ON DESIGN AND REALIZATION OF NEW GENERATION MISSION-CRITICAL APPLICATION SYSTEMS

A Dissertation

by

ZHIBIN MAI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Major Subject: Computer Engineering

ON DESIGN AND REALIZATION OF NEW GENERATION MISSION-CRITICAL APPLICATION SYSTEMS

A Dissertation

by

ZHIBIN MAI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2011

Major Subject: Computer Engineering

ABSTRACT

On Design and Realization of New Generation

Misson-critial Application Systems.

(May 2011)

Zhibin Mai, B.S.; M.S., Shanghai Jiao Tong University

Chair of Advisory Committee: Dr. Wei Zhao

Mission-critical system typically refers to a project or system for which the success is vital to the mission of the underlying organization. The failure or delayed completion of the tasks in mission-critical systems may cause severe financial loss, even human casualties. For example, failure of an accurate and timely forecast of Hurricane Rita in September 2005 caused enormous financial loss and several deaths. As such, real-time guarantee and reliability have always been two key foci of mission-critical system design.

Many factors affect real-time guarantee and reliability. From the software design perspective, which is the focus of this paper, three aspects are most important. The first of these is how to design a single application to effectively support real-time requirement and improve reliability, the second is how to integrate different applications in a cluster environment to guarantee real-time requirement and improve reliability, and the third is how to effectively coordinate distributed applications to support real-time requirements and improve reliability.

Following these three aspects, this dissertation proposes and implements three novel methodologies: real-time component based single node application development, real-time workflow-based cluster application integration, and real-time distributed admission control. For ease of understanding, we introduce these three methodologies and implementations in three real-world mission-critical application systems: single node mission-critical system, cluster environment mission-critical system, and wide-area network mission-critical system. We study full-scale design and implementation of these mission-critical systems, more specifically:

1) For the single node system, we introduce a real-time component based application model, a novel design methodology, and based on the model and methodology, we implement a real-time component based Enterprise JavaBean (EJB) System. Through component based design, efficient resource management and scheduling, we show that our model and design methodology can effectively improve system reliability and guarantee real-time requirement.
2) For the system in a cluster environment, we introduce a new application model, a real-time workflow-based application integration methodology, and based on the model and methodology, we implement a data center management system for the Southeastern Universities Research Association (SURA) project. We show that our methodology can greatly simplify the design of such a system and make it easier to meet deadline requirements, while improving system reliability through the reuse of fully tested legacy models.

3) For the system in a wide area network, we narrow our focus to a representative VoIP system and introduce a general distributed real-time VoIP system model, a novel system design methodology, and an implementation. We show that with our new model and architectural design mechanism, we can provide effective real-time requirement for Voice over Internet Protocol (VoIP).

DEDICATION

To my wife and parents

ACKNOWLEDGEMENTS

This dissertation would not have been possible to complete without the help of so many great people.

I would like to thank Professor Wei Zhao, my advisor, for his constant guidance, support, inspiration, and encouragement throughout my Ph.D. studies and work at Texas A&M University. He taught me the principles of doing research and ways to approach research problems. He gave me insightful advice for my research as well as daily life. He showed me how to maintain a positive attitude in the face of challenges and how to deal with them. He is a master teacher and his comments always hit the mark. What I learned from him will benefit my future.

I would like to express my sincere gratitude to Professor William Lively for his guidance, support and encouragement, especially after Professor Zhao left Texas A&M University.

I want to thank Professor Jyh-Charn Liu for his guidance and for providing insightful comments to improve the quality of my dissertation.

I also want to thank Professor Evan Anderson for serving as my committee member and for sharing his research experiences with me.

Many thanks go to Larisa for helping me edit my research papers and dissertation.

I want to thank all the fellow students in my research group, especially Jianjia Wu, Shengquang Wang, Dan Cheng and Nan Zhang, for their collaboration and insightful comments.

Finally, I would like to thank my parents and my parents-in-law for their unconditional help and support. I owe a lot to my beloved wife who helped me take care of the family and our three lovely daughters, Cheryl, Carol and Cassie.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

As information technologies play increasingly important roles throughout our society, mission-critical application systems are widely deployed in various sectors including health, energy, manufacturing, defense, air and space industries, etc. Generally speaking, the term "mission-critical" refers to a factor (facility, process, procedure, software, etc.) that is crucial to the successful completion of an entire project. It may also refer to a project or system for which the success is vital to the mission of underlying organization [1].

The classification of mission-critical systems is in context. Different projects, systems or business have different kinds of mission-critical systems. For example: 1) The U.S. Department of Defense (DoD) refers to mission-critical systems as those systems that are critical to DoD's ability to meet its responsibilities and include command and control systems, satellite systems, inventory management systems, transportation management systems, medical systems and equipment, and pay and personnel systems [3]; 2) The National Hurricane Center (NHC) refers its mission-critical systems to weather forecast systems, such as the National Centers for Environmental Prediction (NCEP) Advanced Weather Interactive Processing System and the Automated Tropical Cyclone Forecasting (ATCF) system are specific mission-

---

This dissertation follows the style of *IEEE Transactions on Parallel and Distributed Systems.*

critical systems to National Hurricane Center (NHC) [4]; 3) The South Carolina Emergency Management Division (SCEMD) defines its mission-critical systems to be the Automated Data Processing (ADP) equipment that supports the execution of the agency's essential functions during hurricane season [5].

In spite of the differences in definitions and objectives, these mission-critical application systems differ from traditional numerical applications as they are usually tightly coupled with control processes and hence are critical for the accomplishment of mission objectives underlying physical systems. The failure or delayed completion of the tasks in mission-critical systems may cause severe financial loss, even human casualties. For example, failure of an accurate and timely forecast of Hurricane Rita in September 2005 caused enormous financial loss and several deaths [2]. As such, real-time guarantee and reliability have always been two key focuses of mission-critical system design. These issues present challenges that do not occur in traditional computer systems and must be addressed by different strategies.

Many factors affect the real-time guarantee and reliability of a mission-critical system. On the software design perspective, which is the focus of this paper, three aspects are most important. The first of these is how to design a single application to effectively support real-time requirement and improve reliability, the second is how to integrate different applications in a cluster environment to guarantee real-time requirements and improve reliability, and the third is how to effectively coordinate distributed applications to support real-time requirements and improve reliability.

Following these three aspects, this dissertation proposes and implements three novel methodologies: real-time component based application development, real-time workflow-based application integration, and real-time distributed admission control. For ease of understanding, we introduce these three methodologies and implementations in three real-world mission-critical application systems, single node mission-critical system, cluster environment mission-critical system, and wide-area network mission-critical system. We study full-scale design and implementation of these mission-critical systems, more specifically:

1) For the single node system, we introduced a real-time component based application model, a novel design methodology, and based on the model and methodology, we implement a real-time component based Enterprise JavaBean (EJB) System. Through component based design, efficient resource management and scheduling, we show that our model and design methodology can effectively improve system reliability and guarantee real-time requirements.
2) For the system in a cluster environment, we introduced a new application model, a real-time workflow-based application integration methodology, and based on the model and methodology, we implement a data center management system for the Southeastern Universities Research Association (SURA) project. We show that our methodology can greatly simplify the design of such a system and makes it easier to meet deadline requirements, while improving system reliability through reuse of fully tested legacy models.

3) For the system in a wide area network, we narrow our focus to a representative VoIP system and introduce a general distributed real-time VoIP system model, a novel system design methodology, and an implementation. We show that with our new model and architectural design mechanism, we can provide effective real-time requirement for Voice over Internet Protocol (VoIP).

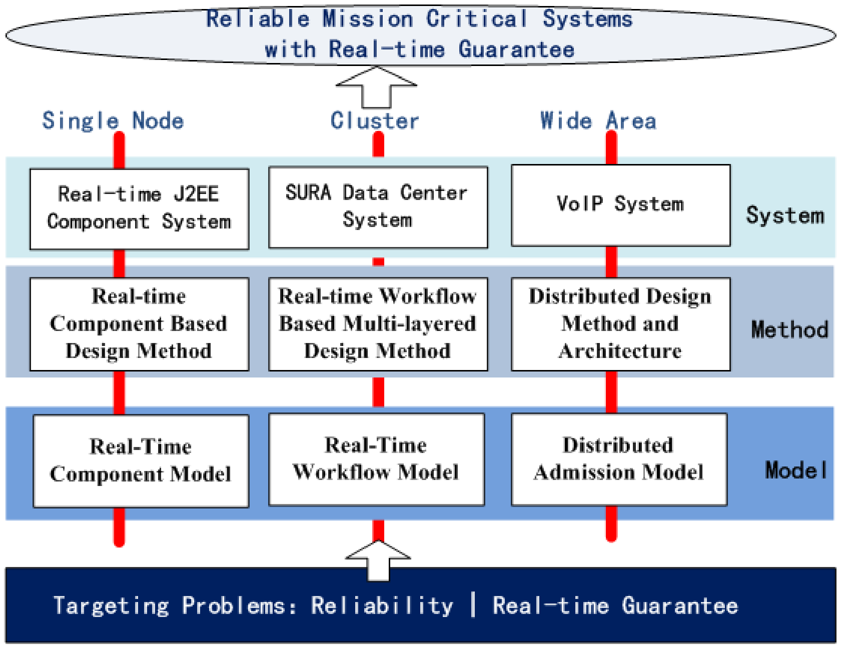

Reliable Mission Critical Systems with Real-time Guarantee
Single Node
Cluster
Wide Area
Real-time J2EE Component System
SURA Data Center System
VoIP System
System
Real-time Component Based Design Method
Real-time Workflow Based Multi-layered Design Method
Distributed Design Method and Architecture
Method
Real-Time Component Model
Real-Time Workflow Model
Distributed Admission Model
Model
Targeting Problems: Reliability | Real-time Guarantee

Figure 1. 3 Systems x 3 Layers Research Methodologies

In summary, we proposed 3 systems x 3 layers research methodologies as shown in Figure 1. We address reliability and real-time guarantee issues in the design and realization of the next generation of mission-critical application systems. We propose three novel system models and design methodologies for single node application, cluster environment application, and wide area network environment applications. We use these novel models and methodologies building three demonstration applications and we show that our models and methodologies can greatly improve system reliability and provide real-time guarantees.

A. Overview of Target Application Systems

1. Single Node Mission-critical System

-- Real-time Enterprise JavaBean (EJB) Component System

A real-time J2EE component system [1] to support mission-critical real-time applications should be considered as another example of the mission-critical system. Component technology has become a central focus of software engineering in research and development due to its success in the market. Reusability is a key factor that contributes to this success [6]. With component technology, software systems are built by assembling components that have already been developed, with integration in mind. With software component frameworks, the nonfunctional codes are automatically generated, and system developers can focus on core business logic parts, without wasting time on common non-functional parts. The reuse of components and developers' focus on core parts lead to a shortening of software development cycles and savings in software development costs.

Although component-based models deal successfully with functional attributes, they provide little support for real-time services. Existing standards – such as CORBA,

[1] J2EE component system here refers to Enterprise Java Bean (EJB) component system. We use these two terms interchangeably in this dissertation.

COM+, and EJB – are unsuitable for real-time applications because they do not address issues of timeliness and predictability of service, which is basically required by real-time systems [7].

The Real-time, Embedded, and Specialized Systems (RTESS) Platform Task Force of the OMG has proposed a specification for a real-time CORBA [8], [9]. The TAO project [10] provided an implementation of the CORBA standard that guarantees that calls across components preserve priority levels and that the overhead in servicing a call request is statically predictable. In [11], Stankovic et al. proposed to use component-based techniques for developing embedded system software, i.e., software for resource constrained systems, and their VEST toolkit aims at providing a rich set of dependency checks based on the concept of aspects to support distributed embedded system development via components. Most of these real-time extensions use traditional approaches to provide real-time service guarantees: Real-time services are typically provided in the form of descriptions of the execution time, period, priority, and deadline to meet expectations for each method invocation. Applications would need intricate knowledge of the underlying hardware architecture and system software (such as subroutine procedures) in the target environment to estimate these parameters accurately. This is a big burden for any application, especially in large-scale and heterogeneous environments. Moreover, with these traditional approaches in component-based systems, components may no longer be reusable in terms of providing real-time services. Therefore, the benefit of component technology may be lost.

Our goal in this work is to develop a real-time component based system via integration with the Commercial Off-The-Shelf (COTS) component system that maintains the reusability of components, thus improving system scalability and dependability and to provide real-time guarantees.

2. Cluster Environment Mission-critical System

-- SURA Scientific Computing Data Center System

Scientific computing helps scientists and engineers use computers to analyze and solve scientific and engineering problems. It plays a critical role in different areas and is considered the third mode of science, complementing theory and experimentation. To support scientific computing, many computing facilities, including national, regional, and laboratory centers, have been established. The major task of these facilities is to enable the transparent execution of complex mathematical models and numerical methods on a large set of scientific data over combined computing power resources provided by the computing facilities, thus allowing scientists to focus on mathematical modeling and knowledge discovery, which best utilizes their expertise.

As part of the Southeastern Universities Research Association (SURA) Coastal Ocean Observing and Prediction (SCOOP) Program, the SURA scientific computing data center at Texas A&M University (referred to as the “SURA data center”) aims to support SURA's coastal research effort. The SCOOP Program is a multi-institutional collaboration whose partners are working to implement a modular, distributed system for real-time prediction and visualization of the impacts of extreme atmospheric events, including water level, inundation, and wave height [12]. The SURA data center is required to provide storage and data management tools for SURA scientists to remotely archive and share Terabytes of atmospheric data collected from radar sites or generated by simulation programs. It also provides high performance computing infrastructure and legacy simulation program management tools for SURA scientists to remotely run the simulations for real-time prediction and visualization of the impacts of extreme atmospheric events, including water level, inundation, and wave height.

The goal of this study is to design and implement a scientific computing data center management system that provides scientists a cloud computing infrastructure [13] to share data, application and hardware resources in order to support real-time prediction and visualization of the impacts of extreme atmospheric events, including water level, inundation, and wave height. We proposed a workflow driven methodology to ease the effort of integrating existing software models into new scientific computing applications.

3. Wide Area Networked Mission-critical System

-- QoS Provisioning System for Voice over Internet Protocol (VoIP)

Voice over IP (VoIP) system is another that has been identified as a mission-critical system to provide voice communication service to the public or organization. Transmission of voice traffic must meet stringent requirements on packet delay as it is an important factor affecting the quality of calls. The International Telecommunication Union (ITU) recommends that a one-way delay between 0-150 ms is acceptable in Recommendation G.114 [14]. However, existing Internet Protocol (IP) networks do not provide Quality of Service (QoS) to voice traffic and, therefore, can only offer best-effort services. New traffic may keep entering the network even beyond the network capacity limit, consequently making both the existing and the new flows suffer packet loss and/or significant delay.

Current VoIP systems have realized the importance of providing QoS guarantees. Their general solution is to introduce a call admission control (CAC) mechanism in order to ensure that sufficient resources are available to satisfy the requirements of both the new and the existing calls after the new call is admitted. Several CAC mechanisms, such as Site-Utilization-Based CAC (SU-CAC) and Link-Utilization-Based CAC (LU-CAC), have been used in current VoIP systems. However, none of current VoIP systems can really provide QoS guarantees to VoIP. The basic reason behind this is that none of them are able to effectively apply and support CAC mechanisms. Their main approach is to use resource over-provisioning, which is expensive and cannot guarantee quality of services.

The goal of this study is to design and implement a practical, scalable and highly efficient VoIP system that can provide the end-to-end QoS guarantees to voice in IP networks. We decide to accomplish our target system by enhancing and integrating with the current Cisco VoIP system, rather than to construct a totally new VoIP system from scratch.

B. Dissertation Contributions

We address issues related to the design and realization of the next generation of mission-critical application systems. We pay special attention to two important aspects of these systems: reliability and real-time guarantees. We demonstrate our methodologies by three types of real-world mission-critical application systems. In particular, our study will address relevant issues via the three real-world cases which are:

1) For the single node system, we introduced the real-time component based design and implementation for Enterprise JavaBean (EJB) and showed that our design can effectively integrate with the Commercial Off-The-Shelf (COTS) component system and provide real-time guarantees for single node mission-critical systems.
2) For the system in a cluster environment, we introduced the real-time workflow-based application integration methodology to effectively integrate the legacy applications to build comprehensive applications and we used it in the data center management system for the Southeastern Universities Research Association (SURA) project. Our methodology greatly simplifies the design of such a system and makes it easier to meet deadline requirements, while improving system reliability through reuse of fully tested legacy models.
3) For the system in a wide area network, we introduced the distributed real-time admission control mechanism to effectively provide real-time guarantee for a networked application and we show that with this mechanism, we can provide QoS provisioning system for Voice over Internet Protocol (VoIP). Our

methodology on system integration makes it feasible to integrate the Commercial Off-The-Shelf (COTS) VoIP systems to provide real-time VoIP service.

C. Dissertation Outline

This dissertation is organized as follows. In Chapter II the related work is given. Chapter III introduces real-time component based single application development. Chapter IV introduces real-time workflow-based application integration in cluster environment. Chapter V introduces real-time Voice over IP (VoIP) system. Summary and conclusions are given in Chapter VI.

CHAPTER II

RELATED WORK

A. Component Systems

Component technology has become a central focus of software engineering in research and development due to its great success in the market. Reusability is a key factor that contributes to this success [6]. Component standards specify widely-accepted interfaces that allow independent components from different suppliers (third parties) to be plugged together and even to interoperate across language, compiler, and platform barriers. The best known examples of such standards are OMG's CORBA [15], Microsoft's COM+ [16] and Sun Microsystems' Enterprise JavaBeans (EJB) [17].

Although component-based models deal successfully with functional attributes, they provide little support for real-time services. Existing standards – such as CORBA, COM+, and EJB – are unsuitable for real-time applications because they do not address issues of timeliness and predictability of service, which is basically required by real-time systems [7]. The Real-time, Embedded, and Specialized Systems (RTESS) Platform Task Force of the OMG has proposed a specification for a real-time CORBA [8], [9]. At the same time, there is not yet a specification for a real-time EJB or a real-time COM+. The TAO project [10] provided an implementation of CORBA standard that guarantees that calls across components preserve priority levels and that the overhead in servicing a call request is statically predictable. In [11], Stankovic et al. proposed using component-based techniques for developing embedded system software, i.e., software for resource constrained systems, and their VEST toolkit aims at providing a rich set of dependency checks based on the concept of aspects to support distributed embedded system development via components.

Most of these real-time extensions use traditional approaches to provide real-time service guarantees: Real-time services are typically provided in the form of descriptions of the execution time, period, priority, and deadline to meet expectations for each method invocation. Applications would need intricate knowledge of the underlying hardware architecture and system software (such as subroutine procedures) in the target environment to estimate these parameters accurately. This is a big burden for any application, especially in large-scale and heterogeneous environments. Moreover, with these traditional approaches in component-based systems, components may no longer be reusable in terms of providing real-time services. Therefore, the benefit of component technology may be lost.

