

AFRL-IF-RS-TR-2003-23
Final Technical Report
February 2003



**EPIQ – A META-COMPUTING FRAMEWORK FOR
SCALABLE, RESPONSIVE AND
RECONFIGURABLE END-TO-END RESOURCE
MANAGEMENT, AND AGILE OBJECTS:
MIDDLEWARE FOR SURVIVABLE
INFORMATION SYSTEMS**

University of Illinois at Urbana-Champaign

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-23 has been reviewed and is approved for publication.

APPROVED:

A handwritten signature in black ink, appearing to read "Edward L. DePalma". The signature is fluid and cursive, with a long horizontal stroke extending to the right.

EDWARD L. DEPALMA
Project Engineer

FOR THE DIRECTOR:

A handwritten signature in black ink, appearing to read "James W. Cusack". The signature is cursive and somewhat stylized, with a large loop at the beginning.

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE FEBRUARY 2003	3. REPORT TYPE AND DATES COVERED Final Feb 97 – May 02	
4. TITLE AND SUBTITLE EPIQ – A META-COMPUTING FRAMEWORK FOR SCALABLE, RESPONSIVE AND RECONFIGURABLE END-TO-END RESOURCE MANAGEMENT, AND AGILE OBJECTS: MIDDLEWARE FOR SURVIVABLE INFORMATION SYSTEMS			5. FUNDING NUMBERS C - F30602-97-2-0121 PE - 62301E PR - E524 TA - 01 WU - 01	
6. AUTHOR(S) Klara Nahrstedt				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Illinois Department of Computer Science 1304 West Springfield Avenue Urbana Illinois 61801			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFSF 525 Brooks Road Rome New York 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2003-23	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Edward L. DePalma/IFSF/(315) 330-3069/ Edward.DePalma@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) The objective of the Quorum Program was to understand the basic principle and algorithms of middleware systems for QoS-aware and survivable information systems. The EPIQ project aimed to develop a meta-computing framework for scalable, responsive, and reconfigurable end-to-end resource management. This meta-computing framework develops end-to-end QoS and resource management strategies that can be customized and integrated to provide guaranteed services of negotiated quality to time-critical C3I applications. Specifically, the framework provides end-to-end QoS and resource management to flexible applications and enables applications to adapt their quality if dynamic changes occur in requirements, demand on resources, and availability of resources. EPIQ multi-dimensional QoS and resource management mechanisms are application-independent, but permit the integration with application-specific and user-oriented mechanisms, and give the user a crucial control in QoS, service and resource allocation adaptation, graceful degradation and recovery. The QoS management and resource management framework is further expanded through off-line QoS programming and compilation environments to allow easy development of flexible multimedia applications within our framework. The framework validation is done through an open real-time run-time environment that provides end-to-end real-time performance computing and communication support for hard-real-time applications as well as end-to-end soft performance guarantees for soft real-time and flexible applications.				
14. SUBJECT TERMS Resource Management, Quality of Service, QoS, Middleware			15. NUMBER OF PAGES 30	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

SUMMARY	1
INTRODUCTION	1
METHODOLOGY	2
ASSUMPTIONS	2
PROCEDURES	3
END-TO-END QoS MANAGEMENT	4
<i>QoS/Resource Model</i>	4
<i>QoS Adaptation Model</i>	5
<i>End-to-End Protocols and Services</i>	5
QoS-AWARE RESOURCE MANAGEMENT	6
QoS NETWORK MANAGEMENT	8
RESULTS AND DISCUSSIONS	11
END-TO-END QoS MANAGEMENT	11
QoS-AWARE RESOURCE MANAGEMENT	13
QoS NETWORK MANAGEMENT	14
CONCLUSION	15
ACCOMPLISHMENTS	15
EDUCATIONAL IMPACT	17
TECHNOLOGY TRANSITION	17
APPENDIX A	18
BIBLIOGRAPHY	18
LIST(S) OF SYMBOLS, ABBREVIATIONS AND ACRONYMS	25

List of Figures

FIGURE 1: EPIQ META-COMPUTING FRAMEWORK	3
FIGURE 2: DISTRIBUTED SYSTEM OF HETEROGENEOUS, DEPENDENT APPLICATIONS MODELED AS PIPELINED TASKS	4
FIGURE 3: INPUT AND OUTPUT QoS FOR EACH TASK T	5
FIGURE 4: CONTROL-BASED QoS ADAPTATION MODEL	5
FIGURE 5: OPEN REAL-TIME SYSTEMS	7
FIGURE 6: EPIQ MIDDLEWARE IN ACTION RUNNING VISUAL TRACKING.....	12
FIGURE 7: PERFORMANCE COMPARISON (FEASIBLE SCHEDULE SUCCESS RATE) BETWEEN EPIQ OPEN SYSTEM AND A CLOSED SYSTEM.	14

Summary

The EPIQ project developed a Quality of Service (QoS) aware middleware and resource management framework

- (a) to allow for development and experimentation with novel algorithms, methods, protocols, services, and applications
- (b) to understand scalability, responsiveness, adaptability and reconfigurability within an end-to-end QoS and resource management
- (c) to validate important approaches for middleware which will be used in survivable information systems.

The report discusses three important pieces of EPIQ project:

- (1) the end-to-end QoS management approaches
- (2) resource management approaches
- (3) QoS network management approaches

The report presents validation of our approaches through implementations and simulations such as the implementation of end-to-end QoS management for hard-real-time applications, control-based algorithms for Agilos adaptive middleware systems, open system integration of middleware and resource management services to allow co-existence of hard-real-time and soft-real-time applications, and others.

Introduction

The objective of the Quorum Program was to understand the basic principle and algorithms of middleware systems for QoS-aware and survivable information systems. Our EPIQ project aimed to develop a meta-computing framework for scalable, responsive, and reconfigurable end-to-end resource management. As this report shows, we have achieved a large range of novel solutions, very much applicable to middleware systems and their underlying end-system and network resource management for C3I applications.

Our meta-computing framework develops end-to-end QoS and resource management strategies that can be customized and integrated to provide guaranteed services of negotiated quality to time-critical C3I applications. Specifically, the framework provides end-to-end QoS and resource management to flexible applications and enables applications to adapt their quality if dynamic changes occur in (a) requirements, (b) demand on resources, and (c) availability of resources. EPIQ multi-dimensional QoS and resource management mechanisms are application-independent, but permit the integration with application-specific and user-oriented mechanisms, and give user a crucial control in QoS, service and resource allocation adaptation, graceful degradation and recovery. The QoS management and resource management framework is further expanded through off-line QoS programming and compilation environments to allow easy development of flexible multimedia applications within our framework. The framework validation is done through an open real-time run-time environment that

provides end-to-end real-time performance computing and communication support for hard-real-time applications as well as end-to-end soft performance guarantees for soft real-time and flexible applications.

In summary, the technical objectives of our EPIQ project are: (1) development of application independent QoS and resource management, (2) consideration of QoS dependency among components, and (3) support of end-to-end QoS guarantees which allowing QoS of system components to vary dynamically.

The outline of the report is as follows: In Methodology section we will discuss the top-down methodology approach to our result presentation and evaluation. The Assumption section presents assumptions for our meta-computing framework. In the Procedures section we will describe algorithms, protocols and services within the end-to-end QoS and resource management. The Results and Discussion section shows the implementation and validation of our EPIQ algorithms. The Conclusion section high-lights the most important accomplishments, the educational impact and technology transfer. In Appendix A, we outline changes in personal and contributions of the individual PIs.

Methodology

The report will use the top-down methodology, presenting three important pieces of the EPIQ work, which were done during the funding period of 1997-2002: (a) end-to-end QoS management, (2) resource management, and (c) QoS network management. The top layer of our EPIQ framework consists of the **end-to-end QoS management**, embedded within a middleware system. Here we have researched negotiation protocols, adaptive algorithms, QoS compilation, user-level and application-specific mechanisms to control and gracefully adapt QoS, and their implementations. The next layer below the middleware system is the **QoS-aware resource management**. In this domain we have obtained results in the area of scheduling approaches for hard-real-time as well as soft-real-time applications, and their validation through implementation and experimentation within and on top of Windows NT 4.0 and Windows 2000. The last layer that we will present is the **QoS networking management**, including high-performance predictable networking software as well as QoS routing results.

Assumptions

In our meta-computing framework we make several assumptions to focus the research and receive good understanding for development of end-to-end QoS and resource management. The assumptions are done in the domain of applications, resources and implementation. In the application domain we assume the class of *multimedia* applications for soft-real-time applications such as the distributed visual tracking application and the video-on-demand application, and *periodic* class of applications for hard-real-time applications such as the inverted pendulum control application. In the resource management domain we are considering the CPU and network resources only. We are considering neither memory, nor disk resources in our QoS-aware resource management. In the implementation domain we assume the Windows NT 4.0, Windows

2000 and Linux operating systems for hard-real-time and soft-real-time multimedia applications, Myrinet/ATM hybrid network infrastructure for high-performance predictable networking of distributed hard-real-time applications, and 100 Mbps Ethernet with TCP/IP protocol stack for adaptive distributed multimedia applications.

Procedures

EPIQ QoS and resource management architecture [Hull97], as shown in Figure 1, provides the interfaces, mechanisms and protocols needed to support QoS versus time/resource tradeoffs for flexible tasks in diverse application domains (see Figure 2).

