, in particular when compared with other synthesis tasks such transformations, partitioning, template matching, and resource allocation. Similarly, currently the only system metrics discussed in the QoS literature are latency and throughput. QoS paradigm will also require that optimization of all traditional design metrics in behavioral and system synthesis is revisited, including area, power, and testability. Our preliminary studies [13] indicate that power optimization under QoS constraints poses a number of high potential challenging optimization tasks.

Behavioral synthesis suffered for many years lack of adequate benchmarks. Obviously, real-life QoS benchmarks are the mandatory prerequisite for CAD QoS tools. At the same time, there is a strong need for well documented design studies of complete systems which address QoS-sensitive applications.

8 Conclusion

Quality of service is intrinsically connected to many major and most popular applications such as multimedia and wireless sensing. In order to properly address design and automatic synthesis of these application into systems-on-chip, tractable and quantitative definitions of QoS are required. We propose the first QoS framework and show how two currently most popular approaches to QoS treatment: Q-RAM and DS-curve (demand/service) can be integrated within this framework. The Q-RAM model is currently the most popular QoS abstraction in the real-time system community and the DS-curve model is widely accepted in network switching and routing community. We explain, how one can treat a number of properties within a QoS model. For instance, we give a new interpretation for synchronization constraints with the DS-curve model.

Furthermore, we explain and illustrate using examples the interaction between QoS and synthesis and compilation tasks. For example, we show how one can optimize CPU-time distribution between two tasks so that power consumption is minimized for a given level of QoS for each task. We concluded by identifying and discussing the future directions related to synthesis of QoS-sensitive systems.

REFERENCES

[1] J. Altmann and P. Varaiya. *INDEX project: user support for buying QoS with regard to user's preferences.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. 101-104, 1998.

[2] C. Aurrecoechea, A.T. Campbell, and L. Hauw. *A Survey of QoS Architectures.* Multimedia Systems, Vol.6, No.3, pp. 138-151, May 1998.

[3] A. Banerjea, D. Ferrari, B. Mah, M. Moran, D. Verma, and H. Zhang. *The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences.* IEEE/ACM Transactions on Networking, Vol.4, No.1, pp. 1-11, February 1996.

[4] S.K. Baruah. *A general model for recurring real-time tasks.* Proceedings 19th IEEE Real-Time Systems Symposium, pp. 114-122, 1998.

[5] D. Chen, A. Mok, and S. Baruah. *On modeling real-time task systems.* Lectures on Embedded Systems. (Ed. G. Rozenberg and F. Vaandrager.) Berlin, Germany: Springer-Verlag, pp. 153-169, 1998.

[6] P. Chandra, A.L. Fisher, C. Kosak, and P. Steenkiste. *Network support for application-oriented QoS.* 1998 Sixth International Workshop on Quality of Service (IWQoS'98), pp. 187-195, 1998.

[7] R.L. Cruz. *Quality of Service Guarantees in Virtual Circuit Switched Networks.* IEEE Journal on Selected Areas in Communications, Vol.13, No.6, pp. 1048-1056, August 1995.

[8] F. Dupuy, C. Nilsson, and Y. Inoue. *The TINA Consortium: Toward Networking Telecommunications Information Services.* IEEE Communications Magazine, Vol.33, No.11, pp. 78-83, November 1995.

[9] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.B. Srivastava. *Power Optimization of Variable Voltage Core-Based Systems.* Proceedings 1998 Design and Automation Conference (35th DAC), pp. 176-181, 1998.

[10] T.F. Lawrence. *The quality of service model and high assurance.* Proceedings. 1997 High-Assurance Engineering Workshop, pp. 38-39, 1997.

[11] A.K. Mok and D. Chen. *A multiframe model for real-time tasks.* IEEE Transactions on Software Engineering, October 1997, Vol.23, No.10 pp. 635-645, 1997.

[12] K. Nahrstedt and J.M. Smith. *The QoS Broker.* IEEE Multimedia, Vol.2, No.1, pp. 53-67, Spring 1995.

[13] G. Qu and M. Potkonjak. *Energy Minimization with Guaranteed Quality of Service.* Unpublished manuscript, 1999.

[14] J.M. Rabaey, C. Chu, P. Hoang, and M. Potkonjak. *Fast prototyping of datapath-intensive architectures.* IEEE Design & Test of Computers, June 1991, Vol.8, No.2, pp. 40-51, 1991.

[15] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. *A resource allocation model for QoS management.* Proceedings. The 18th IEEE Real-Time Systems Symposium, pp. 298-307, 1997.

[16] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. *Practical Solutions for QoS-based Resource Allocation Problems.* Proceedings. The 19th IEEE Real-Time Systems Symposium, pp. 296-306, 1998.

[17] H. Sariowan, R.L. Cruz, and G.C. Polyzos. *Scheduling for quality of service guarantees via service curves.* Proceedings Fourth International Conference on Computer Communications and Networks (ICCCN'95), pp. 512-520, 1995.

[18] C. Topolcic. *Experimental Internet Stream Protocol, Version 2 (ST-II).* Internet RFC 1190, October 1990.

[19] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. *RSVP: A New Resource ReSerVation Protocol.* IEEE Network Magazine, Vol.7, No.5, pp.8-18, September 1993.

[20] http://www.globus.org/