In this work, we aimed to study the design and implementation of a real-time component framework based on an existing J2EE component framework that assists the component developers and application integrators to develop reusable real-time components and build effective real-time applications.

B. Scientific Computing Data Center Systems

Most existing scientific computing data centers can be categorized into data-intensive centers and computational-intensive centers. The data-intensive data centers provide tools for users to discover, convert, retrieve, read and visualize the data set. In terms of data distribution and management, some data centers [18], [19], [20] (called traditional data centers) collect and centrally archive the data sets. The data sets remain isolated in the hands of the individual data center and are very domain-specific. Furthermore, the data format and name convention are normally inconsistent. The end users must spend a lot of effort and time to discover, retrieve and unify the data set from different data centers in order to produce a comprehensive and complete data set. Understanding the limitation of data sharing inherent in traditional data centers, some research projects [21], [22], [23] build cyber infrastructure framework and systems aimed at providing uniform interfaces for the end users to discover, retrieve and unify the data set distributed in a variety of data sources and a set of data and meta-data management and communication standards for the data source owners (individual or organization) to register and post their data services, which are transparent to the end users. Such kinds of data centers normally won't own and archive the data sets. This will reduce the cost of the data centers and avoid the problems of data ownership. However, the design and implementation of those data centers are very complicated. And the data center might become a single point of failure for data discovery. Furthermore, the potential failure of the network between the data center and data source will reduce the availability of the data sets.

The computational-intensive data centers provide resource and application deployment tools for the users to deploy and run their applications in the data centers [24], [25], [26], [27]. The computational-intensive data center can significantly improve the utilization of the resource via resource sharing, which in turn reduces the cost-per-use of the resource. For example, Sun grid of Sun Microsystems already offers an affordable compute utility with a cost of $1/cpu-hr [27]. Furthermore, professional IT engineers in the data center can provide technical support to users with higher quality but lower costs, as compared with technical support teams within individual companies or organizations. As a result, the computational-intensive data centers enable commercial companies to outsource their IT services to have lower costs and better services [28], [29]. However, the current model of the computational-intensive data center is not always applicable to the scientific computing society. As we know, most of the scientific computing applications are not only resource intensive (also called computing-intensive), but they are also data intensive. To reduce the network overload and data staging time, the applications, especially the data intensive application, should be deployed to the resources close to where the data reside. Unlike the data-intensive data center, the computational-intensive data centers do not archive data for the users' applications. It is impractical, if not infeasible, to stage gigabytes of data across the internet between the data center and data source per execution. Furthermore, IT support engineers in the data center are not domain-specific experts. Therefore, they are of minimum help when it comes to the deployment and troubleshooting of domain-specific applications.

The SURA scientific computing data center needs to provide the capabilities of data-intensive centers and computational-intensive centers with a comprehensive data set, high performance computing resources and a range of domain-specific applications. The SURA scientific computing data center management system facilitates different roles of data center users to manage and use those high value resources in the data center. It created a cloud computing [13] infrastructure that shared resources, applications, and data are provided to support SURA scientists' research.

Several research projects, such as LEAD [30], GEON [31], GAP [32], designed and implemented similar scientific computing data center systems or tools. However, our proposed system is superior to those systems in several aspects.

LEAD is a large scale effort to build infrastructure allowing atmospheric science researchers to dynamically and adaptively respond to weather patterns to produce better-than-real time predictions of tornadoes and other "mesoscale" weather events. LEAD has some similarities with our proposed system. Both use workflow technology to reuse the existing software modules to construct and auto-handle larger scale scientific analysis. Both also support event driven to dynamically and adaptively respond to the predefined external events, in order to conduct real time analysis. However, the difference lies in the explicit separation of user roles and the real-time guarantee mechanism. Our proposed system explicitly classifies users into different roles and provides different interfaces and supporting tools to facilitate the separation. Our proposed system uses the scalable admission control mechanism to resolve resource competition issues in order to guarantee the admitted workflows to be completed by their deadlines.

GEON is developing cyber infrastructure for integrative research to enable transformative advances in geosciences research and education. Similar to LEAD, GEON is also based on service-oriented architecture, with an emphasis on supporting geosciences. Different from LEAD and GEON, our proposed system is a generic system for scientific research. It imposes no constraints on the kinds of supported scientific applications and data even though it is currently deployed in the SURA scientific computing data center to support atmospheric research.

GAP is a system to enable legacy scientific applications on computing grids using a service-oriented architecture. It provides the approach to automatically turn an application into a service and generate the user interface from an XML description for the application without the need for coding. Like our proposed system, GAP explicitly classifies users into different roles, such as application specialist, system administrator and users. Similar to GAP, our system can automatically wrap the application to the service element and generate user interface without the application integrator writing any codes. However, our system frees the application integrator from writing tedious xml descriptions, which describe how to turn the application to service element, via the workflow module editing features in the workflow GUI editor. In addition, GAP currently does not support real-time guarantee service and workflow mechanism, although workflow is identified in their future plan.

C. Voice over IP Systems

In the past few years, VoIP has rapidly gained acceptance. Many leading vendors in the traditional voice and data industry have switched to provide VoIP solutions. In general, VoIP vendors are classified into two camps, in terms of the VoIP solutions. Traditional voice vendors, such as Nortel [33], Avaya [34], Alcatel [35], etc., with products based on circuit-switched technologies (Time Division Multiplexing), would like to adapt traditional circuit switched voice solution to the VoIP solution. Meanwhile, data vendors, such as Cisco [36], and 3Com [37], would like to provide a pure VoIP solution, treating voice as just another data stream.

To provide end-to-end delay guarantees in VoIP systems, Call Admission Control (CAC) mechanisms must be in place. CAC algorithms can be roughly grouped into two broad categories: 1) The Measurement-based CAC algorithm: It uses network measurement to estimate the current load of existing traffic. It has no prior knowledge of traffic statistics and makes admission decisions based on the current network state only [38], [39]. 2) The Parameter-based CAC algorithm: It uses the parameters of resource and service to decide whether the network can accommodate the new connection while providing end-to-end delay guarantees. The parameters are used to compute a deterministic bound imposing that, in any traffic situation, the end-to-end delay guarantees are provided for all flows [40], [41], [42]. Utilization-based CAC belongs in this category, where the parameters are the requested bandwidth utilization for each new connection and the available bandwidth utilization in the resource [40], [41]. Through appropriate system (re)configuration steps, the delay guarantee test at run time is reduced to a simple utilization-based test: As long as the utilization of links along the path of a flow is not beyond a given bound, the performance guarantee of the end-to-end delay can be met. Utilization-based CAC renders the system scalable.

In this work, we aimed to study the design and implementation of a utilization-based admission control system, also called QoS-Provisioning system, to seamlessly integrate with existing VoIP systems in order to provide delay guarantee to voice traffic over IP network.

CHAPTER III

REAL-TIME COMPONENT BASED SINGLE APPLICATION DEVELOPMENT

A. Overview

In this section, we focus on how to design single node mission-critical applications to effectively improve system reliability, at the same time providing real-time guarantees.

In a single node software application environment, the major factor affecting software reliability is software defects. It is widely acknowledged that the most effective way to reduce software defects is through reuse. Component-based software design is one of the most successful application reuse approach that has been proven to greatly reduce software defects and improve software reliability.

Although extensive work have been done on component-based application development models, they provide little support for real-time services. Existing standards – such as CORBA, COM+, and EJB – are unsuitable for real-time applications because they do not address issues of timeliness and predictability of service, which is basically required by real-time systems [6]. Some work has been done on real-time extensions of component-based design, but these extensions use traditional approaches to provide real-time service guarantees: Real-time services are typically provided in the form of descriptions of the execution time, period, priority, and deadline to meet expectations for each method invocation. Applications would need intricate knowledge of the underlying hardware architecture and system software (such as subroutine procedures) in the target environment to estimate these parameters accurately. This is a big burden for any application, especially in large-scale and heterogeneous environments. Moreover, with these traditional approaches in component-based systems, components may not be reusable any more in terms of providing real-time services. Therefore, the benefit of component technology may be lost.

Given the great potential benefits derived from component technology improving system reliability, and given the lack of consideration for reusability of components in providing real-time services, our goal in this chapter is to introduce a real-time component based application design model for single node application development. More specifically,

1) we introduce a real-time component based application model for single node applications that not only support component based application development, but that can also provide real-time guarantees; our proposed real-time component based application model includes a general application model, a real-time component model, and a real-time admission control model;

2) on top of the proposed real-time component based application model, we propose a real-time component-based system design methodology, which provides a detailed guide for application developer on implementation of real-time component services, optimization of real-time component services, service adaption, and architecture design;

3) based on the real-time component based application development model and the proposed design methodology, we implement a demonstration system. The performance

results show that our proposed model and design methodology can improve system reliability and provide scalable real-time guarantees to the enhanced J2EE components.

B. Real-time Component-based Application Model

1. Application Model

We consider hybrid open/closed systems, where applications include clients and application servers, each of which hosts one or more components. Each invocation from a client triggers execution of one or more methods, either on a single component or on several components. These components, in turn, can be located on one or across several application servers. A sequence of client invocations resulting in such a sequence of method executions is called a task. In a component-based system, an invocation from a client can pass through several components and we assume that all invocations in the same task will execute on the same components in the same order. Tasks in applications can be modeled as a directed acyclic graph (DAG) which we call task graph. Each node is a component and each task forms a task path. There could be multiple tasks along each task path.

2. Real-time Component Model

In component software, a component has three basic characteristic properties [18]: (i) *Isolation* – A component should be deployable independently as an isolated part. The component is an atomic unit of deployment, as it will never be deployed partially. (ii) *Composability* – A component should be composable with other components. It needs to be a self-contained function unit with well-specified interfaces. A third party can access the component through its contractually specified interfaces. (iii) *Opaqueness* – Neither the environment nor other components or a third party have access to its implementation or other internal details. Figure 2 illustrates a component architecture.

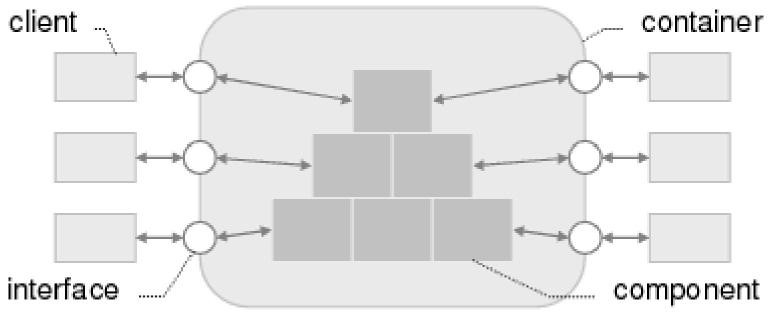


Figure 2. Component Architecture

We extend the component architecture described above to build a real-time component architecture. For this, we augment the largely functional interfaces and context dependencies with contractually specified temporal interfaces and explicit time-related context dependencies. Any such augmentation of the component interface architecture should continue to satisfy the three basic component properties described earlier: (i) The real-time interface architecture should not interfere with the isolation property. Each component should be separated from other components in providing real-time service guarantees. For example, uncontrolled resource conflicts among different components should be avoided. (ii) Composability should be maintained. The real-time service interface should effectively represent the real-time service provided by the component. Applications can access the service through the real-time service interface. (iii) The real-time interface architecture should maintain opaqueness. Applications do not need to know how real-time services are provided by each component. The interfaces should not include information relating to the underlying component implementation, such as methods' worst-case execution time, or any scheduling algorithm used in method execution in components, for example.

We use a very simple contractual interface, which formulates the real-time service provided in terms of the service guarantee (described in form of a deadline) given a worst case arrival (described in form of an arrival function). We first introduce the arrival function.

Definition 3-1. (**Arrival function**) If the maximum number of method invocations during any time interval of length I is bounded by $A(I)$, we define A as an arrival function of this sequence of method invocations.

For example, a bursty arrival can be described using a burst size σ and average arrival rate2 ρ as $A(I) = \sigma + \rho \bullet I$. The arrival function A and the deadline D give a contractual definition of the real-time service provided by the component: If the sequence of invocations of methods in Component e has an arrival function below A, it is guaranteed that any invocation in this sequence will meet its deadline D at Component e. This interface specification clearly meets the isolation, composability, and opaqueness requirements for real-time components.

[2] Here we use a bound instead of using $\lfloor \bullet \rfloor$ operator

However, this specification has two shortcomings in practice: First, a component will only provide a single real-time service to applications. Often, different applications may require different levels of timing requirement. Second, each component usually exposes a number of methods and different methods could be invoked at different times, which have different resource consumptions. In order to let components provide more flexible service and better utilize the underlying resource usage, we extend the above specification by introducing different service levels and taking into consideration different methods exposed by components. We define class of service as the service level for each component. Assuming there are M classes of service, we define Class-i real-time service for Component e as $\left\langle \Theta_{e,i}, A_{e,i}, D_{e,i} \right\rangle$, where $\Theta_{e,i}$ is a group is a group of methods exposed by Component e, $A_{e,i}$ is an arrival function of invocations of methods in, and $D_{e,i}$ is a deadline for any invocation of methods in $\Theta_{e,i}$. In other words, If the sequence of invocations of methods in $\Theta_{e,i}$ has an arrival function below $A_{e,i}$, Component e guarantees a worst-case delay bounded by $D_{e,i}$ for any method invocation in this sequence.

For example, assume that Component e exposes four methods $\theta_1,...,\theta_4$ and defines four classes, a real-time service interface specification is illustrated in Table 1. In this example, θ_1 and θ_2 may represent the main methods exposed by the component, while θ_3 and θ_4 are used for management and auditing of the component. Clients that use Class-2 service can access the component at a higher rate than ones that use Class-1 service, but receive less stringent timing guarantees (0.250 sec instead of 0.050 sec).

Clients that use Class-3 service have a different view of the component than ones that use Class-1 or Class-2 service (they may need to access all methods of the component) and have different real-time requirements. In this example, test suites may need to access all methods exposed by a component, and may need to do this in a timely fashion. Auditing and management applications, on the other hand, may need to access only a small subset of methods, and have only loose time requirements. Note that Class-1 and Class-2 services expose the same set of methods; that is, $\Theta_{e,1}$ and $\Theta_{e,2}$ are identical. The functional aspect of the service interfaces is therefore identical, while their difference lies entirely in the real-time specification, more specifically in the expected arrival and in the timing guarantees.

Table 1. A Real-time Service Interface Specification

class i	$\Theta_{e,i}$	$\mathcal{A}_{e,i}(I)$	$D_{e,i}$
1	θ_1, θ_2	$1 + 2\,I$	0.050 sec
2	θ_1, θ_2	$2 + 8\,I$	0.250 sec
3	$\theta_1, \theta_2, \theta_3, \theta_4$	$3 + 9\,I$	0.150 sec
4	θ_3, θ_4	$1 + 4\,I$	0.300 sec

This separation of functional from timing specification allows for a configuration of real-time components into resource overlays, which in turn allow for the isolation of applications from the details of the low-level specification and management of the underlying computational resources. Figure 3 shows an example of a component-based resource overlay. This figure illustrates how resource overlays separate component development from application development and relieve the application designer of the underlying resource management. In fact, application designers implement and deploy their systems on the resource overlay, which provides well-defined functional abstractions and timing behaviors. The real-time service interface specification in real-time components does not provide access to their underlying implementation. Any change in the implementation of components will, therefore not affect the application design or behavior.

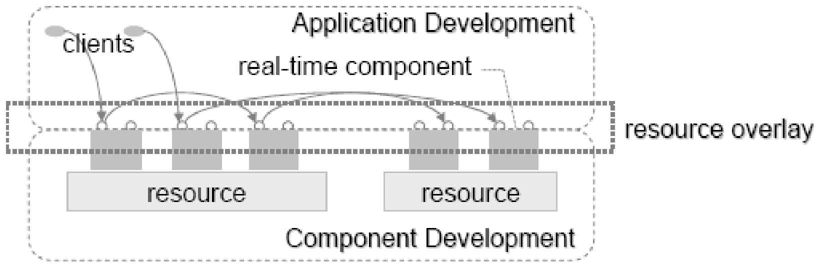


Figure 3. A Component Based Resource Overlay

It is up to the component providers to map the nodes of the component-based resource overlay to the underlying available resources. This is typically done independently of the particular application. In the following, we describe how application designers make use of resource overlays to build real-time applications. Given a set of real-time component services, it is the component providers' responsibility to implement the component functionality defined by its set of interfaces. Component providers must ensure that both functional and timing properties are satisfied for each implemented real-time component. We will address this issue later.

3. Scheduling and Admission Control Model

Any task is associated with an arrival descriptor (in form of the source arrival function) and a timing requirement (in form of the end-to-end deadline). To provide real-time service guarantees, application designers have to ensure that every invocation in a task meets the end-to-end deadline requirement. Moreover, the application designer must ensure that the real-time service specified in each real-time component will not be violated. Since the maximum arrival is part of the real-time service of the component, and the client population is not under control of the application servers, an admission control mechanism has to be in place.

For a new Task T, admission control has to address the two parts of the real-time specification of the components (timing guarantee and arrival descriptor) to provide real-time guarantees. First, what is the worst-case end-to-end delay experienced by any invocation in Task T? In Task T, all of its invocations have an end-to-end deadline requirement D^T. Assume that each invocation in Task T will go through a sequence of components e_h of Class i_h, $h = 1, 2, \ldots, H$, and any method that Task T will call in Component e is in $\Theta_{e,i}$. Recall that the worst-case delay provided by Component e of Class-i is $D_{e,i}$. To guarantee the end-to-end deadline for any invocation in Task T, admission control has to ensure that the end-to-end delay d^T suffered by any client invocation in Task T should be bounded as:

$$d^T = D_{e1,i1} + \ldots + D_{eH,iH} \le D^T \qquad \text{(III-1)}$$

Secondly, what is the consumed resource by Task T ? Provided that a task has an arrival function $A^{in}(I)$ before arriving at a component, the arrival function will become $A^{out}(I) = A^{in}(I+d)$ just after a worst-case delay d at this component. We define A^T as the source arrival function of Task T (before calling the first component). Then the arrival function of T at Component e_h of Class i_h, h = 1, 2, . . . , H , is

$$A^T_{e_h,i_h}(I) = A^T(I + D_{e_1,i_1} + ... + D_{e_{h-1},i_{h-1}}) \qquad \text{(III-2)}$$

If A^T is defined with a burst size σ^T and an average arrival rate ρ^T as $A^T(I) = \sigma^T + \rho^T I$, then the consumed resource by Task T at Component e of Class i is

$$A^T_{e_h,i_h}(I) = (\sigma^T + \rho^T D_{e_1,i_1} + ... + \rho^T D_{e_{h-1},i_{h-1}}) + \rho^T I \qquad \text{(III-3)}$$

Admission control has to ensure that the real-time service specified at each real-time component along the task path of Task T will not be violated, i.e.,

$$\sum\nolimits_{T' \in S_{e_h,i_h}} A^{T'}_{e_h,i_h}(I) \le A_{e_h,i_h}(I) \qquad \text{(III-4)}$$

where S_{e_k,i_k} is the set of existing tasks that use Class-i_h service of Component e_h.