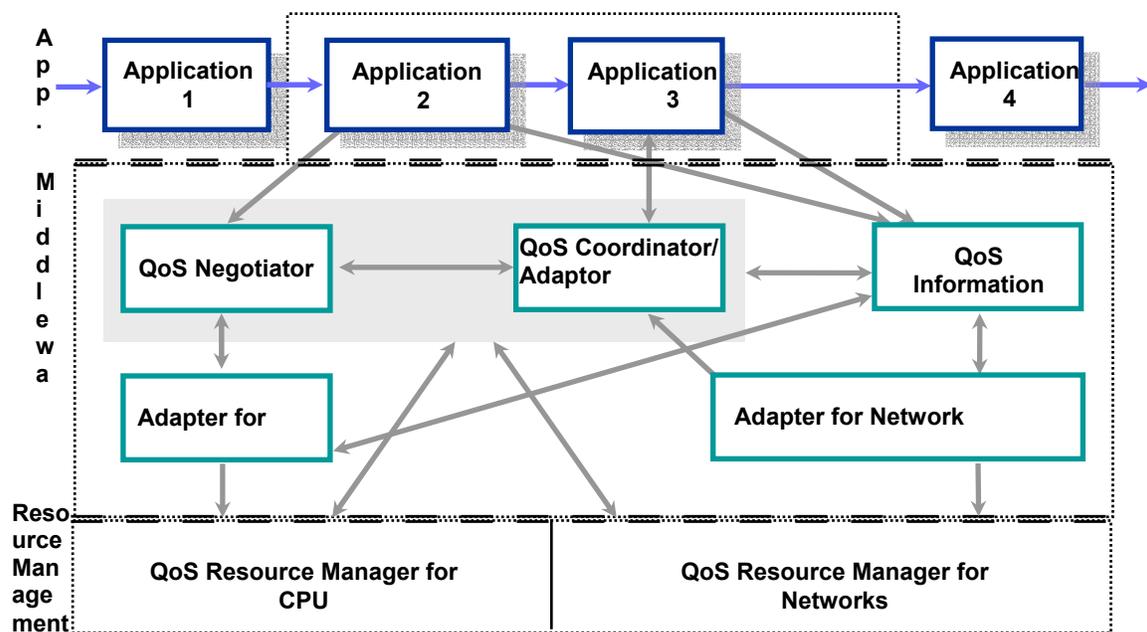


Figure 1: EPIQ Meta-Computing Framework

The architecture is built on a new QoS/resource model [Liu97] as shown in Figure 3. The model differs from existing models in that it captures explicitly the dependency between input QoS, output QoS and resources of dependent system components (i.e., the dependency of output quality of each system component is not only on the resources available to it to produce the output but also on the quality of its input). The issues taken into account include how to support application-specific QoS management policies within a general, application-domain independent framework, how to maintain the overall service quality of an application system while allowing variations in the qualities of services produced by its components, and how to provide users with control in QoS tradeoffs.

End-to-End QoS Management

In the area of end-to-end QoS management we have developed a unique **application model** as shown in Figure 2 that allowed us then to develop the QoS model and individual services in the EPIQ meta-computing framework. We modeled each application as pipelined tasks that have dependencies starting from sources S through various tasks T to destinations U . This model allows us to decompose each application and consider QoS issues at the task level.

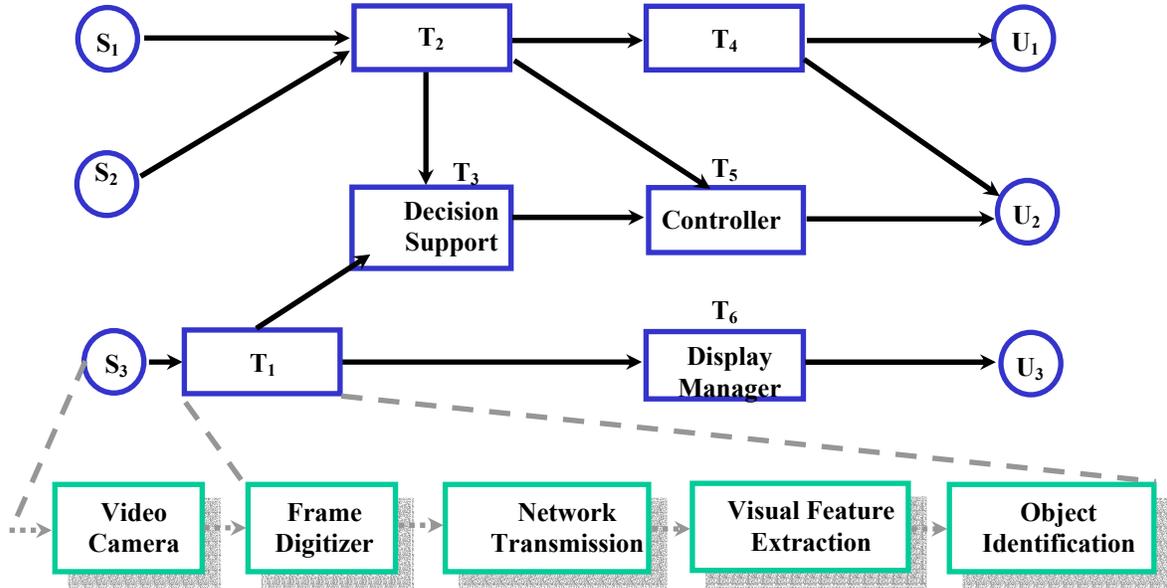


Figure 2: Distributed System of Heterogeneous, Dependent Applications Modeled as Pipelined Tasks

QoS/Resource Model

This application model allowed us then to design a **novel QoS/resource model** per task [Liu97, Hull97] as shown in Figure 3, where each task T gets as an input the input quality $q(i)$ and allocated resources R , and using input QoS and resources, it delivers output QoS $q(o)$. Having this model, we can then start to quantify and reason about end-to-end QoS problems and control of QoS. This QoS/resource model allows us also to consider more complex scenarios. For example, we can have a data stream with an input quality $q(i)$ that arrives at the tradeoff reconciliation component (e.g., transcoder) to transform and partition data stream's input quality $q(i)$ into $q'(i)$ and $q''(i)$ as the data with different input qualities might be needed by two different tasks of a service. After processing of each subtask, they both jointly create a joint output quality $q(o)$. Other scenarios are possible as well. Within the QoS/resource model, we have also investigated representation of QoS parameters, considering single value, pair value representations as well as reward profile representations.

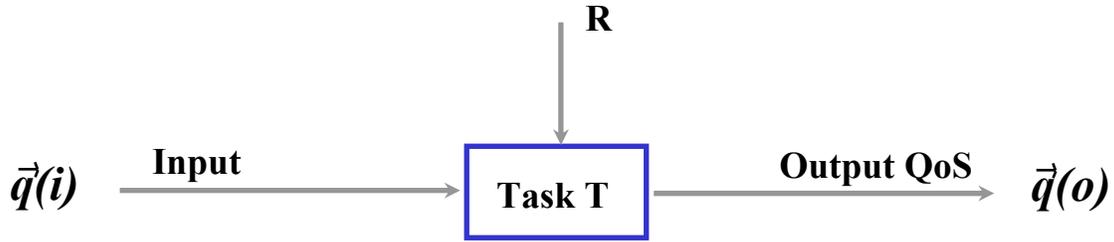


Figure 3: Input and Output QoS for Each Task T

QoS Adaptation Model

Using the task model and the QoS/resource model, we have expanded the task model towards adaptation behavior. We have developed a **control-based task model**, shown in Figure 4, which consists of an application *target task T* with input quality $q(i)$ at first, *monitoring task* that monitors the application task T and its output quality $q(o)$, and the *adaptation task* that uses the monitored values and uses it to adapt in the feedback loop the input quality $q'(i)$ for the application target task T [Li98a, Li98b, Li00c].

Figure 4 shows two target tasks T_1 and T_2 , each with individual adaptive control loop. The loops are concatenated because the monitored output $q(o)$ of T_1 influences the input quality $q(i)$ of task T_2 , and feedback into the adaptive task for final determination of the input quality of task T_2 . The control-based task model for QoS adaptation was further expanded with **fuzzy control model** to include not only QoS parameters adaptation, but also functional adaptation if data adaptation is not sufficient or possible due to scarce resources [Li99a, Li99b, Li99c, Li00c].

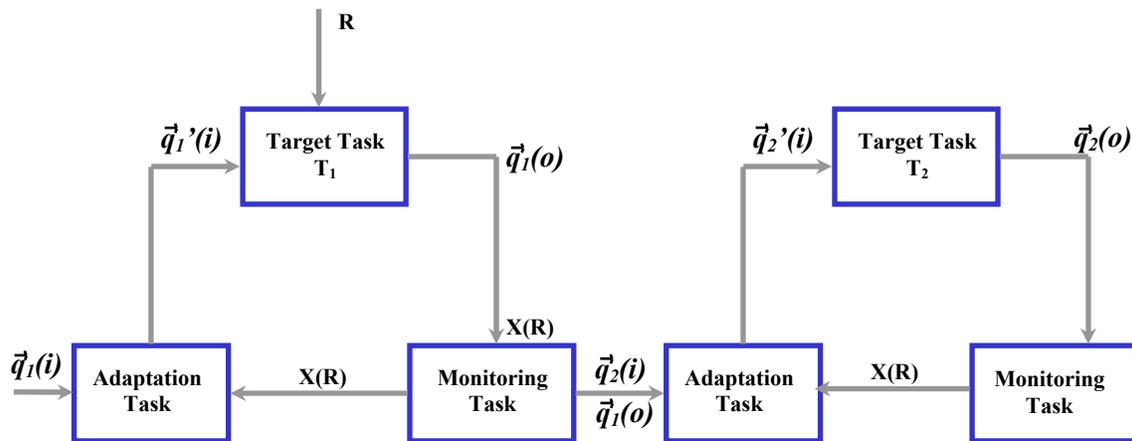


Figure 4: Control-based QoS Adaptation Model

End-to-End Protocols and Services

All of these models were used to develop end-to-end run-time QoS management services within the EPIQ meta-computing framework. Examples of these services are (1) the **end-to-end negotiation protocol**, triggered by the QoS negotiator, (2) end-to-end QoS setup with **discovery and service composition protocols**, (3) **configuration management**

inside the QoS coordinator, (4) **QoS adaptation services and protocols** residing within a QoS adaptor entity, (5) quality aware **scalable service management**, and others [Hull98,Shankar99,Xu00a,Li00b, Li00c, Xu00b, Xu00c, Xu00d, Cui01, Wichadak01, Nahrstedt01, Xu02a,Xu02b,Gu02b, Gu02c].