In summary, admission control ensures that sufficient overlay resources are available to meet the requirements of both the new and the existing tasks whenever a new task has been admitted. In other words, both (III-1) and (III-4) should remain satisfied for both new and existing tasks if a new task is admitted. This admission control mechanism is simple to implement efficiently.

C. Real-time Component Based Application Design Methodology

In this section, based on the previous real-time component based application model, we propose a new application development methodology for single node application development. Specifically, we give a detailed method and/or algorithm on the implemented real-time component service, the optimization of real-time component service, the adaption of real-time component, and a general system architecture for real-time component based single node applications.

1. Service Implementation

Service implementation issues can be divided into two categories: (i) *Inter-component* – Recall that each real-time component should be isolated from others in terms of the underlying resource usage to meet the isolation requirement. The underlying resource could be CPU, memory, link bandwidth, or others (here we focus on CPU). We use a guaranteed-rate scheduler to ensure temporal isolation of components on the same processor, and we allocate the required amount of the processor utilization to each component. A total bandwidth server (TBS) [43] can achieve this; (ii) *Intra-component* – Each component will provide multiple classes of service. To differentiate among classes of service within the same component, we use a simple static priority scheduler, and use the class-id as priority level.

The main remaining implementation issue is how to determine the processor utilization that needs to be assigned to each component. We will address this as follows.

Since the component implementation is bound to the underlying hardware platform, the execution of the component's methods can be characterized at component implementation time. In particular, each exposed method can be associated with its worst-case execution time (WCET) on the specific platform. We aim to compute the worst-case delay suffered by execution of any method in $\Theta_{e,i}$. For this, we denote $C_{e,i}$ as the maximum WCET of all methods in $\Theta_{e,i}$. In conjunction with the arrival function defined as part of the real-time service interface specification, the WCET gives rise to the workload characterization for the method set $\Theta_{e,i}$ on the underlying implementation platform.

Definition 3-2. (**Workload function**) If the cumulated execution time of a sequence of method executions is bounded by $F(I)$ during any time interval with length I , we define F as a workload function of this sequence of method executions.

Given the invocation arrival function $A_{e,i}(I) = \sigma_{e,i} + \rho_{e,i} I$ for Component e of Class i and the associated $C_{e,i}$ of $\Theta_{e,i}$, the workload function for Component e of Class i can be expressed as $F_{e,i}(I) = C_{e,i} A_{e,i}(I)$. If we assume a constant processor utilization α_e to be assigned to real-time Component e, we can use a time demand/supply argument to derive the worst-case delay $d_{e,i}$ suffered by any method invocation in Component e of Class i as follows:

$$d_{e,i} \le \max_{I < I_{e,i}} \left\{ \frac{1}{\alpha_e} \left(\sum_{p<i} F_{e,p}\left(I + d_{e,i}\right) + F_{e,i}\left(I\right) \right) - I \right\} \qquad \text{(III-5)}$$

where $I_{e,i}$ is the maximum busy interval, satisfying

$$I_{e,i} \le \min\left\{ I : \frac{1}{\alpha_e} \sum\nolimits_{p \le i} F_{e,p}\left(I\right) \le I \right\} \qquad \text{(III-6)}$$

If $A_{e,i}(I)$ can be defined using burst size $\sigma_{e,i}$ and average arrival rate $\rho_{e,i}$, we can explicitly express as the following inequality

$$d_{e,i} \le \frac{\sum_{p \le i} C_{e,p}\sigma_{e,p}}{\alpha_e - \sum_{p < i} C_{e,p}\rho_{e,p}} \le D_{e,i} \qquad \text{(III-7)}$$

Therefore, in order to satisfy all classes of service, the allocated processor utilization for components has to be set at least to

$$\alpha_e = \max_{1 \le i \le M} \left\{ \frac{1}{D_{e,i}} \sum_{p \le i} C_{e,p}\sigma_{e,p} + \sum\nolimits_{p < i} C_{e,p}\rho_{e,p} \right\} \qquad \text{(III-8)}$$

When allocating processor utilization to components, component developers should ensure that the overall processor utilization does not exceed the safe utilization level allowed by the specific platform.

2. Service Optimization

The functional specifications $\Theta_{e,i}$ of a real-time component can be defined only in the component design, the timing requirement $D_{e,i}$ is typically defined early on as well. The arrival descriptor $A_{e,i}$, on the other hand, depends on the expected arrival, and requires some understanding of the deployment environment in order to allow efficient resource utilizations. In the following, we describe how component providers can optimally specify arrival descriptor based on the application arrival pattern.

Assume that the task arrival along task path r is a Poisson process with average rate λ, the running duration of any task along task path r is exponentially distributed with average duration $\frac{1}{\mu_r}$, and any task along task path r is associated with real-time source service specification $\langle A_r(I), D_r \rangle$. The average number of task along task path r is given as $v_r = \frac{\lambda_r}{\mu_r}$. Define rejection probability b_r as the probability that a task request for task path r is rejected. Then, the overall admission probability AP for applications in the system can be expressed as

$$AP = \frac{\sum_{r \in R} v_r \left(1 - b_r\right)}{\sum_{r \in R} v_r} \qquad \text{(III-9)}$$

The objective is to find the optimal real-time service specification to maximize the overall admission probability. Then we have the optimization problem:

Input: Task graph G and the set of task paths R; $\Theta_{e,i}$ and $D_{e,i}$ for $i = 1, \ldots, M$;

$\langle A_r(I), D_r \rangle$, λ_r and $\frac{1}{\mu_r}$ for $r \in R$.

Ouput: $A_{e,i}$'s.

Objective: Maximize the overall task admission probability AP.

Constraints: The overall utilization does not exceed the safe utilization does not exceed the safe utilization level for each application server.

It can be summarized as follows:

$$\text{maximize AP} \tag{III-10}$$

$$\text{subject to } \sum_{e\in\gamma}\alpha_e \le \alpha_\gamma, \gamma\in\Gamma \tag{III-11}$$

where e is located in Processor γ belonging to processor set Γ and α_γ is the safe utilization bound for Processor γ.

In (III-9), the rejection probability b_r is not yet determined, and we compute it in the following using Kelly's approximation approach [44].

We first compute the resource need of different Tasks. We do this with help of a reference unit resource. Let the parameters of a unit resource be $\langle\sigma,\rho\rangle$ [3]. Define $\langle\sigma_{r,e,i},\rho_{r,e,i}\rangle$ as the arrival function of any task along Task Path r at Component e of Class i, then

$$\sigma_{r,e,i} = \sigma_r + \rho_r \sum_{\langle e_h,i_h\rangle \prec r\langle e,i\rangle} D_{e_h,i_h}\ , \tag{III-12}$$

$$\rho_{r,e,i} = \rho_r\ , \tag{III-13}$$

where $\langle e_h,i_h\rangle \prec_r \langle e,i\rangle$ denotes all component services $\langle e_h,i_h\rangle$'s along Task Path r before component service $\langle e,i\rangle$.The resource for $\langle\sigma_{r,e,i},\rho_{r,e,i}\rangle$ can therefore be represented by the scale $a_{r,e,i} = \min\{\lfloor\sigma_{r,e,i}/\sigma\rfloor,\lfloor\rho_{r,e,i}/\rho\rfloor\}$. Similarly, the resource at Component e of Class i is represented by the scale $a_{e,i} = \min\{\lfloor\sigma_{e,i}/\sigma\rfloor,\lfloor\rho_{e,i}/\rho\rfloor\}$.

3 $\langle\sigma,\rho\rangle$ can be chosen as the greatest common divisor (g.c.d) of σ_r's and ρ_r's, respectively.

Define rejection probability $b_{e,i}$ as the probability that a task request for Component e of Class i is rejected. Under the assumption that the rejections at all components are independent [44], we have

$$b_r = 1 - \prod_{\langle e,i\rangle \in r} \left(1 - b_{e,i}\right)^{\alpha_{r,e,i}} . \qquad \text{(III-14)}$$

By Erlang's Formula, $b_{e,i}$ is given by

$$b_{e,i} = E\left[\alpha_{e,i}; v_{e,i}\right], \qquad \text{(III-15)}$$

where $E[a;v] = \left(v^a / a!\right) \Big/ \left(\sum_{n=0}^{a} v^n / n!\right)$, and $v_{e,i}$ is the admitted load to Component e of Class i. With the independence assumption, $v_{e,i}$ is given by

$$v_{e,i} = \frac{1}{1-b_{e,i}} \sum_{r \in R_{e,i}} a_{r,e,i} v_r \left(1 - b_r\right) \qquad \text{(III-16)}$$

where $R_{e,i}$ is the set of paths that go through Component e of Class i.

Therefore, b_r is a solution to the fixed point equations (III-14), (III-15) and (III-16). Hence, AP in (III-9) can be obtained.

This leads to the solution of the optimization problem. The objective function of the optimization problem is non-linear while all the constraints are linear. Therefore, the optimization is a linearly-constrained optimization. There are two issues involved in solving the optimization problem:

- Minimum operators appear in both $a_{e,i}$ and $a_{r,e,i}$. From (III-12) and (III-13), we find that the burst size will increase along the path, but the average rate will not. Therefore, during the optimization process, we can set

$a_{e,i} = \lfloor \rho_e, i / \rho \rfloor \le \lfloor \sigma_e, i / \sigma \rfloor$ and $a_{r,e,i} = \lfloor \rho_{r,e}, i / \rho \rfloor \le \lfloor \sigma_{r,e}, i / \sigma \rfloor$. Minimum operators can be removed.

- The objective function is not continuous because $a_{e,i}$ and $a_{r,e,i}$ are integer functions, which make the problem even harder. If $a_{e,i}$ is small, we can use an exhaustive search of $a_{e,i}$ to find the optimal value. Otherwise, we can reset $a_{e,i} = \rho_{e,i} / \rho \le \sigma_{e,i} / \sigma$ and $a_{r,e,i} = \rho_{r,e,i} / \rho \le \sigma_{r,e,i} / \sigma$, and approximate b_r with the uniform asymptotic approximation (UAA) method [45]. Then the objective function becomes continuous and it can be resolved by an optimization toolbox.

In the above, we assume that the application arrival pattern can be predicted a priori. Otherwise, it may be hard or impossible to specify good real-time services of components beforehand. However, the application pattern can be monitored, provided that the application pattern in our system will not change frequently. Based on the observed application pattern, an optimal service specification can gradually be achieved. Periodically updating the service specifications to reflect changes in the application patterns can greatly increase the utilization level of resources.

3. Service Adaptation

Note that in a component-based system, components compete with a limited amount of underlying resources, and the isolation property of components disables the dynamically sharing of the underlying resource among components, which results in an overall resource underutilization. However, due to variations in the application environment, components may not receive a constant rate service request from the application at each service level. Based on these observations, it is necessary to use service adaptation mechanisms to achieve better resource utilization.

The proposed adaptation scheme therefore allows for a load balancing across the component services on the application server. We define the resource residue $\tilde{A}_{e,i}$, i.e., the amount of currently unused resource as

$$\tilde{A}_{e,i}(I) = A_{e,i}(I) - \hat{A}_{e,i}(I) \qquad \text{(III-17)}$$

where $\hat{A}_{e,i}(I) = \sum_{T' \in S_{e,i}} A_{e,i}^{T'}(I)$ is the amount resource currently used by the existing tasks. For any admission request by a task T for component service $\langle e,i \rangle$, we have to borrow resource from some other component service whenever the resource requested by T exceeds the amount of resource currently used, i.e., $\tilde{A}_{e,i} < A_{e,i}^{T}$. We define $A_{e,i}^{*}$ as the optimal service specification obtained by service optimization algorithm and define a threshold $\Delta_{e,i}$ for any real-time component service $\langle e,i \rangle$. If its current assigned resource is no $\Delta_{e,i}$ less than its original optimal assignment, it could be one of candidates whose resource can be borrowed. The details of this algorithm are shown in Algorithm 3-1. In

Step 2, the algorithm identifies the component service with the maximum resource residue, which can be potentially borrowed by other components in the same application server.

Algorithm 3-1 Service adaptation:

```
Admission request phase for any Task T:
1: if $\tilde{A}_{e,i} < A^T_{e,i}$ then
2:    find a component service $\langle \hat{e}, \hat{i} \rangle = \arg\max_{\langle e,i \rangle \in S} \{ \tilde{A}_{e,i} \}$, where S is the set of all possible
      component service satisfying that $A^*_{e,i} - A_{e,i} \geq \Delta_{e,i}$ and the overall new utilization
      will not violate the overall safe utilization bound after resource adaptation;
3:    update the services of $\langle \hat{e}, \hat{i} \rangle$ and $\langle e, i \rangle$ with their corresponding adapted resource.
4: end if
Tear-down phase for any Task T:
1: undo step 3 in the above.
```

Dynamically adapting real-time services of components can improve the statistical multiplexing gain of the underlying resources. We will show this with our evaluation data in Section “Performance Evaluation”.

D. System Architecture

Figure 4 displays the system architecture of the real-time computing framework. In this architecture, there is one module – utilization allocation and scheduling – for component development, and two modules – admission control and policing – for application development.[4]

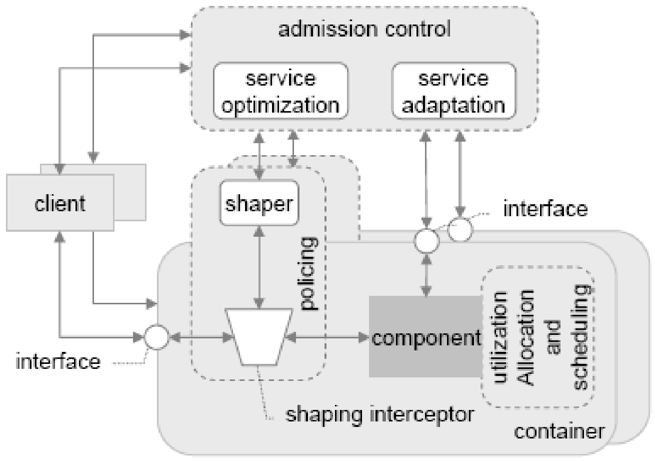


Figure 4. System Architecture of Real-time Computing Framework

Before a real-time Component is deployed, its real-time service interfaces must be specified, and will be loaded into the resource table of the admission control module once the component is deployed. The utilization allocation and scheduling module will reserve the processing utilization assigned to each component with the resource reservation mechanism and schedule the execution of methods at each component with the scheduling algorithm.

[4] In this context, application development is also called component integration, and application developer is also called component integrator. We use both kinds of terms in the chapter interactively.

Admission control is performed at the task level: When the client wants to start a new task, it first sends an admission request to the admission control module. The admission control module will make a decision for this admission request based on the policy in admission control mechanism and the profile including in the admission request. If the admission request is admitted, an acknowledgement message will be sent back to the client, which includes a task ID. At the same time, shaper instances at the corresponding components for this task will be created. During the teardown process of this task, the task will be removed from the existing-task table. Information about the admitted task will be maintained in the existing-task table in the admission control module. The modules for Service optimization and service adaptation are implemented as sub-modules in the admission control module.

When a task is successfully admitted, the system will protect its resources against sources of invocations that exceed their share of invocation arrivals. This is done by appropriately policing the invocations by the policing module. One of the most well-known policing mechanisms in the literature is known as leaky bucket, where the arrival function is defined by a burst size and an average arrival rate. Once policed, the invocations are passed on to the utilization allocation and scheduling module for execution of their corresponding method on the processor.

E. Implementation

1. Background

We used Enterprise JavaBeans (EJB) as the underlying framework of a resource overlay infrastructure based on real-time components. In the following, we first introduce the background information, such as EJB and its implementation JBoss [46], and then address the implementation of our system in details.

a) EJB and JBoss Application Server

EJB technology is a server-side component architecture that simplifies the development and deployment of multi-tier, distributed, scalable, Java enterprise applications. Enterprise beans (beans, to be concise) are server-side components in EJB, which represents a business concept. There are three basic types of beans: (i) entity beans, which represent data in a database, (ii) session beans, which either represent processes or act as agents performing tasks, and (iii) message-driven beans, which are asynchronous message consumers.

JBoss application server is a popular, open-source EJB application server. It provides the basic EJB containers as well as EJB services such as database access (JDBC), transactions (JTA/JTS), messaging (JMS), naming (JNDI) and management support (JMX).

As the foundation for the JBoss infrastructure, JMX [47] provides a common server spine that allows the user to integrate modules, containers, and plug-ins. Service components are declared as Managed Bean (MBean) that are then loaded into JBoss and may subsequently be administered using JMX. MBeans are managed resources and Java objects that follow certain conventions to expose their management interfaces to remote management applications. Remote management applications can access MBeans through JMX agent services. Each MBean is given a unique object name and registered to MBean Server at initial time. MBean Server provides registry service for MBeans. The JBoss EJB server and EJB container are completely implemented using component-based plug-ins onto JMX. When an EJB is deployed into JBoss, a container MBean is created to manage the EJB [48], [49]. In our real-time component-based system, we will build the admission control module as an MBean.

In JBoss, the dynamic proxy approach is used for the server to generate container classes and for the generated container to generate home and remote interfaces of the EJB at run time. Whenever a method invocation is issued on the client-side proxy, the invocation handler creates a special Invocation object, which will reify the method invocation. After traversing a chain of client-side interceptors, the Invocation object is sent by an invoker proxy to an invoker MBean at the server side, where it is routed through the container MBean associated with the target EJB. Each Invocation object includes information about object name, method, and arguments for target EJB. Application developers can place customized interceptors in the interceptor stack traversed by the Invocation object. This interceptor stack mechanism allows developers to add additional services to the called target [48]. We use this interceptor mechanism to build the policing module in our real-time component-based system.

b) Real-time Infrastructure

The implementation of our real-time component-based system is based on JBoss 3.2.1. We use TimeSys Linux RT 3.1 as the underlying real-time operating system [TimeSys]. To complete the platform, we use the RTSJ Reference Implementation (RTSJ-RI) from TimeSys [50] as Java VM. Since TimeSys RTSJ-RI (just as other foreseeable RTSJ implementations) provides only a limited set of Java classes, and JBoss is intended for building enterprise systems, we disabled some advanced features in JBoss while at the same time adding RTSJ compatible Java class libraries. In particular, RTSJ-RI does not support real-time capabilities for Java Remote Method Invocation (RMI) [51] on which the remote invocation is based. As a result, we appropriately extended RMI to make it real-time capable. For this, we eliminated sources for priority inversion, such as, for example, where the listening thread is used to assign incorrect priorities to incoming requests. As a result, this combination of a real-time OS, a real-time capable Java, and Real-time RMI in conjunction with an appropriately trimmed JBoss framework results in a powerful basis for a real-time component-based system.

2. Implementation Detail

The core of the real-time component-based system as described in the previous sections is the Real-time Specification, the admission control, the policer, and the thread scheduler.

- *Real-tTime Service Specification* Each real-time component will expose its real-time service interface. The real-time service is defined as a number of ClassOfService objects. Class ClassOfService is defined as

```
class ClassOfService {
        int classID;
        Method[] groupOfMethods;
        ArrivalFunction arrival;
        long deadline;
        ... // methods not shown
}
```

  where Class ArrivalFunction defines the arrival function for the corresponding class of service.

- *Admission Control Module* The admission control module is implemented as an MBean. This MBean realizes the admission control mechanism and the decision making procedure. The resource table and the existing-task table required by admission control are implemented as entity beans. Service optimization and service adaptation are implemented as sub-modules.

- *Policing Module* After a task request is admitted, a task ID is generated and forwarded to the client. At the same time, a shaper instance (implemented as entity beans) is created. Any invocation of this task will add the task ID to the Invocation object at the Admission Interceptor on the client side. At the Shaping

Interceptor on the server side, the task ID is retrieved, and is used, together with the name of the EJB, as a key to match its corresponding shaper instance.

- *Utilization Allocation and Scheduling Module* As an example of a guaranteed-rate scheduler to provide temporal isolation, a Total Bandwidth Server (TBS) [43] is implemented to allocate the underlying CPU utilization to each real-time component. The input parameters for TBS are WCETs of methods in the component and the allocated CPU utilization. Recall that when a task is admitted, a corresponding shaper instance is initiated. The shaper instance also includes the priority assigned to any run-time method execution in the task. Once an invocation enters Shaping-Interceptor, its corresponding shaper instance can be found in the same way as in the policing module. Then, the priority value can be retrieved, and the priority for the worker thread is set.

F. Performance Evaluation

Recall that the two key foci of this dissertation are improving system reliability and provide real-time guarantee. For reliability, it has been widely studied and acknowledged that component based application development can greatly reduce the number of bugs per thousand lines of code. Our proposed real-time component based application is an extension and enhancement of existing component based methods and in turn, can greatly improve system reliability. In the experiments described below, we focus on the real-time guarantee. Specifically, we will evaluate the performance of our system in terms of the admission probability for each task request and of the latency overhead introduced in our system. In our experiments, we assume that the CPU on the EJB server is the bottleneck and that the other resources, such as memory, disk and network bandwidth, are never a limiting resource.

Admission Probability vs. Task Arrival Rate In this experiment, we choose two Pentium 4 machines with 2.53 GHz CPU and 1 GB memory as application servers. These two machines are in the same subnet together with another machine chosen to host as the clients. The network bandwidth for all connections is 100 Mbps. The application servers are installed with real-time component-based systems software. We deployed three real-time components $\{e_1, e_2, e_3\}$ – real-time session beans – in the first real-time application server and e_4 in the other real-time application server. Component e_i will expose one method e_{ek} and define a single class of service (therefore we can ignore the class index), and its real-time service interface is $\langle \Theta_{ek}, A_{ek}, D_{ek} \rangle$, where $\Theta_{ek} = \{\theta_{ek}\}$, and D_{ek} = 1.000 sec, for k = 1, . . . , 4. Methods are associated with WCET C_{e1} = 0.050 sec, C_{e2} = 0.080 sec, C_{e3} = 0.050 sec, and C_{e4} = 0.030 sec, respectively. The safe utilization for each processor at each application server is 90%. The arrival function $A_{ek}(I)$ will be optimally specified and CPU utilization α_{ek} will be determined with our service optimization algorithm. The runtime method execution will be assigned a single real-time priority. There are three task paths, as shown in Figure 5.

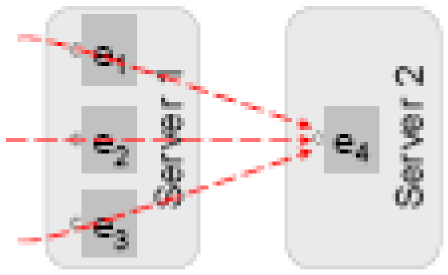


Figure 5. The Experiment Testbed

In this experiment, we choose three different system configurations in the application server: A system with no component isolation and enabled admission control (NCI), our system with component isolation and enabled admission control under service optimization (CI-OPT), and our system with component isolation and enabled admission control under service adaptation (CI-SA). In CI-SA, we choose $\Delta_{e,i} = 2A^T_{e,i}$. Emulated applications consist of task generators, and client-server interactions. Clients will send a sequence of periodic tasks. In each task, the period for invocation arrival is 2 sec and each invocation has a 2 sec deadline requirement. Each task has an exponentially-distributed life time and includes 6 invocations in a life time, on average. The task arrival is a Poisson process and each task will choose a uniform task path randomly. We vary the overall task arrival rate λ from 0.1 per sec to 1.5 per sec.

Figure 6 shows admission probabilities for all task admission requests from clients. As expected, as the task arrival rate increases, admission probability decreases in the systems with enabled admission control for all system configurations. The data show that CI-OPT has a lower admission probability than NCI with only a maximal difference

6.7% as λ = 1.0. With our introduced adaptation algorithm, the admission probability in CI-SA can even be improved close to that in NCI.

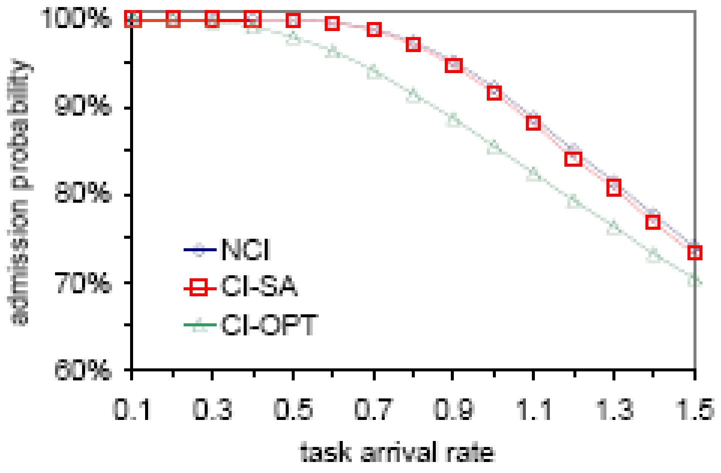


Figure 6. Comparison of Admission Probability vs.Task Arrival Rate

Admission Probability vs. Number of Components In this experiment, we deploy n real-time components in the first application server and none in the second application server. Each component exposes one method with WCET C_{ek} = 0.050 sec, k = 1, 2, · · ·, n. In each periodic task, the period for invocation arrival is 1 sec and each invocation has a deadline 1 sec requirement. We fix the overall task arrival rate as 2.0 per sec. The other configurations are same as the experiment above. We vary n from 2 to 10 and measure the admission probability for each n. Figure 7 shows that the admission probability in NCI keeps constant and the one in CI-OPT will decrease as the number of components increases. With the adaptation algorithm, the admission probability in CI-SA can still be improved close to the one in NCI.

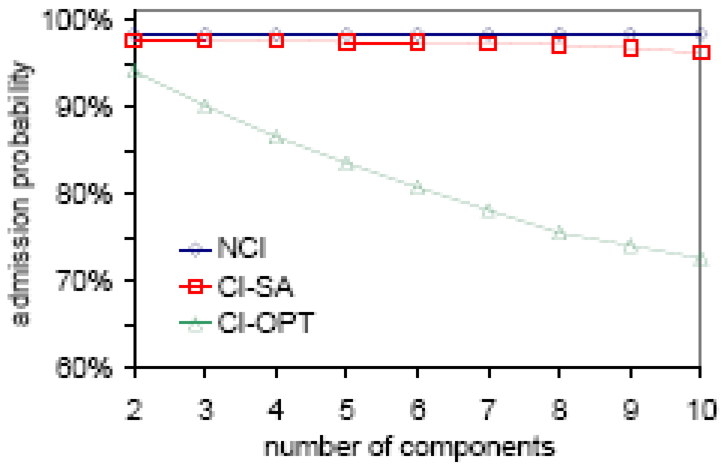


Figure 7. Comparison of Admission Probability vs. Number of Components

Admission Control Latency In the first experiment, we also collect the data about the latency of admission decision conducted by admission control. In Figure 8, the data shows that the average latency will increase slowly as the task arrival rate increases. Recall that the admission control module is implemented as an MBean and the involved resource table and existing-task table are implemented as entity beans. This implementation results in the admission control latency up to 0.035 sec in average as λ = 0.1. Since the queueing for the bursty task arrival may introduce an extra delay, the latency will increase slowly as the task arrival rate increases. The admission control latency only increases 2.02 times as the task arrival rate increases 15 times (from 0.1 to 1.5). Our admission control mechanism is quite scalable.

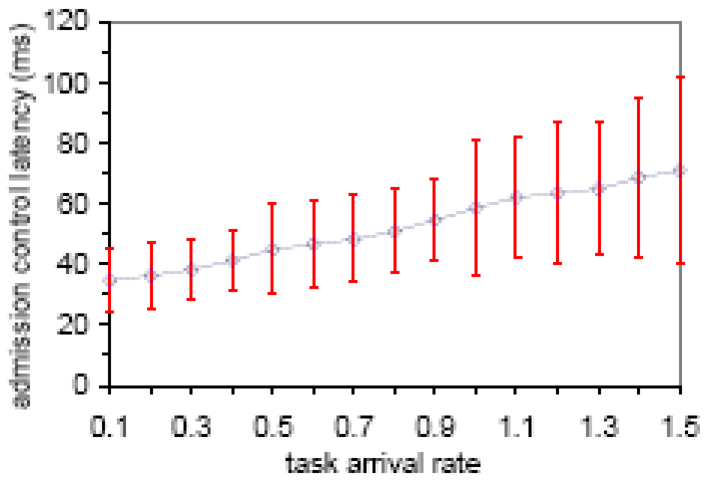


Figure 8. Admission Control Latency

In summary, through above detailed analysis and evaluation, we can see, with our proposed real-time component based application model and development methodology, that we can greatly improve system reliability and provide effective real-time deadline guarantees.

CHAPTER IV

REAL-TIME WORKFLOW-BASED APPLICATION INTEGRATION IN CLUSTER ENVIRONMENT

A. Overview

In this work, we focus on software design issues affecting system reliability and real-time guarantee of mission-critical system operated in a cluster environment.

In a cluster environment, the reliability of a mission-critical system is not only affected by the reliability of a single application, but, more importantly, by the integration methodology of the different heterogeneous application. Such integration is not an easy task since the integration process involves complex legacy model encapsulation, model interaction and invocation control, and dynamic resource allocation. The raditional manual method of integration using individual customized programs or scripts involves significant effort in understanding the underlying business logics, tremendous coding effort, and is difficult to maintain, especially mission-critical systems which require frequent adjustment of the application integration logic. Workflow-based application has been proven to be a feasible approach to simplify the integration practice and thus improve system reliability. However, though extensive research work has been done on workflow-based application integration, little has been done for cluster environment, and especially flexible workflow-based integration model and methodology that can provide real-time guarantees.

In this chapter, on top of the real-time component based application development model, we proposed a novel real-time workflow-based application model and a software design methodology for cluster environment. We developed the new methodology to integrate the legacy applications to build comprehensive applications and used it in the data center management system for the Southeastern Universities Research Association (SURA) project. We show that our model and methodology greatly simplifies the design of such a system and makes it easier to meet deadline requirements, while improving system dependability through reuse of fully tested legacy models.

B. Real-time Cluster Environment Application Model

1. Application Model

We consider that a cluster environment mission-critical application is composed of a set of real-time workflows, and each workflow consists of a set of real-time applications. The scheduling and execution of the workflow and applications is managed by backend resource and application managing subsystems, which dynamically monitor the total system resource usage, the current system workflow and application execution requirements, and allocate resource to different workflow and applications based on predefined scheduling rules.

2. Resource Management and Scheduling Model

The resource management and scheduling is responsible for maintaining the resource catalog and monitoring the status of the computing resources. It allows system administrators to allocate computing resources to workflows and scientists to transparently share their computing resources.

The resource management and scheduling ensures that simulations are finished before the deadline. It includes 1) Admission Control; 2) Timer Service; 3) Job Generator; 4) Scheduler; and 5) Job Dispatcher. Figure 9 shows the architecture of QoS components.

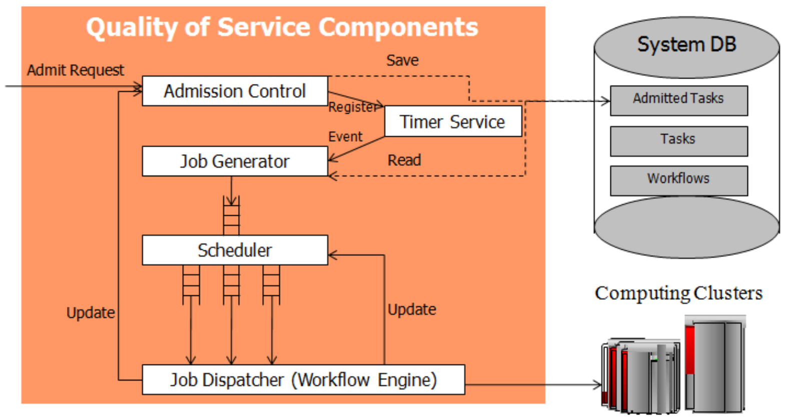


Figure 9. QoS Architecture

The Admission Control component is used to decide whether the new task meets its QoS contract while the admitted QoS contracts won't be violated by the new task. If the answer is yes, the new task is admitted and saved as admitted task in the system database. We use schedulability bound based admission control and r-shaped task model in this implementation [52]. A set of r-shaped tasks is schedulable if

$$\sum_{i=1}^{n} F_i(D_i)/D_i < 1/\lambda \qquad \text{(IV-1)}$$

where λ is the deadline inversion and is calculated as

$$\lambda = \max_{i=1,\ 2,\ \ldots,\ n}\left(\frac{\max_{j\leq i}(D_j)}{D_i}\right) \qquad \text{(IV-2)}$$

To perform the test, we need to calculate the deadline inversion ratio λ. Another important functionality of the admission control component is handling the task departure, e.g. deleted tasks, or finished tasks. The utilization reserved for these deleted tasks must be recycled. In our current implementation, this recycle will not happen until the system is idle. The reason for this is twofold: first, that it has been proven that the reserved resource of a departure task cannot be recycled at the time of departure since the task may have already used the resource, and secondly, recycling the reserved resource at the system idle time is more efficient and safe since the system can be treated as a “restart”.

The timer service is used to create and send an event to the job generator that is responsible for constructing the job. The admission control component will register the admitted task to the timer service when a new task is admitted and will remove the task from the timer service when the task is completed or cancelled. The timer service can be automatically initiated to guarantee that the admitted tasks can be dispatched when the data center management system is restarted.

The job generator is an asynchronous messaging service and is able to receive messaging event from the timer service. It is responsible for constructing the job once it receives a JobReady event from the timer service. The generated jobs are sent to the incoming job queue of the scheduler.

A static priority scheduler is implemented in the current data center management system. The scheduler maintains a job queue for each cluster and the jobs in the queue are arranged based on their priorities. That is, the jobs from the task with higher priority are inserted in front of the lower priority ones. After admission, each task is assigned to a specific cluster for execution. For each new job, the scheduler finds the cluster on which it will run and insert the job to the outgoing job queue corresponding to the cluster. Since the queue is ordered based on their priorities, the insert operations is very efficient using the binary search algorithm with a complexity of O(log(n)).

The job dispatcher is implemented as an active service that periodically checks the cluster status. The job dispatcher is implemented as part of workflow engine component in the data center management system. If the job dispatcher finds that the cluster is free (empty or partially empty), it will perform the following two operations:

- Notify the admission control and scheduler.
- Get the next job from the scheduler.

Note that the job dispatcher may not be able to get a job from the scheduler. If this is the case, the job dispatcher will switch to sleep mode until the next polling time.

C. Cluster Application Integration Methodology

1. Role-based System Design

In this section, we introduce our role-based design methodology of cluster environment software application integration.

Figure 10 shows the high level view of the role-based design. The top layer illustrates four kinds of users in the cluster environment: model developers, model integrators, system administrators and model users. The users manage and access resources through the middle layer. The backend management system provides different kinds of the tools to service different kinds of users. These include tools for model developers to develop the scientific computing models; tools for model integrators to integrate the legacy computing models into the comprehensive workflows; tools for model users to execute the workflows as well as retrieve and visualize the results; and tools for system administrators to manage user accounts and grant the roles and the resource access permissions to the users. The bottom layer includes the managed resources, scientific computing models and workflows, data, and high performance computing clusters / large volume storage.

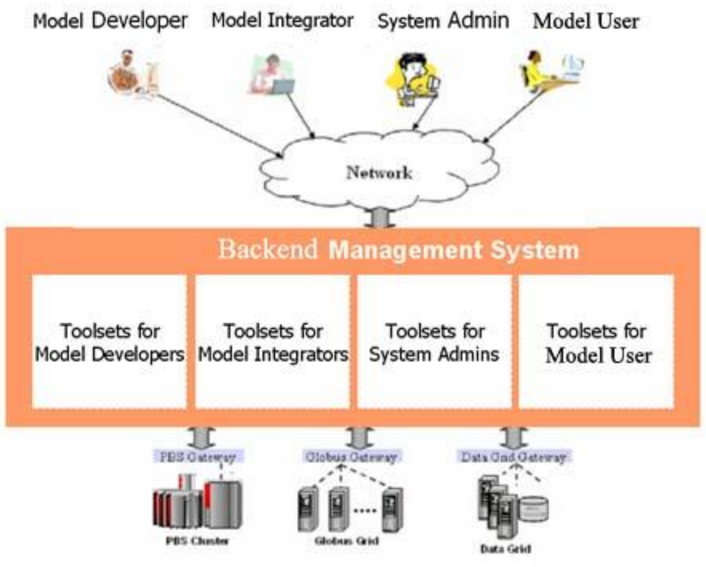


Figure 10. High Level View

2. Workflow Driven System Integration

Unlike traditional application integration methodology, in this dissertation, we propose a real-time workflow driven application integration methodology. In this novel methodology, the integration of applications is supported and managed by backend workflow facilitating components, including: 1) workflow builder; 2) application and workflow management tools; and 3) workflow engine.

The workflow builder helps model integrators to wrap the legacy applications to enable modules and build comprehensive simulation workflows that consist of one or several modules. It consists of two parts: a workflow builder and several EJB components to provide the access to the workflow modules and workflows stored in database.

The model integrators first use workflow builder to wrap the legacy applications as workflow modules that unify the applications with unified input/output interfaces (e.g. <module> -input <xml configuration file>, -output <output directory>), associate the applications with data products as part of the parameters in the input xml file and resolve deployment issues with script. The workflow module specification includes three sections, 1) The general section that provides the generic information about the wrapped applications, such as name, platform, version, description and etc. 2) The execution section that includes scripts to set environment variables, extracts the values of the parameters defined in the xml configuration file, converts parameters to application specific formats, use MPI or not, calls the application in its native command line format and etc.; and 3) The parameter section that defines the application parameters in xml schema. The parameters can be converted to html format to allow the end users to set the parameters using a web browser.