The EPIQ end-to-end QoS management and corresponding middleware system also utilize a novel off-line support for **QoS programming and compilation** to allow application developers to build easily flexible and QoS-aware applications [Gu00, Nahrstedt00d, Wichadak01, Gu02a,Wichadak02]. Furthermore, the EPIQ QoS management framework was extended to consider not only performance QoS metrics but also security. Of special importance were issues such as (a) protection of QoS parameters and values during QoS negotiation, (b) copyright protection of multimedia data distributed within EPIQ's framework, and (c) real-time encryption of multimedia data. EPIQ's framework allows us to analyze how these three security mechanisms influence end-to-end performance QoS guarantees for flexible applications such as multimedia [Talwar01a, Talwar01b, Prabhu02].

QoS-aware Resource Management

The EPIQ end-to-end resource management mechanisms are built on recent advances in scheduling time-critical applications in open systems and synchronization of end-to-end tasks. Hard real-time systems built on current technologies are closed systems. Whether an application can meet its real-time requirements must be carefully considered. It can be determined only by a global schedulability analysis, based on timing attributes of all tasks in all combinations of applications that may run at the same time. The need for such detailed information often forces all applications in the system to be developed together and limits the configurability of the system. We have developed **EPIQ open system environment** that is designed to allow individual real-time component applications to be developed and validated independently as shown in Figure 5 [Deng97a, Deng97b, Hull98, Deng98, Deng99a, Zhang99a, Zhang99b]. It permits hard real-time applications to run on the same platform with soft-real-time and non-real-time applications. Any real-time application has a simple, but accurate acceptance test and provides each accepted real-time application with timing guarantees regardless the behaviors of other applications in the system.

A key component of the open system is the **two-level hierarchical scheduler**. Each real-time application is executed by a server. The CPU scheduler consists of an OS scheduler, which maintains and schedules the servers, and server schedulers, each of which schedules the threads/tasks in a hard-real-time application according to the scheduling algorithm chosen for the application. The hierarchical scheduler is said to be correct if it accepts the request of any application to execute in the real-time mode only when it can guarantee the schedulability of the application and once it accepts the request, it never violates its guarantee [Deng97a, Deng97b, Frei98, Gardner99].

Furthermore, the open system integrated the hierarchical CPU scheduling framework with the *hierarchical scheme* for **scheduling network traffic** over Myrinet [Zhang99b,

Liu99a]. As the CPU scheduling scheme creates a slower virtual processor for each real-time application running on the processor, the message scheduling scheme creates a slower network for the application. At the lower level, the hierarchical message scheduler uses the weighted round robin scheme to partition the network into slower virtual networks for real-time applications running on the cluster. At the high-level, each high-level scheduler uses an algorithm chosen by the application to schedule the periodic messages of the application. We have evaluated four heuristic algorithms that were designed to schedule periodic messages in bufferless switches according to the traditional one-level scheme used in closed systems. Each of our high-level schedulers uses one of these algorithms to schedule periodic messages in the application, serviced by the scheduler. In order to determine their performance, when used in a hierarchical scheduler, we simulated systems with various numbers of applications, distribution of processors and network loads, etc. To our surprise, we found that the hierarchical scheme using any of the heuristic algorithms at the high level outperforms significantly when compared with the corresponding one-level scheme. A closer examination showed that this is due to the fact that these heuristics usually perform better for small systems and simpler applications and the hierarchical scheme in essence partitions a large system into many smaller ones.

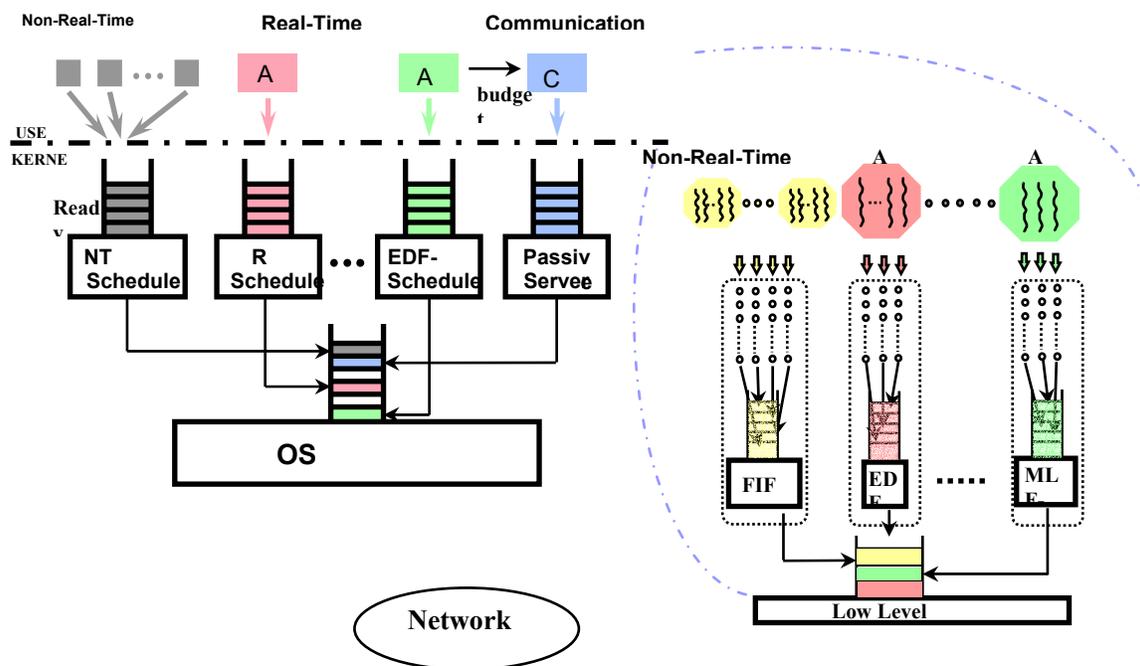


Figure 5: Open Real-Time Systems

Besides the hard-real-time scheduling results within the EPIQ open system environment, we have also designed and evaluated a **Dynamic Soft-Real-Time Scheduling Framework** (DSRT) for scheduling of soft real-time tasks [Viswan00, Yuan01, Yuan02, Gupta02]. Where EPIQ open system environment experimented with kernel implementation of a scheduling framework for hard real-time and non-hard-real-time

tasks, the DSRT environment experimented with user level implementation of a scheduling framework for soft-real-time and non-real-time tasks. DSRT proved to be very useful for multimedia task scheduling. The DSRT system introduced service classes such as the *Periodic Constant Processing Time (PCPT)* class, *Periodic Variable Processing Time (PVPT)* class, *aperiodic* class and *event class*, so that different flexible applications can register with different scheduling class. To implement these classes, DSRT utilized the priority switching mechanism to enforce soft reservation in each class. The DSRT system provides different *soft-reservations* of CPU to meet soft-deadlines. This scheduling service includes also additional services such as the *monitoring* and *probing service* to extract CPU-related parameters for applications to make a reservation with DSRT. Besides the soft-real-time applications, DSRT also allows to run best effort non-real-time applications as it does not use all CPU bandwidth for soft-reservations. The DSRT related services are especially useful for QoS translation between high level QoS specification and system related QoS specifications. Currently, DSRT is being analyzed for multimedia mobile nodes over wireless 802.11 networks to understand the relation between CPU reservation, energy allocation and adaptive hardware [Yuan02].

QoS Network Management

In the area of QoS network management, we have investigated various avenues and problems. Large piece of work within EPIQ meta-computing environment was done on **end-to-end QoS routing** [Chen97a, Chen97b, Chen97c, Chen98a, Chen98b, Chen98c, Chen98d, Chen98e, Chen98f, Chen99a, Chen99b, Lui00a, Lui02a, and Lui02b]. We have investigated various algorithms and developed source routing as well as distributed and hierarchical routing protocols to find a feasible and QoS acceptable path for end-to-end QoS management. Generally, multi-constrained routing is difficult because different constraints can conflict with each other. In particular, the delay-cost constrained routing problem is NP complete. Our novel set of distributed QoS routing algorithms allows for finding an optimized path which satisfies bandwidth, delay jitter and cost quality parameters in polynomial time. Our algorithms are loop-free and utilize only local state information. One class of algorithms is based on probing, the other class is based on distributed recursive computation. Especially noticeable is our *ticket-based distributed QoS routing* that uses a limited amount of tickets to find suitable routes. The algorithm is very close to optimal algorithm, the flooding algorithm, when it comes to success rate to find a QoS suitable route, and it is very close in performance to the optimal algorithm, the Dijkstra shortest path algorithm, when it comes to the message overhead. Another noticeable solution, we have developed, is the efficient *topology aggregation representation and algorithm*, needed when each link carries bandwidth-delay QoS states in large scale networks [Lui00b]. State aggregation is a problem in QoS routing because we need to keep for each path extra QoS state information. Hence, aggregation schemes must be integral part of the QoS network management. We have applied geometrical representation to aggregate multiple QoS value points in a bandwidth-delay plane. Our aggregated approach decreases the space requirements from $O(N)$ to $O(1)$. With aggregation arises a problem of imprecision in QoS representation, and we have developed imprecise network QoS model to work with this representation during QoS routing as well. Our unicast QoS routing algorithms have been further expanded towards

QoS multicasting, providing novel algorithms to find QoS multicast trees for delay-constrained least-cost QoS requirements [Chen00a].