Once the workflow modules are defined, the model integrators can use workflow builder to create the workflows that consist of one or multiple workflow modules to construct different simulation scenarios. The workflow builder allows the model integrators to easily create the workflows as directed acyclic graphs (DAG). Like the workflow module, the workflow specification is defined using an xml schema. DAG can be re-constructed and displayed to end user based on the definition in xml.

A workflow created by a model integrator cannot be used by the model users until system administrators use the application and workflow management components to: 1) allocate hardware resources to the workflows, e.g. clusters and number of cluster nodes; 2) define the QoS parameters, such as priority (e.g. 1-100), worst case execution time (WCET) for each workflow module; and 3) assign the workflow to model users. A new non-admitted task is created after the administrator allocates resources, defines QoS parameters and assigns the workflow to the scientists.

Once the tasks are created by the system administrators, the model users can use a client side tool to remotely schedule the execution of the predefined and authorized tasks, and visualize and retrieve the simulation results. In order to guarantee the simulation results to be generated in a timely manner, model users need to define and submit the QoS contract (e.g. relative deadline, first execution time, frequency of the execution) for the selected workflows. The tasks admitted by the QoS components are called admitted tasks. Each execution of the admitted task is called a job, which includes the owner of the job, workflow specification, associated resource and QoS contract. When the task is admitted, the QoS contract is stored in the system database as part of the admitted task information and the admitted task is registered to the timer service. The timer service will automatically dispatch the jobs based on the QoS contracts to the job queue for execution.

Once the job is dispatched from the job queue, the workflow engine will parse the workflow of the job, run workflow modules in the allocated cluster and monitor the execution of the workflow modules.

With introduction of the workflow modules, which wrap the original command line applications as web enabled applications and handle the deployment issues (e.g. dependency on data and hardware, different argument formats of the original application), cluster environment application management system enables the model users to transparent access of data, applications, and computing resources remotely. It also facilitates the reuse of the existing software modules/applications. With introduction of the workflow, cluster environment management tools allows the model integrators to construct different research scenarios to meet the need of the users. The integrated QoS framework ensures that the simulation results can be generated in a timely manner. Since the concept and implementation of workflow modules and workflows is application generic, it makes the cluster environment application management and integration components adaptable and extendable to support other requirements.

3. Multi-layered System Architecture

Figure 11 shows 4-layer software architecture for cluster environment application integration. We will focus on the top 3 layers: User Interface Layer, Management and Integration Layer and Uniform Access Layer.

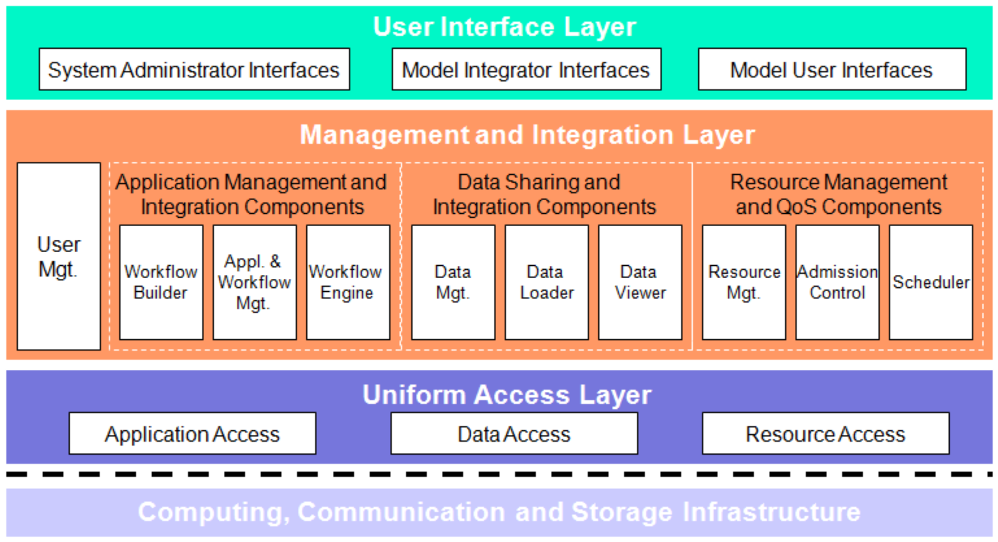

User Interface Layer
System Administrator Interfaces
Model Integrator Interfaces
Model User Interfaces
Management and Integration Layer
User Mgt.
Application Management and Integration Components
Workflow Builder
Appl. & Workflow Mgt.
Workflow Engine
Data Sharing and Integration Components
Data Mgt.
Data Loader
Data Viewer
Resource Management and QoS Components
Resource Mgt.
Admission Control
Scheduler
Uniform Access Layer
Application Access
Data Access
Resource Access
Computing, Communication and Storage Infrastructure

Figure 11. Software Layer Architecture

User Interface Layer provides separate interfaces for the system administrator, model integrator and scientist to manage and access the resources, applications and data in the data center. For example, the user interface includes system administrator interfaces for managing user account, resource, data, applications and workflows; it also includes model integrator interfaces for creating workflows; and includes scientist interfaces for scheduling and executing the workflows, accessing and viewing the data.

Management and Integration Layer contains a set of core service components. It includes four components including user management components, resource management and QoS components, application management and integration components, and data management and integration components.

The application management and integration components include key components, such as the application and workflow management component, workflow builder component and a workflow engine. The application and workflow management component is responsible for maintaining the application and workflow catalog. The workflow builder component is responsible for assisting the model integrator to build workflow specification. The workflow engine is responsible for parsing and executing the workflow specification.

The data management and integration include the data management component, data loader toolset and data viewer toolset. The data management component is responsible for maintaining the metadata and sources of the data. The data loader toolset is provided to upload and download data, while the data viewer toolset is provided to view the data.

The resource management and QoS components include the resource management component and QoS components, such as admission control and scheduler. The resource management component is responsible for maintaining the catalog and monitoring the status of the computing resources. The admission control service component is introduced here to guarantee the timely completion of the tasks in the system and the scheduler is the unit that resolves resource usage conflict.

Uniform Access Layer provides uniform interfaces to integrate with the existing computing, communication and storage infrastructure to allow the upper layer components to access the computing resources, application and data. The "Uniform Access Layer" also provides a mechanism to support new types of computing resources and data via configurable accessors, called drivers.

Computing, Communication and Storage Infrastructure Layer provides basic instruments to manage, store, and access computing resources, applications and data. Although it is not part of the integrated system design and implementation, our design of the platform allows the existing infrastructure to be easily integrated into the system and used by the scientific computing data center.

D. Implementation and Performance Evaluation

Based on the previous system model and design methodology, we build a real-world cluster environment system – data center management system for the Southeastern Universities Research Association (SURA) project. The system is implemented in a LINUX environment using J2EE technologies. Its design and implementation is based on 3-tier architecture. Figure 12 shows the implementation architecture of the SURA data center system.

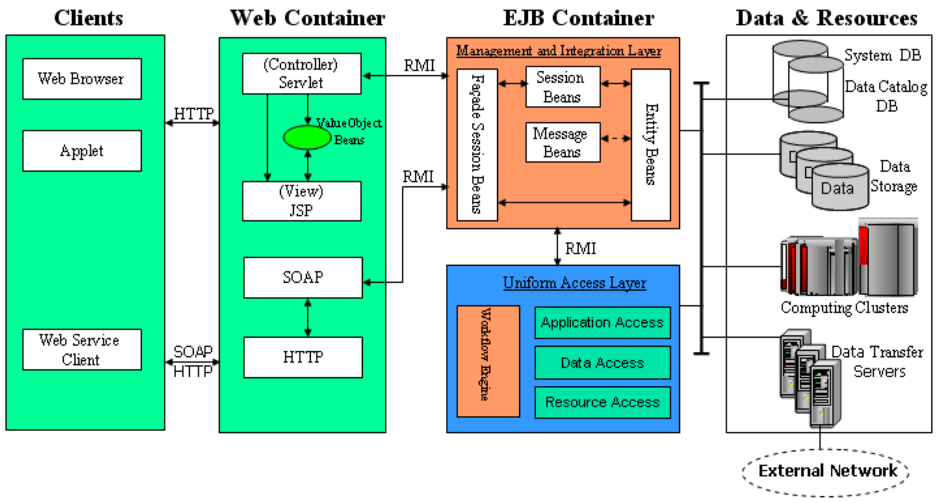


Figure 12. Implementation Architecture

The system is integrated with different kinds of open source platforms. In the first tier, it uses Struts/Tomcat/Axis as the web container that supports dynamic web interfaces and web services. In the second tier, it uses JBoss as the EJB container for the EJB components that provide the business logics of the data center management system. As part of the second tier, a set of RMI services are designed and implemented as proactive services to monitor and access the resources, e.g. hardware, data and applications. In the third-tier, it uses PostgreSQL database. The design of data center management allows us to easily switch to other relational database management systems.

The system is also integrated with other applications to manage hardware resources and transfer data, e.g. using Portable Batch Scheduling System (PBS) [53] to manage the clusters, using Unidata Local Data Manager (LDM) [54] to capture and distribute large volumes of data from/to other research labs and national data centers.

1. User Interface Layer

The SURA data center system basically provides a web-based interface for its users, especially scientists, to manage and access data center resources. The design and implementation of the web interface follow the Model-View-Controller (MVC) architecture to separate concerns and improve the sustainability and reusability of the data center software components. Apache Struts [55] is a free open-source framework to support the MVC architecture. The implementation of SURA data center system web-based interfaces is based on the Apache Struts framework.

The SURA data center system also uses Java Applet to support highly interactive operations with low latency tolerance, e.g. workflow builder GUI, visualization toolkits. The web services are provided to allow the third party applications and the other data centers applications to interact with the data center management system. For example, the data center data transfer service uses web services to trigger the update of the data catalog after new data are captured and archived in the data center storage; the third-party applications can use the web services to schedule the execution of the workflows.

2. Management and Integration Layer

Management and Integration Layer contains a set of core service components. It provides management and integration services. We address all the key issues of a data center when we design and implement the management and integration layer.

a) User Management

The user management is responsible for authenticating and authorizing users' access to data center resources. The user management components use Java Authentication and Authorization Service (JAAS) [56], [57]to separate the concerns of user authentication so that they can be managed independently. A user with single sign-on can be assigned to one or multiple roles, e.g. model integrator and system administrator.

The user management supports both descriptive role-based access control mechanism and programming group-based access control mechanism to provide different levels of access controls. The descriptive role-based access control mechanism is a coarse-grained security mechanism to control what kinds of components and methods can be accessed by each role. The descriptive role-based access control is not managed by the user management components. Its access control list (ACL) is defined in xml configuration files. The access control is enforced by the EJB container, JBoss, in run-time [58]. This greatly simplifies access control of a data center system. It also gives the administrators the capability to update the access control without re-programming the user management components. The programming group-based access control mechanism is a fine-grained security mechanism to control which resources, e.g. workflows (integrated applications), data and clusters, may be accessed by a group of users with the same or different roles.

With the MVC architecture on interface design and flexible access control mechanisms on user management, we can create different user interfaces and related toolsets with access control to support the separation of concerns.

b) Data Management and Integration

Data management and integration include key components, such as data management components, data loader toolset and data viewer toolset. The data management components manage two kinds of data, scientific computing data and their metadata, called data products. The scientific computing data are used or generated by scientific computing applications. Data products are metadata about the scientific computing data.

Data products are modeled by a generic model with common information, e.g. id, name, description, criteria (e.g. time, geographic location) for scientific computing data selection, location of the scientific computing data and access methods as described below. The data products are created by the (data) system administrator or programs and stored in the system database.

Scientific computing data can be stored in the relational database or the self-contained files with well-defined formats. If the data are stored in the self-contained files, the data loader toolset will extract the minimum information from the data files and store them as metadata so that the users or tools can find data based on the metadata and extract the data from the data files. As mentioned above, one of the key requirements for the SURA data center management system is to be adaptable and extendable to support other scientific research. In terms of data management and integration, it is impossible to develop a set of common data models to accommodate all scientific computing data from different research domains. It is even impossible to predefine a set of data models to accommodate all atmospheric data collected from radar sites or generated by the simulation applications as new simulation applications are developed and new data will be collected or generated for and by those new simulation applications. The most feasible solution to manage and integrate the heterogeneous information/data is to develop a federated architecture [59]. Figure 13 shows the federated architecture for data management and integration.

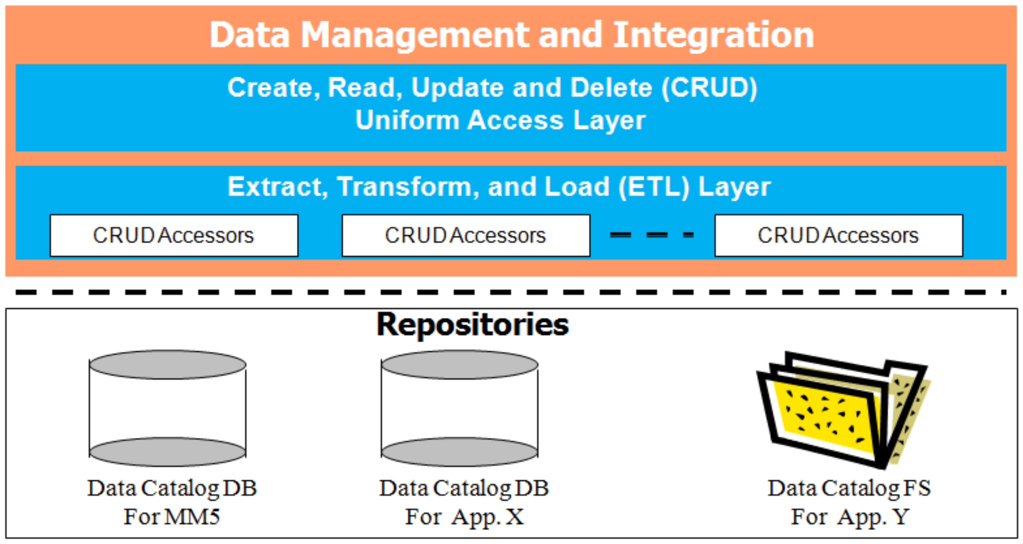
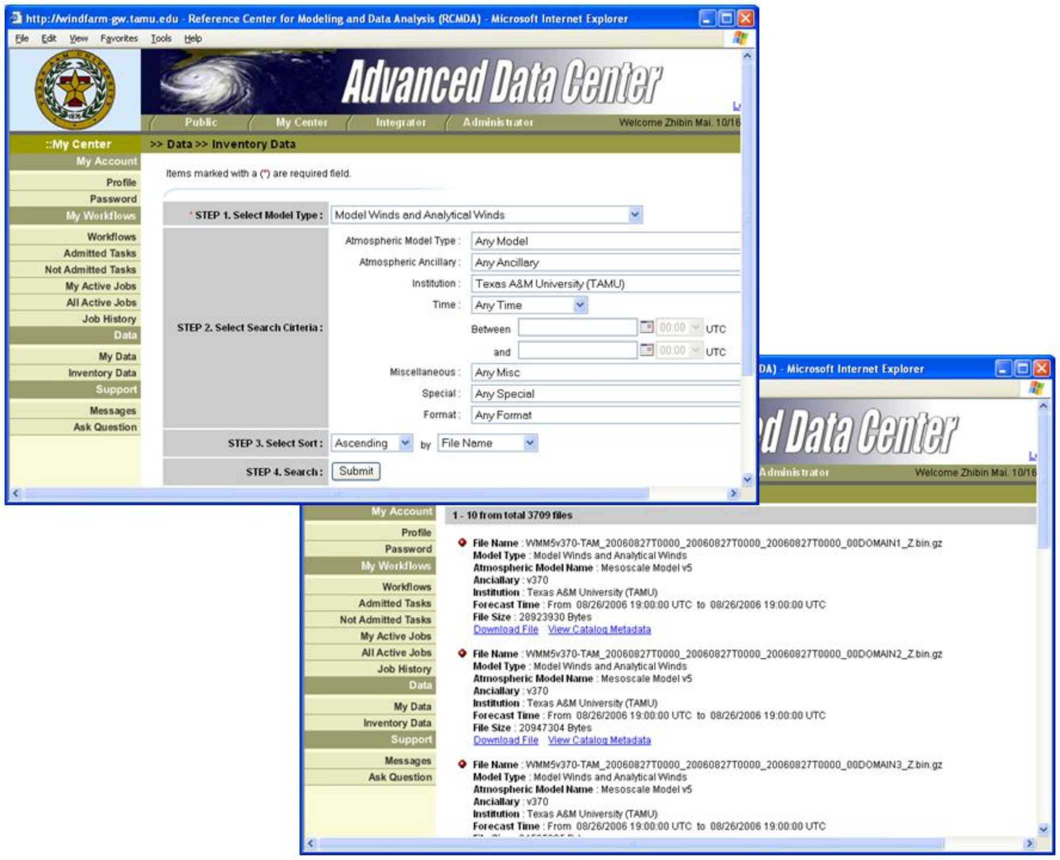

Data Management and Integration
Create, Read, Update and Delete (CRUD) Uniform Access Layer
Extract, Transform, and Load (ETL) Layer
CRUD Accessors
CRUD Accessors
CRUD Accessors
Repositories
Data Catalog DB For MM5
Data Catalog DB For App. X
Data Catalog FS For App. Y

Figure 13. Federated Architecture

The top layer is the uniform access layer for creating, reading, updating and deleting (CRUD) functions [60]. The middle layer is the extracting, transforming, and loading (ETL) [61]layer that includes a set of CRUD accessors. The CRUD accessors map and wrap the underlined heterogeneous repositories (e.g. database, file system, and applications) to provide common CRUD functions to service its upper layers. To improve the reusability of CRUD accessors, CRUD accessors are separated into two parts, data mapping and CRUD methods. The data mapping is specific to each data model or table of the underlining repositories while the CRUD methods can be generic for the same repositories. For example, we can develop four generic CRUD methods for the relational databases based on JDBC. Each data product should refer a set of CRUD accessors in order to provide information on how to access the scientific computing data. The bottom layer consists of the repositories that are used to store the scientific computing data and metadata. The repositories can be the database, files or applications. The CRUD accessors in the ETL layer are responsible for hiding the detail of the repositories and providing uniform access to the data in the repositories. In the SURA data center, most of the atmospheric data are stored as self-contained files and their metadata are stored in a relational database, called data catalog database (DB). Figure 14 shows search and retrieve data interfaces.



>> Data >> Inventory Data

Items marked with a (*) are required field.