Another large area of interest in the EPIQ network resource management was centered around the design of a **distributed open system architecture** which enables hard real-time applications to be developed and validated independent of each other and configured dynamically to run on a cluster of machines. Especially, our research concentrated on **Control Area Networks (CAN)** [Zhang99a, Zhang99b]. CAN is a high-integrity serial data communications bus that is widely deployed in the area of industrial automation and robotics for decentralized control of field devices. CAN uses the established method known as CSMA/CD but with the enhanced capability of non-destructive bit-wise arbitration to provide collision resolution. The real-time scheduling problem within CAN means to assign identifier to messages with different timing characteristics. There exist several approaches to solve the scheduling problem in CAN which support an event driven communication model, namely the static priority mechanism (e.g., Rate Monotonic Scheduling) and the dynamic priority mechanism (e.g., EDF Scheduling). Both mechanisms are over constraining in a sense that all timing characteristics must be known a priori. This is becoming increasingly difficult to comply with. As the control applications become more and more complex, and application software is modularized, it is unrealistic to assume that a developer has global knowledge of other applications that will be run concurrently in the system while she designs and validates her application. Moreover, modern distributed real-time systems usually require a certain level of flexibility to accommodate on-line reconfiguration. These changes may include adding or removing applications, or changing the timing parameters of existing message streams. The open system architecture addresses these issues by providing isolation between real-time applications and hence allowing them to be designed and validated independently. This work proposed the use of sporadic servers to arbitrate and preserve bandwidth for each node in CAN. Messages in the sporadic server are scheduled using the EDF algorithm. Each sporadic server is associated with a budget and period. Every node connected to the CAN bus keeps a table of all the real-time sporadic servers in the system. Contention is resolved implicitly by scheduling the sporadic servers as periodic messages using the EDF algorithm. Each node keeps two message queues for aperiodic (with soft deadline) and non-real-time traffics respectively. Whenever a transmission slot is not used by hard real-time messages, the nodes contend for channel access by encoding a fixed priority level (using the rate monotonic algorithm) into the message identifier. The winner node then transmits a message from the head of its aperiodic message queue. When only non-real-time traffic is present, round-robin is used to facilitate fair channel access. Simulation study has been done to show the proposed hierarchical scheduling scheme allows higher network utilization as well as providing timing isolation among real-time applications

We have also considered the **high-performance predictable network software** and agile objects. The EPIQ networking software provides support for QoS management as well as expands the utility of high-speed networks by improving usable bandwidth and predictability of performance. This work is built on the novel software architecture of Illinois Fast Messages (FM), which can deliver network link bandwidth to the application

for small packets, and the use of small packets in turn enables the system to provide predictable performance. FM delivers latency of on the order of milliseconds and peak bandwidth of approximately 76 Mbytes a second to applications. FM-QoS extends FM to deliver predictable performance (e.g., deterministic latencies and guaranteed bandwidth) for very high speed cluster interconnects [Connelly97]. It exploits a novel communication architecture based on the network feedback in wormhole routed networks. Feedback is used to enforce loose synchronization, which when combined with self-synchronizing schedules, can avoid resource conflicts for network links and outputs. Elimination of such conflicts leads to predictable communication performance. Key elements of the architecture include feedback-based synchronization (FBS) of senders and a class of self-synchronizing communication schedules for which FBS is effective. A Petri nets model is used to precisely characterize the structure of self-synchronizing schedules and to analyze the clock drifts that can be tolerated. This analysis indicates that our architecture can share network resources with predictable performance at granularities of a few microseconds, which is orders of magnitude better than previous software-based schemes and comparable to hardware-intensive approaches such as virtual circuits (e.g., ATM).

The real-time scheduling framework from the EPIQ distributed open system environment was further applied to **IEEE 802.11 wireless network**. Energy constraint has become an important factor in the design of MAC protocols. This work addresses the issue of providing deterministic timing guarantees in the wireless LAN while minimizing energy consumption of the wireless nodes. We have designed a centralized energy efficient algorithm, called *Scheduled Contention Free Burst (S-CFB)*, which is built upon the recently proposed Hybrid Coordination Function (HCF) in the IEEE 802.11 standard. The key idea of this scheme is to bundle message transmissions into multiple contention free bursts in order to reduce control overhead and allow wireless nodes to sleep whenever possible. S-CFB exploits the flexibility provided by the HCF standard, and combines the EDF and Weighted-Round-Robin algorithm to balance between power consumption and bandwidth utilization, while still providing hard timing guarantee for real-time messages. Our performance studies have shown that : 1) S-CFB meets the timing requirement for real-time traffic; 2) allows wireless nodes to switch to idle state for a longer period of time; and 3) reduces the number of control frames required to maintain contention free transmission.

Moreover, at the end host, we have designed a **generalized communication server** to provide a generic resource management mechanism [Zhang99b, Viswan00]. (A communication server was proposed in the distributed open system environment to manage the real-time connections in the open system. Messages generated by each application are scheduled and transferred as though they are in a slower virtual network.) The objective is to provide a uniform mechanism for global resource sharing among tasks in different applications with bounded priority inversion time. A special server, called *global resource server*, is used to manage CPU budget in servicing tasks in global critical section (i.e. tasks accessing a global resource). Whenever a task requests for a global resource, the task is moved to the ready queue of the critical resource server. When the task finishes its critical section, it will be moved back to the ready queue of its original application server. The global resource server keeps track of CPU budget consumed by the task and subtract it from the application server. The global resource server execute at the highest priority of all the tasks blocked on the resource. The correctness of this mechanism has been proved and a corresponding schedulability test provided.

Results and Discussions

The EPIQ project achieved a large set of implementations and experimentations. We will enumerate some of them and refer to more details in publications, listed in the bibliography.

End-to-End QoS Management

We have implemented several end-to-end QoS management middleware versions to test the various approaches as they evolved over the duration of the funding:

1. Our first version of the **end-to-end QoS management** was implemented in Java. It included end-to-end negotiation protocols, and the task and QoS/resource models. Whereas the preliminary version was done to prove the feasibility and practicality of our concepts, in the design of the next version we have paid special attention to efficiency, flexibility and scalability [Shankar99].
2. Our second version of the end-to-end QoS management was implemented in Java/C++/CORBA, validating the adaptive control-based task model as well as fuzzy control to provide QoS guarantees to a distributed visual tracking application. This middleware system was called **Agilos** and included QoS negotiator, and QoS adapter [Li98c, Li98d, Li99b, Kalter00a, Li00c]. It also relied on CORBA middleware services such as event service, trader service, naming service and others. We have demonstrated the feasibility of this middleware together with the open system resource management at the DARPA meeting in Washington DC, 1999. The DARPA demonstration showed that using the open system scheduling environment we could run a hard-real-time application, the inverted pendulum, as well as the end-to-end QoS management system Agilos with the adaptive visual tracking application. Figure 6 shows the QoS adaptation results of the flexible visual tracking application under network bandwidth and CPU resource variability. The results show that if we want to preserve the critical quality of the visual tracking application, the *tracking precision*, then other quality parameters will suffer and need to be degraded such as the frame size and/or frame rate.