* STEP 1. Select Model Type :		Model Winds and Analytical Winds
STEP 2. Select Search Cirteria :	Atmospheric Model Type :	Any Model
	Atmospheric Ancillary :	Any Ancillary
	Institution :	Texas A&M University (TAMU)
	Time :	Any Time
	Between	00:00 UTC
	and	00:00 UTC
	Miscellaneous :	Any Misc
	Special :	Any Special
	Format :	Any Format
STEP 3. Select Sort :	Ascending	by File Name
STEP 4. Search :		Submit

1 - 10 from total 3709 files

File Name : WMM5v370-TAM_20060827T0000_20060827T0000_20060827T0000_00DOMAIN1_Z.bin.gz
Model Type : Model Winds and Analytical Winds
Atmospheric Model Name : Mesoscale Model v5
Anciallary : v370
Institution : Texas A&M University (TAMU)
Forecast Time : From 08/26/2006 19:00:00 UTC to 08/26/2006 19:00:00 UTC
File Size : 28923930 Bytes
Download File View Catalog Metadata

File Name : WMM5v370-TAM_20060827T0000_20060827T0000_20060827T0000_00DOMAIN2_Z.bin.gz
Model Type : Model Winds and Analytical Winds
Atmospheric Model Name : Mesoscale Model v5
Anciallary : v370
Institution : Texas A&M University (TAMU)
Forecast Time : From 08/26/2006 19:00:00 UTC to 08/26/2006 19:00:00 UTC
File Size : 20947304 Bytes
Download File View Catalog Metadata

File Name : WMM5v370-TAM_20060827T0000_20060827T0000_20060827T0000_00DOMAIN3_Z.bin.gz
Model Type : Model Winds and Analytical Winds
Atmospheric Model Name : Mesoscale Model v5
Anciallary : v370
Institution : Texas A&M University (TAMU)
Forecast Time : From 08/26/2006 19:00:00 UTC to 08/26/2006 19:00:00 UTC

Figure 14. Search and Retrieve Data

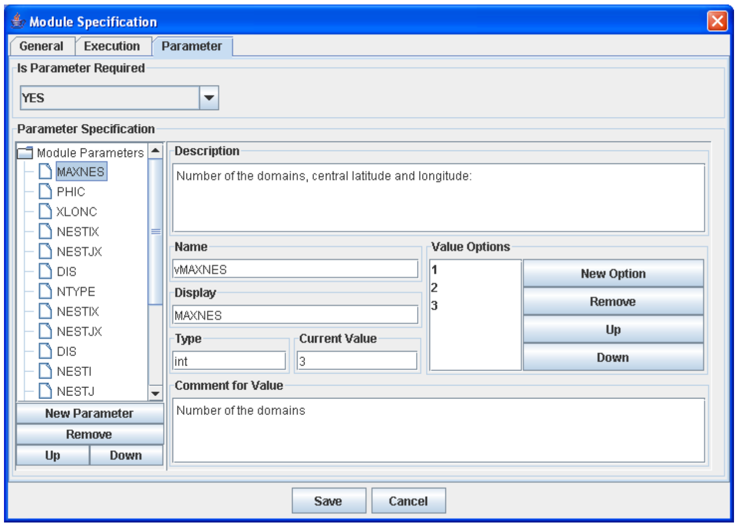
With the introduction of the data product concept that hides the detail on where and how to access the scientific computing data, the data center management system can provide the scientists transparency of sharing (scientific computing) data. The federated architecture design on managing and integrating data from heterogeneous repositories allows the data center system to be able to manage and integrate new data repositories to support new applications without changing the architecture of the data center management system.

c) Application Management and Integration

Most scientific computing applications are “legacy” command-line based applications. The applications could have different kinds of parameters and data as inputs. Some applications require special third-party libraries (e.g. Message Passing Interface (MPI)) and applications (e.g. data transfer application). In order to support the scientists to remotely run the legacy and heterogeneous applications in a timely manner without detailed knowledge of the application, data and hardware as well as their dependency, the application management and integration implements three kinds of major components: 1) workflow builder; 2) application and workflow management; and 3) workflow engine.

The workflow builder helps model integrators to wrap the legacy applications to web-based enable modules and build comprehensive simulation workflows that consist of one or several modules. It consists of two parts: a workflow builder GUI deployed as java applet in the first-tier and several EJB components in the middle tier to provide the access to the workflow modules and workflows stored in database.

The model integrators first use workflow builder to wrap the legacy applications as workflow modules that unify the applications with unified input/output interfaces (e.g. <module> -input <xml configuration file>, -output <output directory>), associate the applications with data products as part of the parameters in the input xml file and resolve deployment issues with script.. Figure 15 shows a screenshot of the workflow module specification GUI of the workflow builder. The workflow module specification GUI includes three sections, 1) the general section that provides the generic information about the wrapped applications, such as name, platform, version, description and etc. 2) The execution section that includes scripts to set environment variables, extract the values of the parameters defined in the xml configuration file, convert parameters to application specific formats, use MPI or not, call the application in its native command line format and etc.; and 3) The parameter section that defines the application parameters in xml schema. The parameters can be converted to html format to allow the scientists to set the parameters using a web browser. Figure 16 demonstrates the auto-conversion of xml format to html format for the workflow module parameters.



Is Parameter Required: YES

Parameter Specification

Module Parameters
- MAXNES
- PHIC
- XLONC
- NESTIX
- NESTJX
- DIS
- NTYPE
- NESTIX
- NESTJX
- DIS
- NESTI
- NESTJ

Description: Number of the domains, central latitude and longitude:

Name: vMAXNES

Display: MAXNES

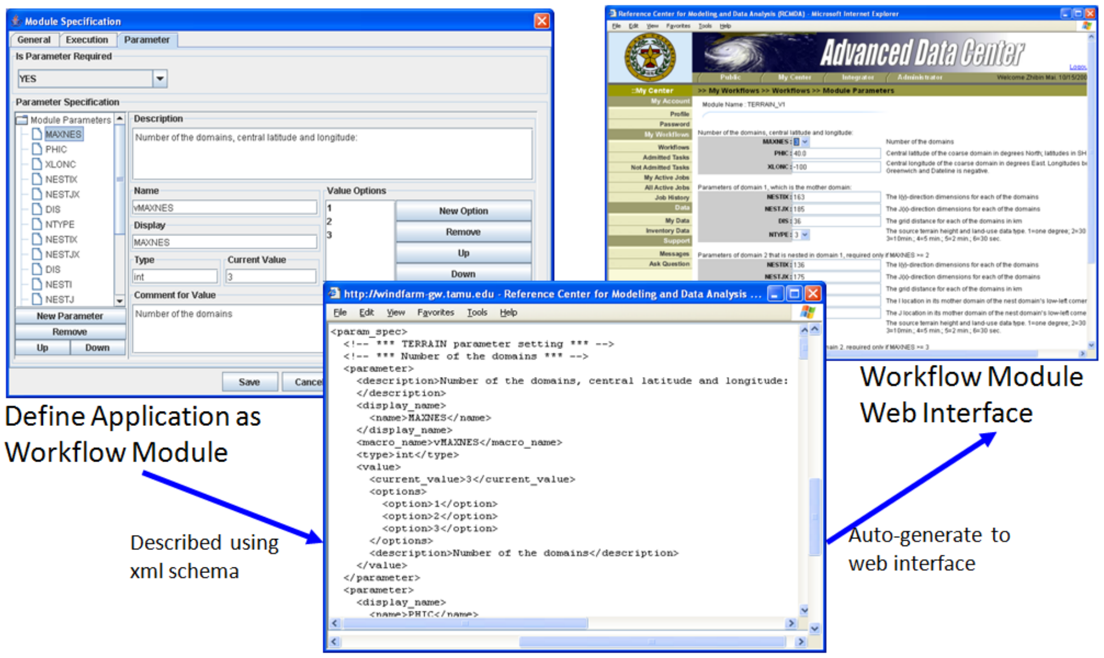
Type: int

Current Value: 3

Value Options: 1 2 3

Comment for Value: Number of the domains

Figure 15. Module Specification



```
<param_spec>
  <!-- *** TERRAIN parameter setting *** -->
  <!-- *** Number of the domains *** -->
  <parameter>
    <description>Number of the domains, central latitude and longitude:
    </description>
    <display_name>
      <name>MAXNES</name>
    </display_name>
    <macro_name>vMAXNES</macro_name>
    <type>int</type>
    <value>
      <current_value>3</current_value>
      <options>
        <option>1</option>
        <option>2</option>
        <option>3</option>
      </options>
      <description>Number of the domains</description>
    </value>
  </parameter>
  <parameter>
    <display_name>
      <name>PHIC</name>
```

Figure 16. Conversion of Workflow Module XML to HTML

Once the workflow modules are defined, the model integrators can use the workflow builder to create the workflows that consist of one or multiple workflow modules to construct different simulation scenarios. The drag & drop feature of the workflow builder allows the model integrators to easily create the workflows as directed acyclic graphs (DAG). Figure 17 shows the workflow builder interface that defines workflow as DAG. Like the workflow module, the workflow specification is defined using xml schema. DAG can be re-constructed and displayed in web interface based on the definition in xml. Figure 18 demonstrates the process.

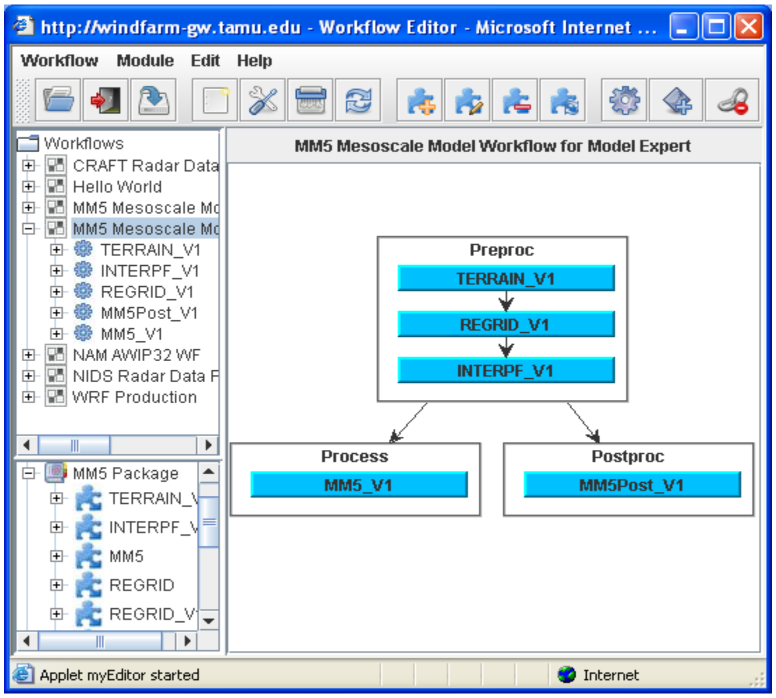
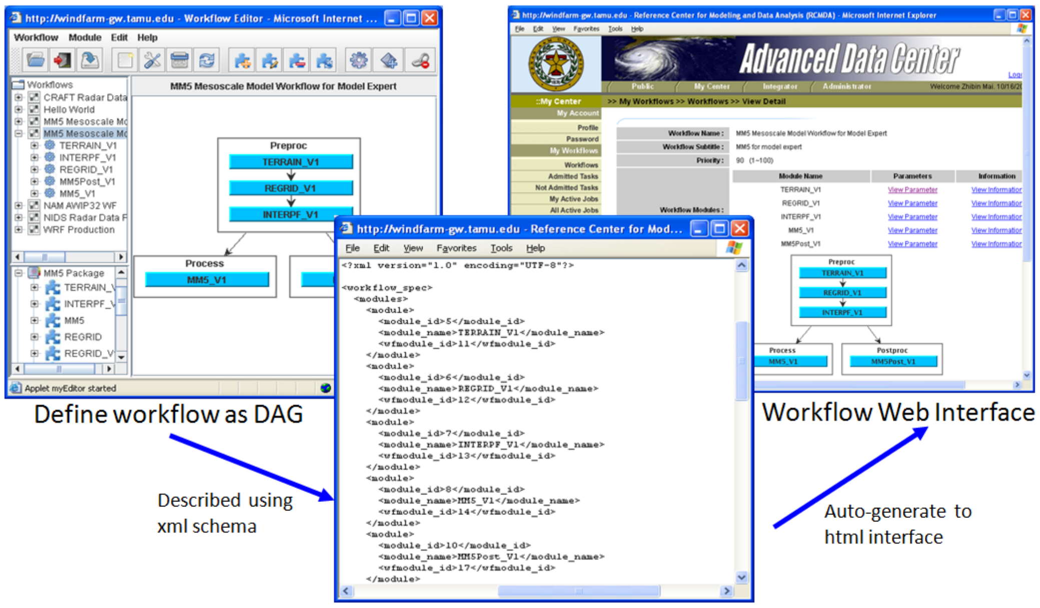


Figure 17. Define Workflow as DAG



```
<?xml version="1.0" encoding="UTF-8"?>

<workflow_spec>
  <modules>
    <module>
      <module_id>5</module_id>
      <module_name>TERRAIN_V1</module_name>
      <wfmodule_id>11</wfmodule_id>
    </module>
    <module>
      <module_id>6</module_id>
      <module_name>REGRID_V1</module_name>
      <wfmodule_id>12</wfmodule_id>
    </module>
    <module>
      <module_id>7</module_id>
      <module_name>INTERPF_V1</module_name>
      <wfmodule_id>13</wfmodule_id>
    </module>
    <module>
      <module_id>8</module_id>
      <module_name>MM5_V1</module_name>
      <wfmodule_id>14</wfmodule_id>
    </module>
    <module>
      <module_id>10</module_id>
      <module_name>MM5Post_V1</module_name>
      <wfmodule_id>17</wfmodule_id>
    </module>
```

Figure 18. Conversion of Workflow XML to HTML

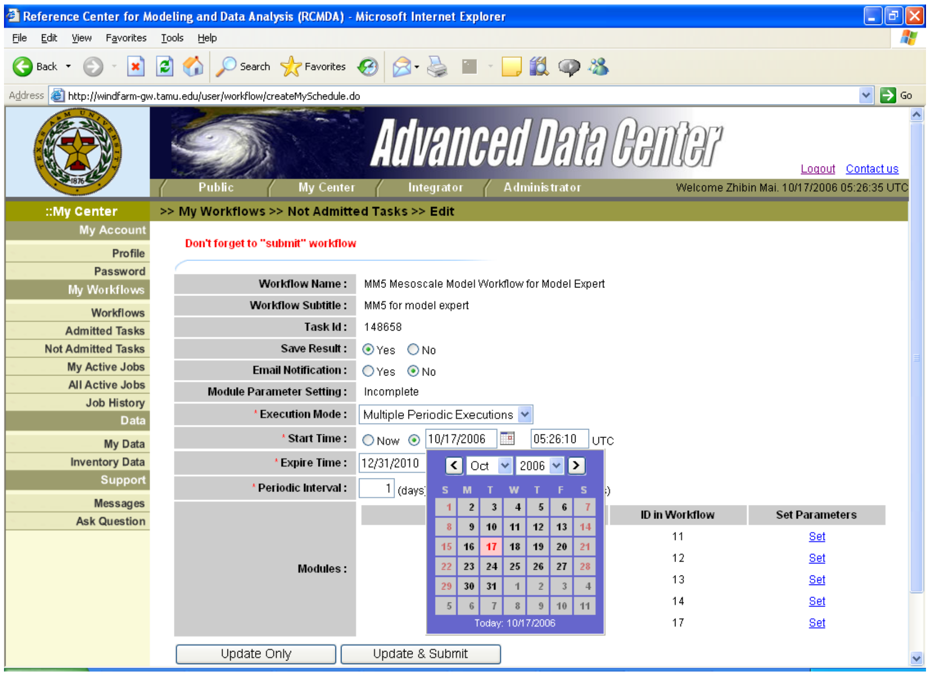
A workflow created by a model integrator cannot be used by the scientists until system administrators use the application and workflow management components to: 1) allocate hardware resources to the workflows, e.g. clusters and number of cluster nodes; 2) define the QoS parameters, such as priority (e.g. 1-100), worst case execution time (WCET); and 3) assign the workflow to scientists. A new non-admitted task is created after the administrator allocates resources, defines QoS parameters and assigns the workflow to the model users.

Once the tasks are created by the system administrators, the scientists can use a web browser to remotely schedule the execution of the predefined and authorized tasks, and visualize and retrieve the simulation results. In order to guarantee the simulation results to be generated in a timely manner, scientists need to define and submit the QoS contract (e.g. relative deadline, first execution time, frequency of the execution) for the selected workflows. The tasks admitted by the QoS components are called admitted tasks. Each execution of the admitted task is called job, which includes the owner of the job, workflow specification, associated resource and QoS contract. When the task is admitted, the QoS contract is stored in the system database as part of the admitted task information and the admitted task is registered to the timer service. The timer service will automatically dispatch the jobs based on the QoS contracts to the job queue for execution.

Once the job is dispatched from the job queue, the workflow engine will parse the workflow of the job, run workflow modules in the allocated cluster and monitor the execution of the workflow modules.

With the introduction of the workflow modules, which wrap the original command line applications as web enabled applications and handle the deployment issues (e.g. dependency on data and hardware, different argument formats of the original application), the data center management system enables the scientists transparent access of data, applications, and computing resources remotely. It also facilitates the reuse of the existing software modules/applications. With introduction of the workflow, data center management allows the model integrators to construct different research scenarios to meet the need of the scientists' research. The integrated QoS framework ensures that the simulation results can be generated in a timely manager. Since the concept and implementation of workflow modules and workflows is application-generic, it makes the data management and integration components adaptable and extendable to support other scientific research.

As the summary of this section, Figures 19-22 show the interfaces to support the scientists to run the workflows using web browser.



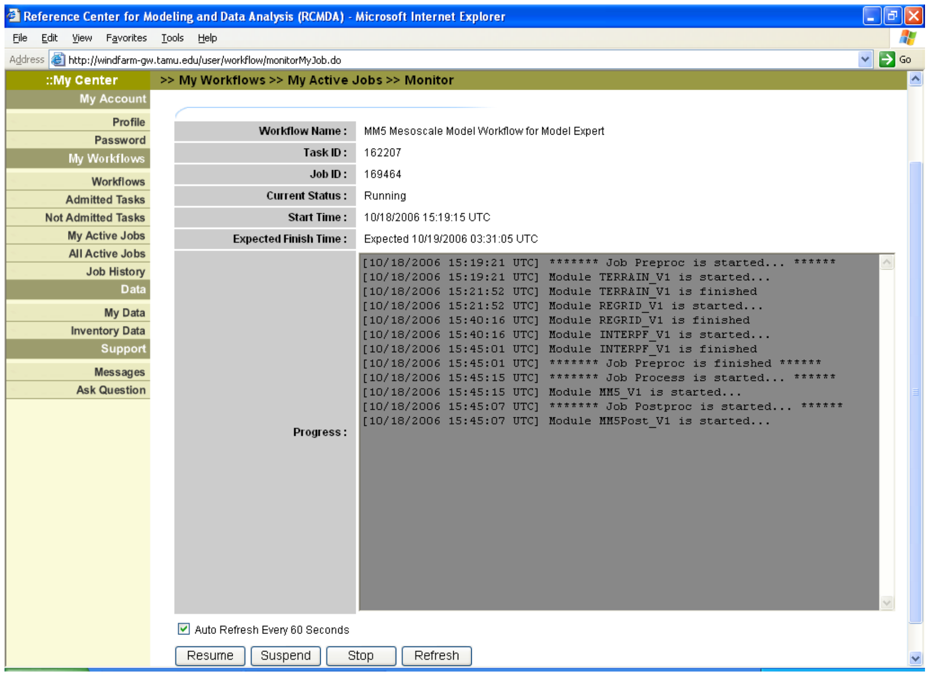
>> My Workflows >> Not Admitted Tasks >> Edit

Don't forget to "submit" workflow

Workflow Name :	MM5 Mesoscale Model Workflow for Model Expert
Workflow Subtitle :	MM5 for model expert
Task Id :	148658
Save Result :	[✓ Yes] No
Email Notification :	Yes [✓ No]
Module Parameter Setting :	Incomplete
* Execution Mode :	Multiple Periodic Executions
* Start Time :	Now [✓ 10/17/2006] 05:26:10 UTC
* Expire Time :	12/31/2010
* Periodic Interval :	1 (days)

ID in Workflow	Set Parameters
11	Set
12	Set
13	Set
14	Set
17	Set

Figure 19. Set Module Parameters and Schedule the Workflow



>> My Workflows >> My Active Jobs >> Monitor

Workflow Name :	MM5 Mesoscale Model Workflow for Model Expert
Task ID :	162207
Job ID :	169464
Current Status :	Running
Start Time :	10/18/2006 15:19:15 UTC
Expected Finish Time :	Expected 10/19/2006 03:31:05 UTC

Progress :

```
[10/18/2006 15:19:21 UTC] ******* Job Preproc is started... ******
[10/18/2006 15:19:21 UTC] Module TERRAIN_V1 is started...
[10/18/2006 15:21:52 UTC] Module TERRAIN_V1 is finished
[10/18/2006 15:21:52 UTC] Module REGRID_V1 is started...
[10/18/2006 15:40:16 UTC] Module REGRID_V1 is finished
[10/18/2006 15:40:16 UTC] Module INTERPF_V1 is started...
[10/18/2006 15:45:01 UTC] Module INTERPF_V1 is finished
[10/18/2006 15:45:01 UTC] ******* Job Preproc is finished ******
[10/18/2006 15:45:15 UTC] ******* Job Process is started... ******
[10/18/2006 15:45:15 UTC] Module MM5_V1 is started...
[10/18/2006 15:45:07 UTC] ******* Job Postproc is started... ******
[10/18/2006 15:45:07 UTC] Module MM5Post_V1 is started...
```

[✓] Auto Refresh Every 60 Seconds

Figure 20. Monitor Execution of the Workflow

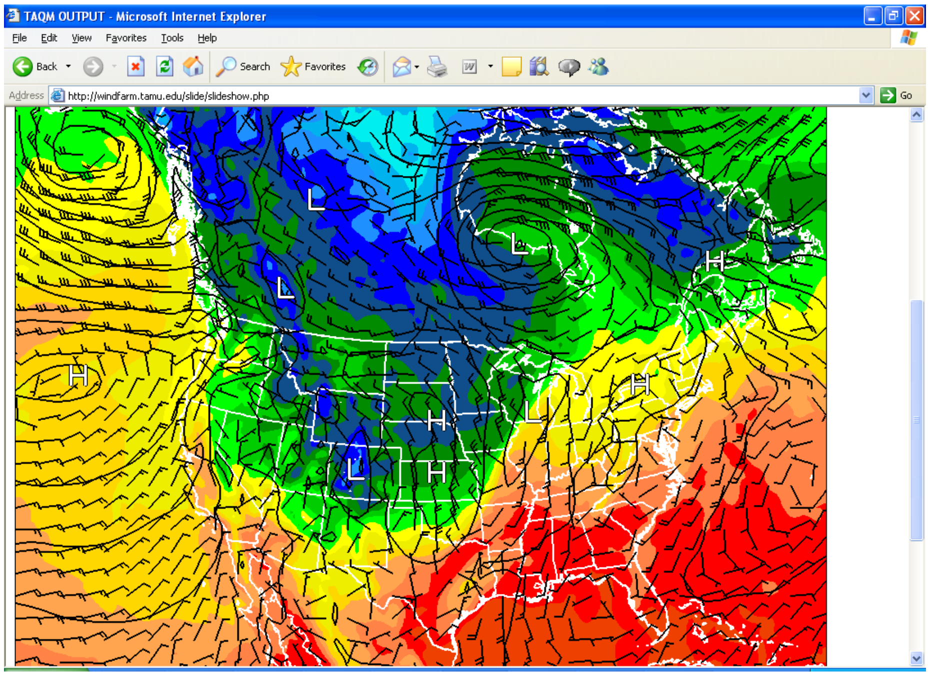
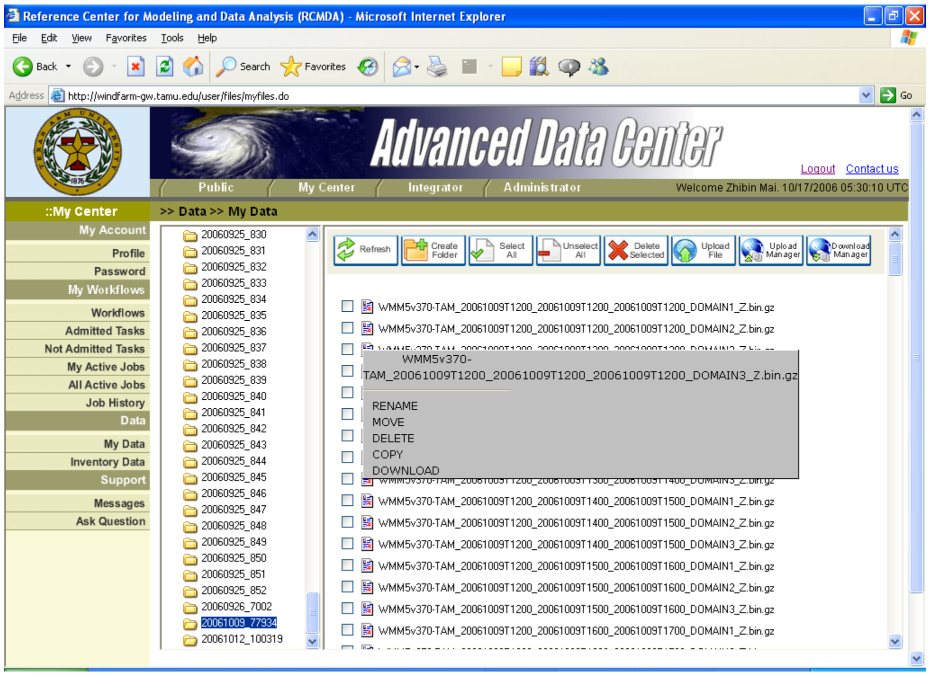

TAQM OUTPUT - Microsoft Internet Explorer
http://windfarm.tamu.edu/slide/slideshow.php
L
L
L
H
H
H
H
L
H
L

Figure 21. Visualize Results



Welcome Zhibin Mai. 10/17/2006 05:30:10 UTC

>> Data >> My Data

20060925_830
20060925_831
20060925_832
20060925_833
20060925_834
20060925_835
20060925_836
20060925_837
20060925_838
20060925_839
20060925_840
20060925_841
20060925_842
20060925_843
20060925_844
20060925_845
20060925_846
20060925_847
20060925_848
20060925_849
20060925_850
20060925_851
20060925_852
20060926_7002
20061009_77934
20061012_100319

WMM5v370-TAM_20061009T1200_20061009T1200_20061009T1200_DOMAIN1_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1200_20061009T1200_DOMAIN2_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1200_20061009T1200_DOMAIN3_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1400_20061009T1500_DOMAIN1_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1400_20061009T1500_DOMAIN2_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1400_20061009T1500_DOMAIN3_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1500_20061009T1600_DOMAIN1_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1500_20061009T1600_DOMAIN2_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1500_20061009T1600_DOMAIN3_Z.bin.gz
WMM5v370-TAM_20061009T1200_20061009T1600_20061009T1700_DOMAIN1_Z.bin.gz

Figure 22. Disseminate the Results

d) Resource Management and Quality of Service

The resource management component is responsible for maintaining the resource catalog and monitoring the status of the computing resources. It allows system administrators to allocate computing resources to workflows and scientists to transparently share the computing resources.

As mentioned in the previous section, “Application Management and Integration”, scientists need to submit QoS contracts to ensure that the simulation (job) can be finished in a timely manner, as described in the contract. The details on how to provide real-time guarantee to workflow applications in the cluster environment was introduced in the previous section, “Resource Management and Scheduling Model”.

3. Uniform Access Layer

Uniform Access Layer provides uniform interfaces to integrate with the existing computing, communication and storage infrastructure to allow the upper layer components to access the computing resources, application and data.

The federated architecture and CRUD accessors in the ETL sublayer provide the foundation to access the heterogeneous data from different repositories. The workflow modules with their execution scripts provide the foundation to execute the heterogeneous applications with different formats and environmental dependency. The cluster adaptors implement a common resource monitoring interface. The factory design pattern used for cluster adaptors allows upper layer components in the data center management system to access and monitor the heterogeneous computing resources using uniform methods.

E. Deployment

The SURA data center management system has been deployed in the Texas A&M SURA data center. Figure 23 shows part of the hardware resource managed by the SURA data center management system in the data center. With the SURA data center management system, the SURA atmospheric research scientists can remotely archive and share Terabytes of atmospheric data and run the simulations for real-time prediction and visualization of the impacts of extreme atmospheric events with transparency of data store infrastructure, computing infrastructure and application storage infrastructure in the Texas A&M SURA data center. They can use the web-based interfaces or web service interfaces of the data center management systems to access the integrated atmospheric applications of the data center. The data center management system provides delay guarantee on the simulation application with its utilization-based admission control components and static priority scheduler.

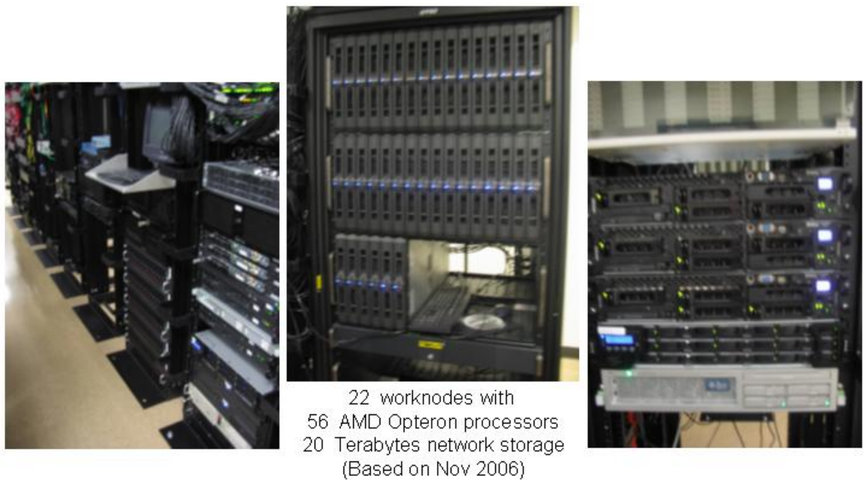


Figure 23. Hardware Deployment in Texas A&M SURA Data Center

With real-time workflow-based application integration, the SURA data center management system not only was able to provide scalable real-time guarantees for cluster environment applications, but was also able to support the integration of legacy models, thus facilitating the improvement of system reliability though reuse of fully tested components or software modules. A portion of the research results have been published in [62].

CHAPTER V

REAL-TIME VOIP SYSTEM

A. Overview

In this work, we focus on software design issues affecting integration and real-time guarantee of mission-critical system operated in a wide-area network. Given that there are many different kinds of wide area networked applications and study all of them in a single dissertation is almost impossible. In this dissertation, we select one of the most representative applications, the Voice over Internet Protocol (VoIP) system, for study. For this VoIP system, we will further narrow our focus to the real-time deadline guarantees since the reliability issues have already been addressed in previous two chapters.

We proposed a novel VoIP system model, a new design methodology, and based on the model and methodology, we implemented a demonstration system. We show that our new model and design methodology can effectively provide real-time guarantee for networked applications. With this mechanism, we can provide QoS provisioning system for VoIP. Our system integration methodology makes it feasible to integrate the QoS provisioning system with the Commercial Off-The-Shelf (COTS) VoIP systems to provide real-time VoIP service.

B. A General Model of VoIP Systems

VoIP systems are rapidly gaining acceptance. Currently, some of the leading vendors have made announcements about their strategies and product directions for such systems. Some VoIP systems, such as Cisco's and Alcatel's VoIP systems, have been put on the market. The existing VoIP systems aim to provide a certain degree of QoS to voice over IP networks. However, none of these systems can provide end-to-end QoS guarantees. In the following, we introduce a general architecture of the existing commercial VoIP systems and illustrate why these VoIP systems cannot provide QoS guarantees.

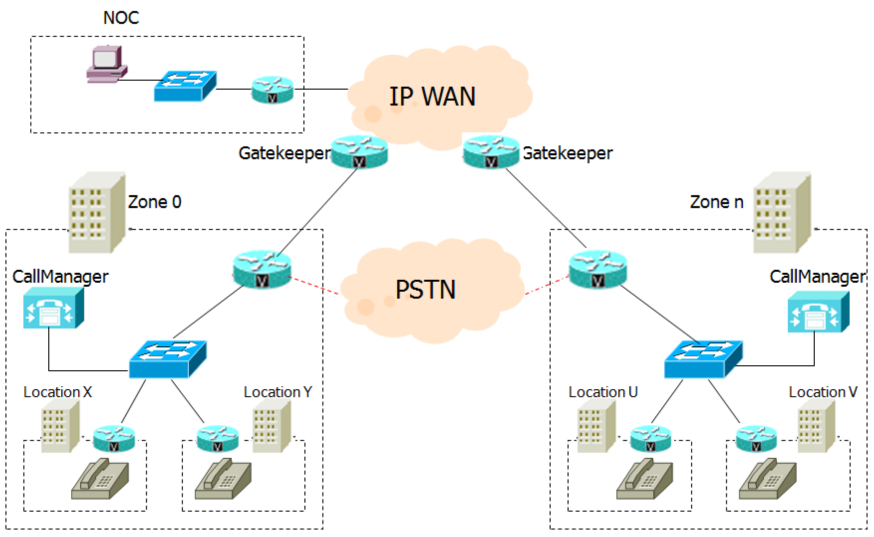


Figure 24. Typical Architecture of Existing Multisite VoIP Systems

Figure 24 illustrates a typical multisite architecture of the VoIP system. CallManager and Gatekeeper are the main components in the architecture. CallManager provides the overall framework for communication within a corporate enterprise environment [5]. Gatekeeper can provide services, such as address translation and call admission control, to the calls. These two components communicate with each other by using the H.323 signaling protocol [63], [64]. In Figure 24, a location defines a topological area connected to other areas by links with limited bandwidth registered to a CallManager. A zone is a collection of H.323 endpoints[6] that registers to the same Gatekeeper.

CallManager, as well as Gatekeeper, performs admission control for calls between locations in a zone or calls between zones, aiming to provide a certain degree of QoS to voice over IP networks. To call within a zone, only the CallManager located in the enterprise environment is invoked to perform CAC. However, for a call traversing multiple zones, not only CallManagers (both in the environment where the call is originated and in the one where the call is terminated), but also the related Gatekeeper(s) may be involved to perform CAC.

Utilization-based CAC is one type of CAC mechanism. It makes an admission control decision based on a predefined utilization bound: For each call request, as long as the used resource utilization plus the requested resource utilization are not beyond the predefined resource utilization bound that is computed offline and set at the configuration time, the service guarantee can be provided. Two kinds of Utilization-

[5] We adopt Cisco CallManager term to refer to the component that manages calls within a corporate enterprise environment, which can be a proprietary product such as Cisco CallManger or a standard H.323 Gatekeeper.
[6] An H.323 endpoint can be a H.323 terminal, a gateway, or a CallManager, which represent a corporate enterprise environment.

based CAC mechanisms, Site-Utilization-Based CAC (SU-CAC) and Link-Utilization-Based CAC (LU-CAC), have been adopted by some VoIP systems.

The basic idea of SU-CAC is to perform CAC based on the bandwidth which is preallocated to the sites. In this strategy, the site can be a location to the CallManager or a zone to the Gatekeeper. Bandwidth preallocation to sites is performed at the configuration time (i.e., at offline). A new call can be admitted if there is enough bandwidth left for the related site, otherwise, the call will be rejected. The core of SU-CAC is how to do bandwidth preallocation to the sites. Bandwidth preallocation (or provisioning) determines the certainty of QoS that a VoIP system can provide to voice over IP networks. Unfortunately, so far, the existing VoIP systems do not define a proper way to do that. Currently, bandwidth preallocation is performed in an ad-hoc manner. As a matter of fact, this is the reason why the end-to-end QoS guarantees cannot be achieved in current VoIP systems. The main advantage of SU-CAC is that it is simple and it enables CAC to be performed in a distributed fashion. It neither sends probes to test the availability of resources nor dispatches messages to make reservations. However, since the bandwidth has been preallocated to the sites at the configuration time, links cannot be fully shared by dynamic calls and, accordingly, high network resource utilization cannot be achieved (in this paper, the network resource is mainly referred to the link bandwidth). The Link-Utilization-Based CAC (LU-CAC) aims to address this issue.

The main idea of LU-CAC is to perform CAC directly based on availability of the individual link bandwidth. With this mechanism, call multiplexing can be performed at the link level, hence, high network resource utilization can be obtained. The disadvantage of LU-CAC is its complexity. Current VoIP systems have to rely on the resource reservation protocols, such as RSVP, to perform explicit resource reservation within the whole network. To achieve that, all the routers within the network should support resource reservation, which is not practical. Also, in current high speed networks, there are potentially thousands of flows passing through the core-routers. The overhead of the core routers within the network to support resource reservation can be significantly large. The overhead of resource reservation at the core-routers will compromise their main function, i.e., packet forwarding, which will degrade the whole network performance.

C. A New Qos-Provisioning VoIP System Design Methodology

1. Design Rationale

The goal of this study is to design and implement a practical, scalable, and highly efficient QoS-provisioning VoIP system that can provide end-to-end QoS guarantees to VoIP networks. Our main strategies to achieve this goal and address the challenging issues are listed as follows:

1) We build up our target system by enhancing the current existing VoIP systems, rather than build up a totally new VoIP system from scratch. The QoS architecture includes two planes: the data plane and the control plane. The data plane is responsible for packet scheduling and forwarding, while the control plane is for resource management and admission control. The simple utilization

test in utilization-based CAC (both SU-CAC and LU-CAC) can render the control plane scalable. We plan to enhance current VoIP systems by integrating a new QoS-provisioning system to enable both SU-CAC and LU-CAC to be well utilized and supported. With this system, the overhead of resource reservation at the core routers will be pushed to the agents in the QoS-provisioning system, which overcomes the weakness of the current VoIP system in applying the LU-CAC mechanism. We will also address the issue of performing resource allocation to better support the SU-CAC mechanism.