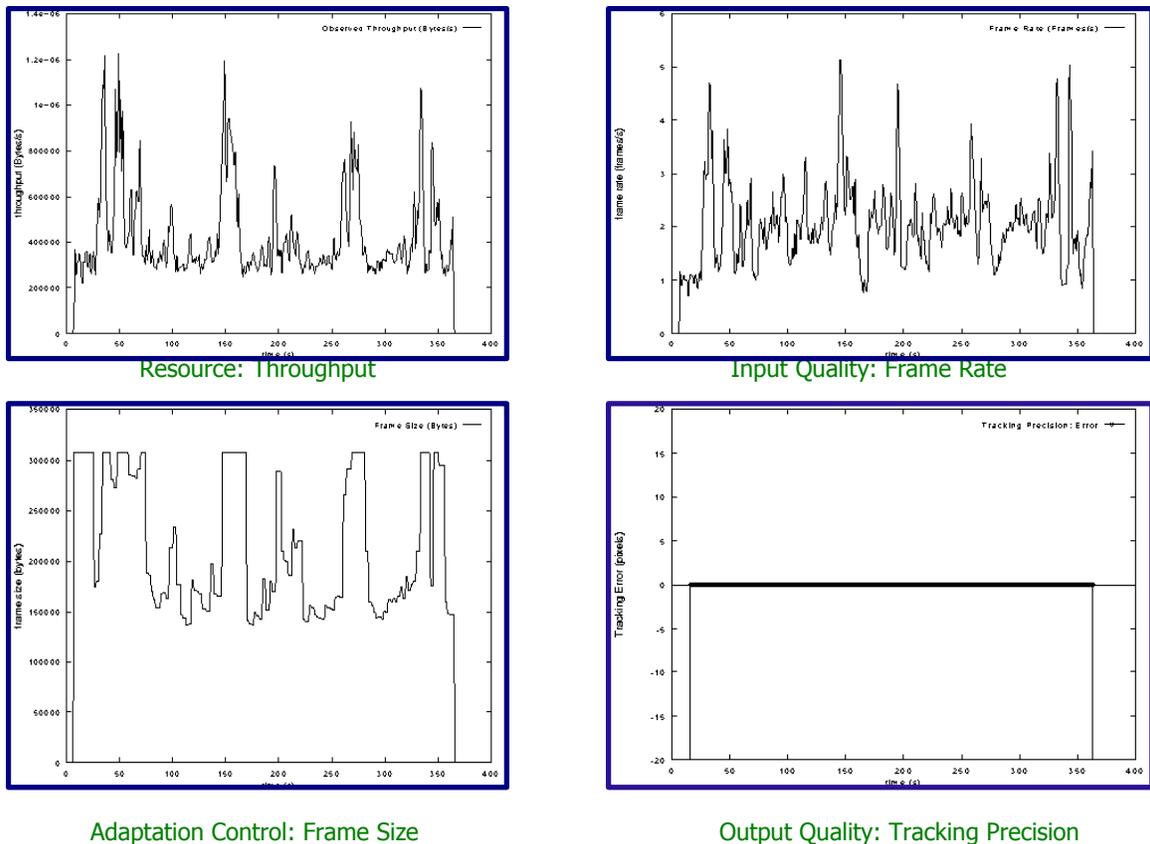


Figure 6: EPIQ Middleware In Action Running Visual Tracking

3. Our third version of the end-to-end QoS management system, called **2KQ+**, was implemented in Java/C++/CORBA and tested the service management services and protocols such as QoS-aware discovery protocols to discover appropriate services in the end-to-end path, and QoS-assured service composition for flexible applications. This middleware system utilized the DSRT scheduling framework in the underlying resource management as well as network service broker for bandwidth management [Nahrstedt00d, Gu00, Cui01, Xu01, Wichadak01, Xu02a, Wichadak02, Gupta02].
4. The end-to-end QoS management implementation, 2KQ+, was accompanied by the implementation of tools to allow for QoS programming and compilation [Gu00, Gu01a, Gu02a, Nahrstedt00, Wichadak01, Wichadak02]. We have developed the tool, called **QoS Talk**, that represents a visual programming environment for an application developer. The underlying language, into which application's visual graph is translated, is called HQML, and it belongs to the XML language class. HQML representation allows the QoS Talk tool to check for consistency of QoS specification. Another tool we have developed is called **QoS compiler** that takes the high level application programming graph with QoS specifications and translates it to system and resource specific representations. QoS compiler takes sound HQML representations, as well as application configurations from the QoS Talk and translated them into service component graphs where each component is associated with system-related QoS parameters. This middleware and their tools are currently part of the distributed operating

system Gaia that runs in our smart room spaces. The 2KQ+ services support various multimedia services in the smart room environment.

5. As part of the 2KQ+ middleware, we have implemented **security services** such as real-time encryption, authentication, and developed the concept of quality of security [Talwar01a, Talwar02]. This middleware testbed allowed us to investigate the tradeoffs between performance and security in mobile multimedia applications such as mobile audio and video. We have tested standard implementations such as DES and triple DES and their delay impact on multimedia transmission. The results are encouraging as many implementations are being done very efficiently.

QoS-aware Resource Management

We have implemented several versions of the resource management to support resource scheduling for hard-real-time, soft-real-time and non-real-time applications.

1. We have implemented the **open system** on the Windows NT 4.0 platform as well as on Linux. We have thoroughly tested and debugged our uniprocessor open system prototype and tested it with the inverted pendulum application to demonstrate its capability and performance. We have presented the EPIQ open system architecture and prototype at Sony Distributed System Lab, HP Labs and Microsoft for the purpose of technology transfer. The fact that our prototype on Windows NT requires only 2172 additional lines of C code for both the kernel extension and Real-Time API functions is a proof that the important open-system capability can be easily obtained within the framework of any operating system. It does not create any backward compatibility problem and leads to insignificant amount of performance degradation [Deng97a, Deng97b, Deng98, Liu99a, and Shih00].
2. We have implemented the **dynamic soft-real-time scheduling (DSRT)** framework on Windows NT, 2000, Solaris and Linux platforms. We have tested this framework with many different multimedia applications ranging from visual tracking, video on demand, video-phone applications to mobile video applications. We have also augmented DSRT and tested it on HP laptop under Linux where the processor changes speeds depending on energy consumption [Yuan01, Yuan02].
3. In our effort to enhance the suitability of Windows NT for hard-real-time applications, we have implemented two tools. The first tool is a **user-level set priority function**, implemented as a device driver, that does constant ration mapping of user thread priorities chosen by the application to one of the 16 real-time priority levels supported by Windows NT. The application calls this function, rather than NT's setThreadPriority() to set thread priority. We have also implemented a library function that approximates the **non-preemptive critical section protocol**. It can be used with mutexes and reader/write locks in Windows NT to control priority inversion [Deng98, Shih00].

QoS Network Management

In the QoS network resource management, we have concentrated on different ways to validate our results ranging from real implementation to simulation studies.

1. We have implemented the EPIQ open system where the CPU scheduling is integrated with the **communication server** within the open environment. The communication server uses Illinois Fast Messages on Myrinet to connect 4 PCs to a cluster. Figure 7 shows performance comparison between open and closed systems when using different scheduling algorithms [Zhang99a, Zhang99b].

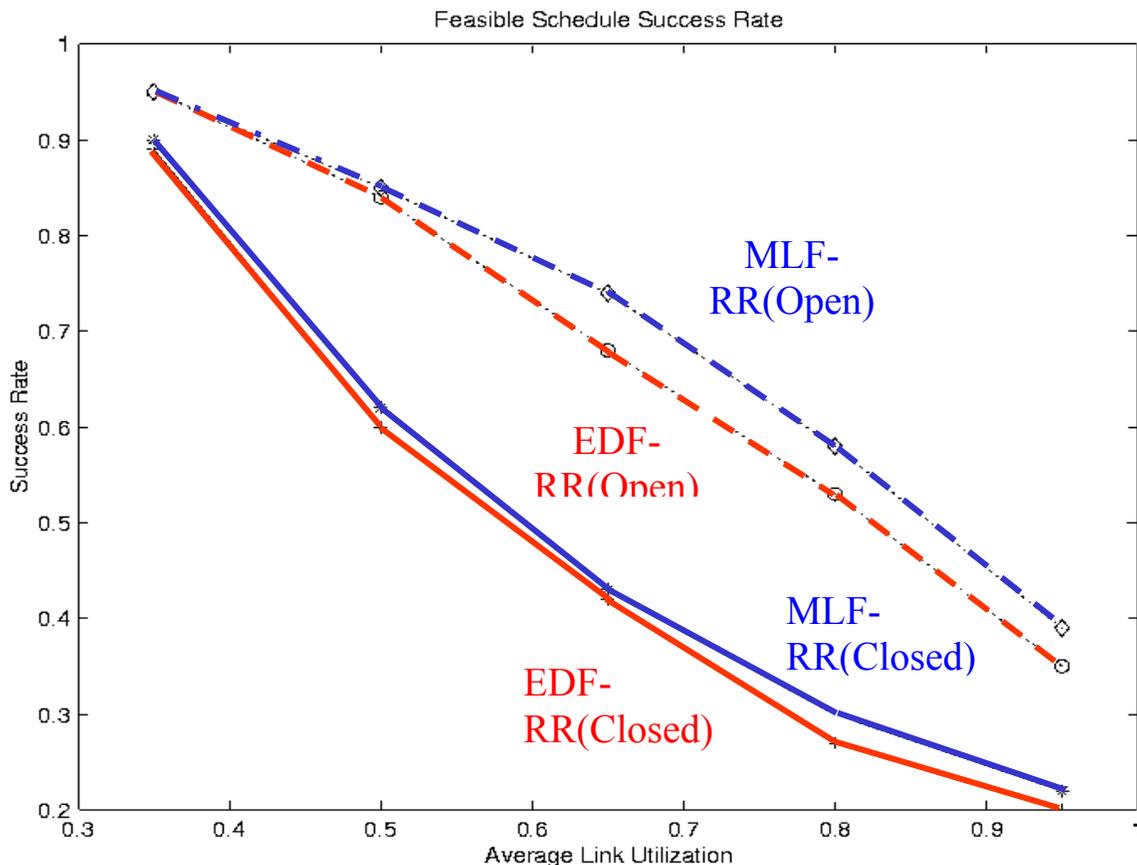


Figure 7: Performance Comparison (Feasible Schedule Success Rate) between EPIQ Open System and a Closed System.

2. We have implemented a simple version of the **edge device** on PCs to connect ATM network cloud with the RSVP/IP enabled ISP (Internet Service Provider) environment where our end-points are connected [Chawla99a, Chawla99b]. The edge devices provide QoS bridge for heterogeneous networks. Specifically, we have implemented QoS translations between RSVP flows and CRB traffic class specifications, and end-to-end connection setup protocol which takes into account that RSVP is receiver-oriented protocol and ATM uses a sender-oriented signaling protocol. Moreover, we have conducted experiments between end-

- points across laboratories. The results showed that the end-to-end establishment protocol over heterogeneous networks works correctly and efficiently. Furthermore, the video transmission when conforming to the negotiated and established QoS contract performs well.
3. We have implemented a **virtual network testbed** on PCs under Windows NT. Using this testbed we have implemented various **QoS routing** protocols such as the source QoS routing, and several distributed QoS routings algorithms. We have also simulated various QoS routing algorithms on different Internet topologies to verify our algorithms [Lewites99].
 4. We have developed the multi-network **Fast Messages** substrate and demonstrated it over the Myrinet/ATM hybrid network infrastructure. FM-QoS is implemented on Myricom's Myrinet Network (a 1.2 Gbps cluser network). We have also implemented a router (bridge) of FM which allows FM on Myrinet to be tunneled through virtual circuits in an ATM network [Connelly97].

Conclusion

In conclusion we will summarize (a) our accomplishments through the entire EPIQ project and highlight conceptual and experimental results of this metacomputing framework, (b) the educational impact of the DARPA funding and (c) technology transition efforts.

Accomplishments

1. We have designed and implemented a set of robust versions of end-to-end QoS management systems for hard-real-time applications, soft-real-time applications and non-real-time application. The QoS management systems are able to accommodate a mixture of applications, and manage different dimensions and measures of QoS, taking into account our new QoS/resource model.
2. We have successfully implemented and evaluated set of end-to-end exploration protocols for QoS setup such as the end-to-end negotiation protocol, discovery protocol, end-to-end reservation protocol. Using these protocols we are able to establish an end-to-end connectivity with feasible quality levels at end-points for our distributed applications.
3. We have successfully designed the adaptive QoS framework and validated it with Agilos adaptive QoS middleware system and visual tracking application implementations. This work received the IEEE Communications Society Leonard Abraham Price, awarded in June 2000.
4. Based on Agilos, we have designed next version of end-to-end QoS middleware system, 2KQ+, that allows hand-held devices and mobility. The middleware is based on the middleware kernel, called SMART, which is designed as an onion. A service kernel represents the core services, required at each device, and it is then expanded through enhancement services, present at more computationally powerful devices.
5. We have applied the early QoS/resource model and task model to QoS specification and the component-based service model to allow for configurable and exchangeable services. This work resulted in the development of the QoS

- Talk programming environment for the 2KQ+ middleware system, QoS Compiler and run-time multimedia service setup model.
6. We have enhanced 2KQ+ with dynamic end-to-end service configuration and distribution capabilities. We have also added security considerations where access to any multimedia service and middleware QoS service is authenticated and the data are protected via real-time encryption.
 7. We have successfully demonstrated integration of the EPIQ open system and the Agilos adaptive QoS middleware.
 8. We have successfully investigated the interrelation of the adaptive control algorithms applied to the CPU adaptor and network bandwidth adaptor within the Agilos adaptive QoS middleware.
 9. We have expanded the Agilos adaptive QoS by introducing gateway capabilities to provide load balancing, and functional adaptations such as service switching. We have explored adaptive fuzzy control mechanisms and appropriate rules in gateway middleware system to support multiple users and multiple video cameras within the distributed visual tracking application that used Agilos.
 10. We have developed EPIQ open system scheduling framework based on hierarchy.
 11. We have thoroughly tested and debugged the EPIQ open environment, which is implemented on the Windows NT platform. Its soundness was demonstrated with the inverted pendulum controller.
 12. The EPIQ open system was ported to Linux OS, Version 5.0.36. We have implemented the rate monotonic real-time scheduling framework on this platform.
 13. We have expanded the EPIQ open system environment on Linux with the QoS-aware network service broker. The network service broker was designed to decouple the bandwidth-scheduling problem from the CPU scheduling and hide the networking details from the application. Through simulation studies we have also showed that the hierarchical scheduling scheme, compared with one level scheduling, allows higher network utilization and timing isolation among real-time applications.
 14. We have implemented the EPIQ distributed open system within the Myrinet cluster PCs running Windows NT OS.
 15. We have implemented DSRT scheduling framework on Windows NT and 2000, and integrated it with 2KQ+ middleware system.
 16. We have enhanced DSRT towards considerations of adaptive hardware. We can currently support CPU-related quality requirements for adaptive multimedia applications under adaptive CPU.
 17. We have designed and tested via simulations and prototypes several important QoS unicast routing algorithms: (a) ticket-based distributed routing, (b) source-based QoS routing, (c) hierarchical QoS algorithms with efficient topology aggregation algorithms.
 18. We have completed extensive experiments with edge devices to interconnect heterogeneous networks such as ATM, Myrinet and RSVP/IP to achieve QoS-aware connectivity.

Educational Impact

This grant achieved not only incredible amount of research results, but it also had a tremendous impact on **education**. It funded fully or partially 26 students during the duration of the grant. From the set of DARPA funded students eight PhD students finished with PhD (Z. Deng, A. Shankar, D. Hull, M. Gardner, S. Chen, B. Li, D. Xu. K.S. Lui) and nine MS students finished with MS degrees. Many undergraduate students benefited from the research experiences as well as they worked together with PhD or MS students side by sides and helped with various experiments. The PhD student, B. Li, received the IEEE Communications Society Leonard Abraham Price for control-based QoS adaptation model. Two MS students, X. Gu and V. Talwar, received the best Master Thesis Kuck Award, given by the Department of Computer Science at UIUC, for their MS theses. The MS student, D. Gupta, received the highly competitive Siebel scholarship for his achievements. Professor K. Nahrstedt received the Ralph and Catherine Fisher Professorship in 2002. This professorship is given by the Engineering College at UIUC to junior faculty members for their excellent research results.

Technology Transition

The **Agilos adaptive QoS middleware system** architecture and prototype were presented to IBM and NASA Ames Research Lab for the purpose of technology transition. Agilos and the distributed visual tracking application were installed in NASA AMES and extensive wide-area experiments were conducted over NASA high-speed network testbed [Li98c, Nahrstedt99].

The **QoS programming environment and QoS compiler** were presented at the Open System Standard committee meeting and were received with great interest to consider many concepts in their end-to-end solution. Furthermore, NASA is interested to test their applications via our QoS programming environment and have it translated into system specific representations via our QoS compiler.

The **2KQ+ middleware system**, the QoS Talk and QoS compiler are part of the distributed operating system, Gaia, that runs in smart rooms in the current Department of Computer Science at UIUC. The EPIQ's results represent the core of QoS support for this operating system. The final goal is to run Gaia system in each conference/seminar room within our new Siebel Center, the new Computer Science Building.

The **EPIQ open system architecture** and prototype were presented at Sony Distributed Systems, HP Labs, and Microsoft for the purpose of technology transition. The EPIQ open system kernel, the corresponding API code, and the embedded inverted pendulum controller software were transferred to Microsoft [Liu99a].

The **DSRT system** was transferred to the ETL research laboratory, Tsukuba, Japan, to support configurable OS platforms. Furthermore, many other research labs and universities are using the DSRT system for multimedia scheduling as the system runs on multiple platforms (Windows, Solaris, Linux) [Nahrstedt00b].

Appendix A

The credit for the overall EPIQ framework and research goals goes to all PIs and their students. Professors J. Liu and K. Nahrstedt with their students worked jointly on the QoS/resource model and its relation to the task model.

Professor J. Liu and her students worked on the (a) first version of the Java-based end-to-end QoS management with end-to-end negotiation capabilities, (b) EPIQ open system with the hierarchical scheduling framework, allowing to schedule mixture of hard-real-time and non-hard-real-time applications, (c) EPIQ distributed open system with the communication server, communication broker and scheduling algorithms for communication scenarios, (d) CAN problems, and (e) scheduling problems in 802.11 networks. (Professor J. Liu moved from UIUC to Microsoft in 1999.)

Professor A. Chien and his students worked on the (a) Fast Messaging System, (b) on FM-QoS framework over Myrinet, and (c) QoS-aware interconnectivity between Myrinet and ATM. (Professor A. Chien moved from UIUC to UC San Diego in 1998.)

Professor K. Nahrstedt and her students worked on the (a) adaptive QoS model, based on control theory and fuzzy control, (b) adaptive QoS middleware system, Agilos, and its extensions towards CPU/bandwidth integration, gateway enhancement, and building of an extensive visual tracking application to validate Agilos, (c) hybrid QoS middleware system, 2KQ+, with support of soft-guarantees as well as QoS adaptation, advanced QoS setup protocols, discovery protocols, end-to-end reservation, QoS adaptation, service composition, and monitoring, (d) Dynamic Soft-Real-Time (DSRT) system for scheduling of soft-real-time and non-real-time tasks, (e) QoS routing problems in the unicast and multicast domain, (f) QoS-aware interconnectivity between ATM and RSVP/IP networks, (g) QoS programming and QoS compilation for multimedia applications, and (h) tradeoffs between performance and security QoS metricsBibliography.

1997 Publications, Theses and Talks:

- [Lui97] J.W.S. Liu, K. Nahrstedt, D. Hull, S. Chen, B. Li, "EPIQ QoS Characterization", Draft Technical Report 1997, <http://epic.cs.uiuc.edu>
- [Deng97a] Z. Deng, J.W.S. Liu, J. Sun, "A Scheme for Scheduling Hard Real-Time Applications in Open System Environment", 9th Euromicro Workshop on Real-Time Systems, June 1997, pp. 191-199.
- [Connelly97] K. Connelly, A. Chien, "FM-QoS: Real-Time Communication Using Self-Synchronizing Schedules", Supercomputing'97, November 1997.
- [Nalini97] N. Venkatasubramanian, K. Nahrstedt, "An integrated Metric for Video QoS", ACM Multimedia, 1997, Seattle, WA, pp. 371-381
- [Chen97a] S. Chen, K. Nahrstedt, "Distributed QoS Routing", Technical report, UIUCDCS R-97-2017, CS Department, UIUC, October 1997.
- [Chen97b] S. Chen, K. Nahrstedt, "Routing by Distributed Recursive Computation and Information Reuse", Technical report, UIUCDCS-R-97-2028, CS Department, UIUC, October 1997.
- [Hull97] D. Hull, A. Shankar, K. Nahrstedt, J. W. S. Liu, "An End-to-End QoS Model and Management Architecture", IEEE Workshop on Middleware for Distributed Real-Time Systems, December 1997.
- [Deng97b] Z. Deng, J.W.S. Liu, "Scheduling Hard-Real-Time Applications in Open Environment", IEEE Real-Time Systems Symposium, December 1997.
- [Chen97c] S. Chen, K. Nahrstedt, "On Finding Multi-constrained Paths", Technical report, UIUCDCS-R-97-2026, CS Department, UIUC, October 1997.

1998 Publications, Theses and Talks:

- [Deng98] Z. Deng, J.W.S. Liu, L. Zhang, A. Frei, M. Seri, "An Open Environment for Real-Time Applications", Real-Time Systems Journal, 2002.
- [Hull98] D. Hull, "An End-to-end Imprecise Computation Environment", PhD Thesis, December 1998, University of Illinois at Urbana-Champaign.
- [Frei98] A. Frei, "Scheduling of DPC and LPC to Enhance Timing Preditability of Windows NT", Master Thesis, December 1998, University of Illinois at Urbana-Champaign.
- [Li98a] B. Li, K. Nahrstedt, "An Open Task Control Model for Quality of Service Adaptation", 14th International Conference on Advanced Science and Technology (ICAST 98), April 1998, Chicago, IL, pp. 29-41
- [Chen98a] S. Chen, K. Nahrstedt, "Max-min Fair Routing in Connection-Oriented Networks", Euro-Parallel and Distributed Systems Conference, July 1998, Vienna, Austria.
- [Chen98b] S. Chen, K. Nahrstedt, "Distributed Routing with Imprecise State Information", IEEE 7th International Conference on Computer Communication and Networks (ICCCN), October 1998, Lafayette, LU.
- [Chen98c] S. Chen, K. Nahrstedt, "Distributed Quality-of-Service Routing in High-Speed Networks Based on Selective Probing", IEEE 23rd Annual Conference on Local Computer Networks (LCN), October 1998, Boston, MA, pp. 80-89.
- [Chen98d] S. Chen, K. Nahrstedt, "Distributed Quality of Service Routing with Imprecise State Information for Next Generation Internet", NASA NREN QoS Workshop, August 1998, NASA, CA.

- [Li98b] B. Li, K. Nahrstedt, "A Control-Theoretical Model for Quality of Service Adaptation", IFIP International Workshop on QoS (IWQoS), May 1998, Napa, CA.
- [Li98c] B. Li, K. Nahrstedt, "Adaptive QoS Middleware Framework for Complex Flexible Applications", NASA NREN QoS Workshop, August 1998, NASA, CA.
- [Li98d] B. Li, D. Xu, K. Nahrstedt, J.W.S. Liu, "End-to-end Support for Adaptive Applications Over the Internet", SPIE Symposium on Voice, Video and Data Communications, November 1998, Boston, MA.
- [Xu98] D. Xu, B. Li, J.W. Liu, K.Nahrstedt, "Providing Seamless QoS for Multimedia Multicast in Real-Time Packet Cellular Networks", SPIE Symposium on Voice, Video and Data Communications, November 1998, Boston, MA.
- [Chen98e] S. Chen, K. Nahrstedt, "An Overview of Quality of Service Routing for the Next Generation High-Speed Networks: Problems and Solutions", IEEE Network, Special Issue on Transmission and Distribution of Digital Video, November/December 1998, Vol.12, No.6, pp. 64-79 (received from IEEE Communications Society Best Tutorial Award).
- [Chen98f] S. Chen, K. Nahrstedt, "On Finding Multi-constrained Paths", IEEE International Conference on Communication (ICC), 1998, Atlanta, GA.
- 1999 Publications, Theses, and Talks:**
- [Deng99a] Z. Deng, J. W.-S. Liu, L.Y. Zhang, M.. Seri, and A. Frei, "[An open environment for real-time applications](#)," Real-Time Systems Journal, 16(2):155-186, May 1999.
- [Gardner99] M. K. Gardner and J. W. S. Liu, "[Performance of Algorithms for Scheduling Real-Time Systems with Overrun and Overload](#)," In Proceedings of the Eleventh Euromicro Conference on Real-Time Systems, 9-11 June 1999, University of York, York, England.
- [Zhang99a] L.Y. Zhang, Z. Deng, I. Philp, and J. W. S. Liu, "[A hierarchical scheme for scheduling messages in open real-time environment](#)", In Proceedings of IEEE Symposium on Real-Time Systems, December 1999.
- [Shankar99] M. Shankar, M. DeMiguel, J.W.S. Liu, "An Application Domain-Independent QoS Management Architecture" IEEE Symposium on Real-Time Applications and Systems (RTAS), June 1999pp. 176-189.
- [Zhang99b] L. Zhang, J.W.S. Liu, "An Open Real-Time Environment on PC Clusters", 11th Euromicro Workshop on Real-Time Systems, Work in Progress Section, June 1999.
- [Liu99a] J.W.S. Liu, "EPIQ Open Environment for Real-Time Applications", Invited Talk at Sony Distributed Systems Lab, San Jose, CA, March 1999; HP Research Laboratories, Palo Alto, CA, March 1999; Microsoft, Seattle, WA, April 1999.
- [Chen99a] S. Chen, K. Nahrstedt, "Routing by Distributed Recursive Computation and Information", IEEE International Performance, Communication and Control Conference (IPCCC), February 1999, pp. 393-403.
- [Chen99b] S. Chen, "Routing Support for Providing Guaranteed End-to-End Quality of Service", April 1999, PhD Thesis, University of Illinois at Urbana-Champaign.
- [Nahrstedt98] K. Nahrstedt, "Coexistence of QoS and Best Effort Flows: Routing and Scheduling Analysis", Invited talk to Ohio State University, December 1998
- [Nahrstedt99a], K. Nahrstedt, "Adaptive QoS Framework and its Application to Visual Tracking", Invited Talk, Purdue University, March 1999; Polytechnic University, New York, March 1999, University of California, Irvine, January 1999.

- [Li99a] B. Li, K. Nahrstedt, "Optimal State Prediction for Feedback-Based QoS Adaptations", IFIP/IEEE International Workshop on Quality of Service (IWQoS), June 1999, London, England.
- [Li99b] B. Li, K. Nahrstedt, "Dynamic Reconfiguration for Complex Multimedia Applications", IEEE International Conference on Multimedia Computing and Systems (ICMCS), June 1999, Florence, Italy.
- [Chen99c] S. Chen, K. Nahrstedt, "Hierarchical Scheduling for Multiple Classes of Applications in Connection-Oriented Integrated Service Networks", IEEE International Conference on Multimedia Computing and Systems (ICMCS), June 1999, Florence, Italy.
- [Chawla99a] M. Chawla, Y. Zhou, K. Nahrstedt, "QoS Translation and End-to-End Signaling Protocols for RSVP over CBR/ATM", OPNETWORK'99, August 1999, Washington, DC.
- [Chen99d] S. Chen, K. Nahrstedt, "Distributed Quality of Service Routing in Ad-Hoc Networks", IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Ad-Hoc Networks, Vol. 17, No.8, August 1999, pp. 1-18.
- [Chawla99b] M. Chawla, "Design, Implementation and Evaluation of an Edge Device Architecture", August 1999, Master Thesis, University of Illinois at Urbana-Champaign.
- [Lewites99] S. Lewites, "QuoSAR: Implementation Testbed for QoS Routing", June 1999, Master Thesis, University of Illinois at Urbana-Champaign.
- [Li99c] B. Li, K. Nahrstedt, "A Control-based Middleware Framework for Quality of Service Adaptation", IEEE JSAC, Vol. 17, No. 9, September 9, pp. 1632-1650. (received IEEE Communications Society Leonard C. Abraham Paper Award).
- [Nahrstedt99b] K. Nahrstedt, D. Wichadakul, "QoS-aware Active Gateway for Multimedia Communication", 6th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS), October 1999, Toulouse, France.
- 2000 Publications, Theses and Talks:**
- [Shih00] Ch. Shih, J. Liu, J. Qian, M. Jonnalagadda, and J. Li, "[Open Real-time Linux](#)," 2nd Real-time Linux Workshop, Orlando, Florida, 2000.
- [Chen00a] S. Chen, K. Nahrstedt, Y. Shavitt, "A QoS-aware Multicast Routing Protocol", IEEE INFOCOM, March 2000, Israel.
- [Li00a] B. Li, W. Jeon, B. Kalter, K. Nahrstedt, J. Seo, "Adaptive Middleware Architecture for a Distributed Omni-directional Visual Tracking System", SPIE Multimedia Computing and Networking Conference (MMCN) January 2000, San Jose, CA.
- [Nahrstedt00a] K. Nahrstedt, "QoS Compilation and Runtime Systems", Invited Talk to Purdue University, November 2000.
- [Nahrstedt00b] K. Nahrstedt, "Adaptive QoS Framework and its Application to Visual Tracking", Invited talk to Technical University, Karlsruhe, Germany, April 2000; ETL Laboratory, Tsukuda, Japan, March 2000.
- [Nahrstedt00c] K. Nahrstedt, "Control-based Adaptive QoS Framework", Invited Talk to Tokyo University of Engineering, Tokyo, Japan, March 2000.
- [Xu00a] D. Xu, D. Wichadakul, K. Nahrstedt, "Resource-aware Middleware for Active and Configurable Distributed Services", Active Middleware Services, Eds. Salim Hariri, Craig. A. Lee, Cauligi S. Raghavendra, Kluwer Academic Publishers, 2000, pp. 167-176.
- [Li00b] B. Li, K. Nahrstedt, "QualProbes: Middleware QoS Profiling Services for Configuring Adaptive Applications", in Lecture Notes in Computer Science, ACM-

Springer, Vol. 1795, Eds. J. Sventek, G. Coulson; Also in proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000), April 2000.

[Chen00b] S. Chen, K. Nahrstedt, Y. Shavitt, "A QoS-Aware Multicast Routing Protocol", IEEE JSAC, Vol.18, No. 12, December 2000.

[Lui00a] K. Lui, K. Nahrstedt, S. Chen, "Hierarchical QoS Routing in Delay-Bandwidth Sensitive Networks", IEEE International Conference on Local Computer Networks (LCN), November 2000.

[Lui00b] K. Lui, K. Nahrstedt, "Topology Aggregation and Routing in Bandwidth-Delay Sensitive Networks", IEEE Globecom'2000, November 2000.

[Talwar00] V. Talwar, K. Nahrstedt, "Securing RSVP for Multimedia Applications", ACM Multimedia Security Workshop, November 2000, Los Angeles, CA.

[Nahrstedt00d] K. Nahrstedt, D. Wichadakul, D. Xu, "Distributed QoS Compilation and Runtime Instantiation", IFIP/IEEE International Workshop on QoS (IWQoS), June 2000, Pittsburgh, PA.

[Xu00b] D. Xu, D. Wichadakul, K. Nahrstedt, "QoS and Contention-Aware Multi-Resource Reservation", IEEE International Conference on High-Performance Distributed Computing (HPDC), August 2000, Pittsburgh, PA, pp. 3-10.

[Kalter00a] W. Kalter, B. Li, W. Jeon, K. Nahrstedt, J. Seo, "A Gateway-Assisted Approach Towards QoS Adaptations", IEEE International Conference on Multimedia and Expo (ICME), July 2000, New York, NY, pp. 855-858.

[Xu00c] D. Xu, D. Wichadakul, K. Nahrstedt, "Resource-Aware Configuration of Ubiquitous Multimedia Services", IEEE International Conference on Multimedia and Expo (ICME), July 2000, New York, NY, pp. 851-854.

[Xu00d] D. Xu, D. Wichadakul, K. Nahrstedt, "Multimedia Service Configuration and Reservation in Heterogeneous Environments", IEEE International Conference on Distributed Systems Computing, April 2000, Taiwan, pp. 512-519.

[Li00c] B. Li, "Adaptive QoS Framework and Its Application to Visual Tracking", PhD Thesis, May 2000, University of Illinois at Urbana-Champaign.

[Kalter00b] W. Kalter, "Design, Implementation and Experimentation of a Gateway Architecture for End-to-End QoS Adaptation", Master Thesis, August 2000, University of Illinois at Urbana-Champaign.

[Viswan00] A. Viswanathan, "Design and Evaluation of a CPU-aware Communication Broker for RSVP-based Networks", Master Thesis, May 2000, University of Illinois at Urbana-Champaign.

[Gu00] X. Gu, "Visual QoS Programming Environment for Multimedia Applications", Master Thesis, December 2000, University of Illinois at Urbana-Champaign (received Best MS 2001 Kuck Award).

2001 Publications, Theses and Talks:

[Talwar01a] V. Talwar, S. Nath, K. Nahrstedt, "RSVP-SQoS: A Secure RSVP Protocol", IEEE International Conference on Multimedia and Expo (ICME), August 2001, Tokyo, Japan, electronic proceedings.

[Cui01] Y. Cui, D. Xu, K. Nahrstedt, "SMART: A Scalable Middleware Solution for Ubiquitous Multimedia Service Delivery", IEEE International Conference on Multimedia and Expo (ICME), August 2001, Tokyo, Japan, electronic proceedings.

- [Gu01a] X. Gu, K. Nahrstedt, "Visual QoS Programming Environment for Ubiquitous Multimedia Services", IEEE International Conference on Multimedia and Expo (ICME), August 2001, Tokyo, Japan, electronic proceedings.
- [Li01] B. Li, D. Xu, K. Nahrstedt, "Towards Integrated Runtime Solution in QoS-aware Middleware", ACM Multimedia Middleware Workshop, October 2001, Ottawa, Canada.
- [Gu01b] X. Gu, K. Nahrstedt, "An Event-Driven, User-Centric, QoS-aware Middleware Framework for Ubiquitous Multimedia Applications", ACM Multimedia Middleware Workshop, October 2001, Ottawa, Canada.
- [Yuan01] W. Yuan, K. Nahrstedt, X. Gu, "Coordinating Energy-Aware Adaptation of Multimedia Applications and Hardware Resource", ACM Multimedia Middleware Workshop, October 2001, Ottawa, Canada.
- [Nahrstedt01] K. Nahrstedt, D. Xu, D. Wichadakul, B. Li, "QoS-aware Middleware for Ubiquitous and Heterogeneous Environments", IEEE Communication Magazine, Vol. 39, No. 11, November 2001, pp. 140-148.
- [Wichadak01] D. Wichadakul, K. Nahrstedt, X. Gu, D. Xu, "2KQ+: An Integrated Approach of QoS Compilation and Reconfigurable, Component-based Run-time Middleware for Unified QoS Management Framework", IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2001), November 2001, Heidelberg, Germany, pp. 373-394.
- [Xu01] D. Xu, "A QoS-aware Framework for Ubiquitous Multimedia Service Provision", PhD Thesis, December 2001, University of Illinois at Urbana-Champaign.
- [Talwar01b] V. Talwar, Master Thesis, May 2001, University of Illinois at Urbana-Champaign (received Best MS 2002 Kuck Award).
- 2002 Publications, Theses and Talks:**
- [Gu02a] X. Gu, K. Nahrstedt, "An XML-based Quality of Service Enabling Language for Web", Journal on Visual Languages and Computing, Academic Press, Special Issue on Multimedia Languages for the Web, Vol. 13, No. 1. February 2002, pp.61-95.
- [Yuan02] W. Yuan, K. Nahrstedt, "Integration of Dynamic Voltage Scaling and Soft-Real-Time Scheduling for Open Mobile Systems", Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), February, 2002.
- [Xu02a] D. Xu, K. Nahrstedt, D. Wichadakul, "MeGaDiP: A Wider Area Media Gateway Discovery Protocol", Information Science Journal, Elsevier Science Publisher, Vol. 141, No. 1-2, March 2002, pp. 37-59.
- [Xu02b] D. Xu, K. Nahrstedt, "Finding Service Paths in a Media Service Proxy Network", SPIE International Conference on Multimedia Computing and Networking (MMCN), Electronic Imaging Symposium, January 2002, San Jose, CA, pp. 171-185.
- [Wichadak02] D. Wichadakul, K. Nahrstedt, "A Translation System for Enabling Flexible and Efficient Deployment of QoS-aware Applications in Ubiquitous Environments", IFIP/ACM 1st International Working Conference on Component Deployment, June 2002, Berlin, Germany.
- [Gu02b] X. Gu, K. Nahrstedt, "Dynamic QoS-aware Multimedia Service Configuration in Ubiquitous Computing Environments", IEEE International Conference on Distributed Computing Systems (ICDCS), July 2002, Vienna, Austria.
- [Gu02c] X. Gu, K. Nahrstedt, "A Scalable QoS-aware Service Aggregation Model for Peer-to-peer Computing Grids", IEEE International Conference on High-Performance Distributed Computing (HPDC), July 2002, Edinburgh, Scotland.

- [Lui02a] K. Lui, K. Nahrstedt, S. Chen, “Routing with Topology Aggregation in Bandwidth-Delay Sensitive Networks”, accepted to IEEE/ACM Transactions on Networking, 2002.
- [Lui02b] K. Lui, “STAR Bridge Protocol”, PhD Thesis, May 2002, University of Illinois at Urbana-Champaign.
- [Gupta02] G. Gupta, “Distributed Dynamic Soft-Real-Time Scheduling for Focus-based Multimedia Applications”, Master Thesis, May 2002, University of Illinois at Urbana-Champaign.
- [Prabhu02] R. Prabhu, “Software Framework for Secure Distributed Multimedia Applications”, Master Thesis, May 2002, University of Illinois at Urbana-Champaign.

List(s) of Symbols, Abbreviations and Acronyms

ATM	Asynchronous Transfer Mode
CAN	Control Area Network
CBR	Constant Bit Rate
DSRT	Dynamic Soft-Real-Time Scheduling
EDF	Earliest Deadline First
FBS	Feedback-based Synchronization
FM	Fast Messages
HCF	Hybrid Coordination Function
IP	Internet Protocol
ISP	Internet Service Provider
PC	Personal Computer
PCPT	Periodic Constant Processing Time
PVPT	Periodic Variable Processing Time
QoS	Quality of Service
RR	Round Robin
RSVP	Resource Reservation Protocol
S-CFB	Scheduled Contention Free Burst
WRR	Weighted Round Robin