2) We leverage our research results on absolute differentiated services in static-priority (SP) scheduling networks to provide scalable QoS guarantees to the VoIP system. The static-priority scheduler is classbased. Its overhead introduced to the data plane is small and independent from the flow population. The data plane is scalable with the static-priority scheduler. It is not the case for the guaranteed-rate schedulers, such as Weighted Fair Queuing (WFQ), which needs to maintain per flow information, hence making the data plane not scalable. However, the static-priority scheduler does not provide flow separation as the guaranteed-rate schedulers do. With the static-priority scheduler, the utilization-based admission control mechanism such as SU-CAC and LU-CAC cannot be directly applied to provide QoS guarantees. It cannot simply admit a new flow by checking the availability of utilization due to the flow interference with the static-priority scheduler. To account for such interference, the runtime overhead of admission control will be very large. In our previous research work, we

derived a novel utilization-based delay analysis technique in static-priority scheduling networks. With this technique, the runtime overhead to perform admission control is moved to the configuration time while sustaining scalability in the data plane. The existing VoIP systems do not specify a particular type of packet scheduler to be used in its data plane. This gives us room to select the proper scheduler for our purpose. We decided to use a static-priority scheduler in the data plane of the VoIP system. Particularly, all the voice traffic shares the same priority, which is higher than the one used to transmit the non-voice traffic. In this way, our previous research results on absolute differentiated services in static-priority scheduling networks can be applied directly.

3) We apply the linear programming approach to optimize the resource allocation in the control plane. As we know, the SU-CAC mechanism tends to underutilize the network resource. Care must be taken to prevent wasting too many resources in applying this CAC mechanism. In this study, we will use the linear programming approach to optimize the resource utilization while still providing the end-to-end guarantees with SU-CAC mechanism.

2. Integrated System Architecture

Having discussed the design rationale of our QoS-provisioning system, in the following, we will introduce the architecture of this system and its components. Figure 25 shows our QoS-provisioning system integrating with the commercial VoIP system. It consists of three kinds of components: QoS Manager (QoSM), Call Admission Control Agent (CACA), and Integration Component (IC).

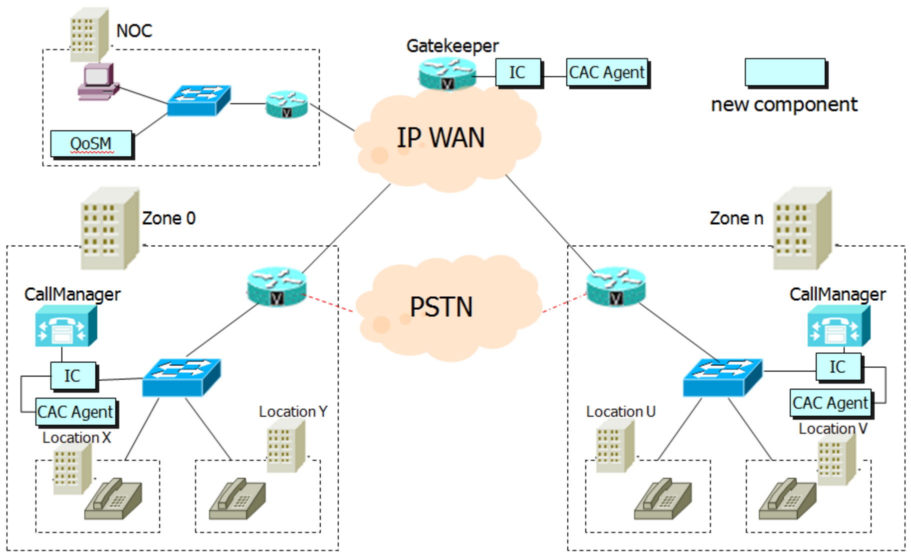


Figure 25. VoIP System with QoS Provisioning System

Figure 26 shows the components of the QoS-provisioning system. The main functions of these components are as follows:

1) QoS Manager (QoSM)—The QoSM implements three basic functions: 1) It provides user interface to control and monitor the components, which are in the same QoS domain. 2) It provides registration to the distributed agents and coordination among the distributed agents in the same QoS domain. 3) It cooperates with the peer QoSMs that belong to other QoS domains.

2) Call Admission Control Agent (CACA)—The CACA has two modules: 1) Utilization Computation Module, which performs deterministic or statistic delay

analysis to obtain the maximum bandwidth utilization. 2) Admission Decision Making Module, which performs admission control with specific CAC mechanisms.

3) Integration Component (IC)—The IC integrates CACA into existing VoIP systems and provides call signaling processing modules the ability to monitor and intercept call setup signaling from Gatekeeper or Call-Manager, withdraws the useful message and passes it to CACA, and executes call admission decision made by CACA.

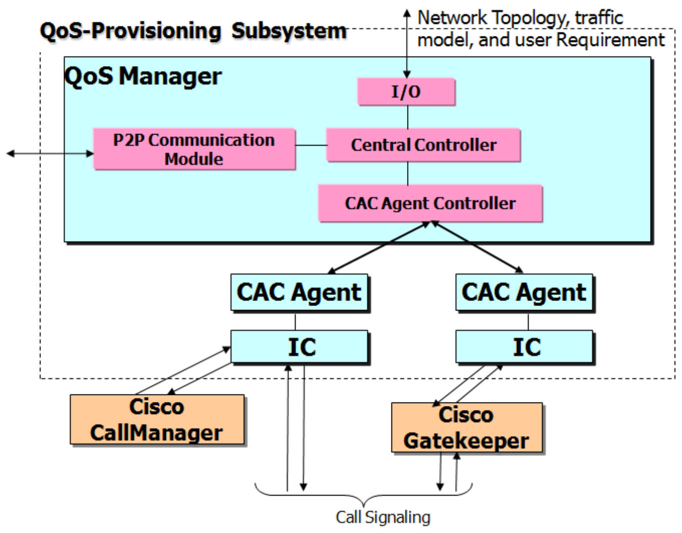


Figure 26. Components of the QoS-Provisioning System

It is the CACA that does delay analysis and makes admission control that are the main functions of the QoS Provisioning system to provide QoS guarantees for VoIP. Most of the challenging problems we encountered are in designing and implementing CACA. We will devote Section D to this important component. Our QoS-provisioning system is not a stand-alone system. It should integrate with the existing VoIP system. Such integration is conducted through the Integration Component (IC) which will be discussed in Section E.

D. Call Admission Control Agent (CACA)

The Call Admission Control Agent (CACA) is a key component in the QoS-provisioning system. It consists of two modules: Utilization Computation Module and Admission Decision Making Module. The utilization computation module performs delay analysis and computes the maximum bandwidth utilization. It usually runs at the configuration time. The computed utilization will be allocated to either links in the LU-CAC mechanism or to sites in the SU-CAC mechanism. At the runtime, the admission decision making module will make an admission decision for each incoming call request, based on the allocated bandwidth utilization (by the utilization computation module) and the currently consumed bandwidth. In the following, we discuss the details of these two modules.

1. Utilization Computation Module

The utilization computation module has two submodules: 1) the Link Utilization Computation Submodule and 2) the Site Utilization Computation Submodule. The first submodule is to compute the maximum link utilization for LU-CAC, while the second submodule is to compute the maximum site utilization for SU-CAC. The maximum link utilization is the maximum value of the link utilization under which the end-to-end delay can be guaranteed with LU-CAC. The maximum site utilization has a similar definition to SU-CAC. The above two submodules are closely related to each other. The maximum utilization allocated to sites is constrained by the maximum utilization allocated to the links since each pair of sites is connected by links. In the following, we will describe these submodules in detail.

a) Link Utilization Computation Submodule

The main task of this submodule is to compute the maximum link utilization for LU-CAC by calling a procedure, named the utilization verification procedure. It is shown in Figure 27.

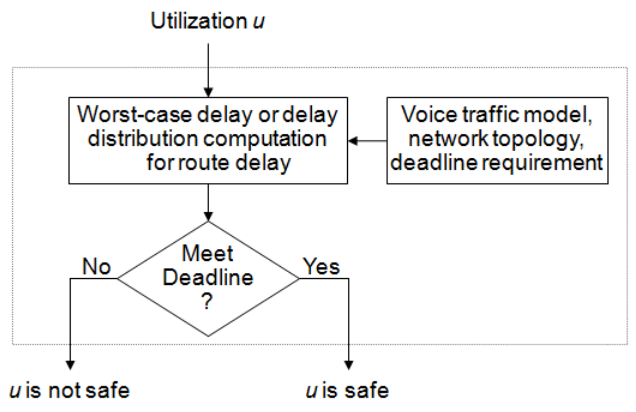

Utilization *u*
Worst-case delay or delay distribution computation for route delay
Voice traffic model, network topology, deadline requirement
Meet Deadline ?
No
Yes
u is not safe
u is safe

Figure 27. The Utilization Verification Procedure

Given the voice traffic model, the network topology, and the voice traffic deadline requirement, for any input of link utilization u, we compute the worst-case delay (deterministic case) or delay distribution (statistical case) with our delay analysis methods. Then, we can verify whether or not the utilization is safe in order to make the end-to-end delay meet the deadline requirement. Using the binary searching method for utilization, we can obtain the maximum link utilization. As shown, the most challenging thing is to do delay analysis.

Generally, there are two distinct types of delays suffered by a voice packet from source to destination: fixed and variable. Fixed delays include propagation delay, transmission delay, and so on. Variable delays arise from queuing delays in the output buffers. All fixed delays can be obtained by well-known experimental data or by using existing tools. However, it is difficult to obtain the variable delays. There is a significant amount of research on queuing delay analysis [65], [66] for both deterministic and statistical delay analysis.

Recall that we consider the VoIP system where static priority scheduling is used and voice traffic is assigned the highest priority. This scheduling does not provide flow separation. The local queuing delay at each output queue depends on detailed information (number and traffic characteristics) of other flows, both at the output queue under consideration and at the output queues upstream. Therefore, all the calls currently established in the network must be known in order to compute queuing delays. Delay formulas for this type of system have been derived for a variety of scheduling algorithms. While such formulas could be used (at quite some expense) for flow establishment at the runtime, they are not applicable for delay computation during the configuration time, as they rely on information about flow population. In the absence of such information, the worst-case delays or deadline violation probabilities must be determined assuming a worst-case combination of flows.

We assume that the network topology is known in advance, which includes the potential end-to-end path information and link bandwidth information. Each voice traffic will be regulated by a leaky bucket with burst size σ and average rate ρ at the entrance of the network. Link k is of capacity C_k. The link bandwidth utilization allocated to voice traffic is assumed to be u_k at link k. Since the deadline requirement can be either deterministic or statistical, resulting in deterministic services and statistical services, we classify the delay analysis as utilization-based deterministic delay analysis and utilization-based statistical delay analysis, respectively.

Utilization-based Deterministic Delay Analysis: If the deadline requirement is deterministic, we can bound the worst case queuing delay by the following theorem [65].

Theorem 5-1. The worst-case queuing delay d_k suffered by any voice packet with the highest priority at the buffer of output link k is bounded by[7]

$$d_k \le \frac{c_k - 1}{c_k - u_k} u_k \left(\frac{\sigma}{\rho} + Y_k\right) \qquad \text{(V-1)}$$

where $c_k = \sum_{j \in L_k} C_j / C_k$, $Y_k = \max_{R \in S_k} \sum_{s \in R} d_s$, L_k is the set of the input links of output link k with capacity C_k, and S_k is the set of all subroutes used by voice packets with the highest priority upstream from output link k.

The proof can be found in [65]. In **DisplayText cannot span more than one line!**, the value of Y_k, in turn, depends on the delays d_s's experienced at output link s other than k. Then, we have a circular dependency. Therefore, d_k can be determined iteratively.

[7] If $c_k \le 1$, then $d_k = 0$.

Furthermore, the end-to-end worst-case delay can be obtained, which only depends on the link utilization u_k, the parameters for voice traffic (burst size σ and average rate ρ), and the network topology.

Utilization-Based Statistical Delay Analysis: If the deadline requirement is probabilistic, we can bound delay violation probabilities as follows [66]:

Theorem 5-2. In this case, d_k is a random variable and D_k is denoted as its deadline. The violation probability of delay for any voice packet with the highest priority suffered at the buffer of output link k is bounded by

$$P\{d_k > D_k\} \le \begin{cases} \frac{1}{\sqrt{2\pi}} \exp(-24 \frac{1-u_k}{u_k^{\;2}} \frac{D_k}{\frac{\sigma}{\rho}}), & u_k \ge \frac{D_k}{\frac{\sigma}{\rho}} \\ \frac{1}{\sqrt{2\pi}} \exp(-6 \frac{1-u_k}{u_k^{\;3}} (u_k + \frac{D_k}{\frac{\sigma}{\rho}})^2), & u_k < \frac{D_k}{\frac{\sigma}{\rho}} \end{cases} \qquad \text{(V-2)}$$

The proof can be found in [66]. The end-to-end deadline violation probability can be bounded as

$$P\{d^{e2e} > \sum_{K \in R} D_K) \le 1 - \prod_{k \in R} (1 - P\{d_k > D_K\}) \qquad \text{(V-3)}$$

which only depends on the link utilization uk, the parameters for voice traffic (the burst size σ and the average rate ρ), and the network topology.

Our utilization-based delay analysis techniques show that, under the given network topology and traffic model, the queuing delay or deadline violation probability at each output queue depends on link bandwidth utilization. By limiting the utilization of link bandwidth, the overall delay or deadline violation probability can be bounded. Given the deadline requirement, with the utilization-based delay analysis techniques, the maximum link utilization computation can obtain the maximum link utilization, which will be applied in the LU-CAC mechanism to perform admission control.

b) Site Utilization Computation Submodule

The main task of this submodule is to compute the maximum site utilization for SU-CAC. As we mentioned, the SU-CAC mechanism tends to underutilize the network resource while providing end-to-end delay guarantees. The objective in the maximum site utilization computation is to optimize the overall site bandwidth utilization. Our maximum link utilization computation will be based on the maximum link utilization computation and further splitting each maximum link utilization to pairs of sites that share this link. Given the network topology and the limitation of link bandwidth allocated to voice traffic, we can optimize the overall bandwidth utilization to sites, defined as follows:

$$\text{Maximize } \sum_{R} u_R \tag{V-4}$$

$$\text{Subject to } \sum_{R \in k} u_R \le u_k \text{, for each link k} \tag{V-5}$$

$$u_R^0 \le u_R \le u_R^1 \text{, for each route R} \tag{V-6}$$

where u_k is the maximum bandwidth of link k allocated to voice traffic (obtained in the above section), $R \in k$ represents all routes among any pair of sites R going through link k, u_R is the bandwidth for R allocated to voice traffic, and u_R^0 and u_R^1 are the lower and upper bandwidth bounds for R allocated to voice traffic, respectively.

In the above equations, (V-4) is the overall bandwidth utilization, (V-5) shows that the bandwidth preallocation to each pair of sites is constrained by the link bandwidth limitation, and (V-6) is the user requirement for bandwidth preallocation to each pair of sites. This is a linear programming problem, which can be solved in polynomial time. The output, i.e., the preallocated bandwidth, will be used as bandwidth limitation in the SU-CAC mechanism.

c) Discussions

The utilization computation module has three input parameters: 1) The network topology, which can be obtained by existing network management tools. 2) The worst-case delay bounds or deadline violation probability bounds, which are predefined according to the timing requirement of the voice traffic. 3) The burst size and the average rate for voice flow model, which are predefined according to the characteristics of the voice traffic. Except for the network topology, all these parameters are fixed. When the network topology changes, our system will re-compute the link/site bandwidth utilization. Generally speaking, the network topology is relatively stable and predictable compared with the dynamic traffic. Even in the Internet, the study in [67] shows that the majority of the routing paths are stable. Therefore, the utilization re-computation will not happen frequently. We can further reduce the frequency of such re-computation by using the delay formula (7) in [68]. This formula depends only on the network diameter. A change in the network topology does not necessarily cause a change in the network diameter. With this formula, the utilization re-computation happens much less frequently.

Once the maximum link or site bandwidth utilizations are obtained and set in the LU-CAC mechanism and the SU-CAC mechanism, respectively, the Admission Decision Making Module will make the admission decision for the incoming call based on the overall bandwidth and the currently consumed bandwidth, which we discuss in the following.

2. Admission Decision Making Module

The Admission Decision Making module supports both the LU-CAC mechanism and the SU-CAC mechanism8. In this section, we will describe the data structure and the working process of the two mechanisms in the Admission Decision Making module.

a) Admission Decision Making in LU-CAC Mechanisms

To support the LU-CAC mechanism, the Admission Decision Making module has to keep the network topology information and the routing information. There are two tables for supporting this mechanism: the Bandwidth Table and Routing Table. The Bandwidth Table is used to keep the information about how much of the configured bandwidth on the links is currently consumed by voice traffic and how much link bandwidth is available for calls as shown in Table 2. Note that overall bandwidth in Table 2 Table 2Table 2Table 2Table 2Table 2Table 2Table 2is the maximum link utilization from the

[8] The Admission Decision Making module for the SU-CAC mechanism is an optional implementation in CACA since the SU-CAC mechanism in the current VoIP system can simply provide QoS guarantees by applying the result from the Utilization Computation Module.

Utilization Computation module. The routing information can be found in the Routing Table.

Table 2. The Bandwidth Table in LU-CAC

link	overall bandwidth	available bandwidth
$link_1$	40.0 Mbps	21.0 Mbps
$\vdots$	$\vdots$	$\vdots$
$link_L$	35.0 Mbps	15.0 Mbps

Once a call request comes, each link along the call route will be checked to see if there is sufficient bandwidth left in the Bandwidth Table. First of all, the call route with the source and destination of the call should be found in the Routing Table as shown in Table 3. If all links along the call route have sufficient bandwidth left, then the CAC module will admit the call and decrease the available bandwidth of all call links by the requested bandwidth; otherwise, it will reject it. Once the call tears down, the bandwidth requested by the call will be returned to the pool for each link along the call route.

Table 3. The Routing Table in LU-CAS

source	destination	links
src_1	dst_1	$src_1 \rightarrow \cdots \rightarrow node_1^i \rightarrow \cdots \rightarrow dst_1$
$\vdots$	$\vdots$	$\vdots$
src_R	dst_R	$src_R \rightarrow \cdots \rightarrow node_R^i \rightarrow \cdots \rightarrow dst_R$

b) Admission Decision Making in SU-CAC Mechanisms

To support the SU-CAC mechanism, the Admission Decision Making module keeps neither the information about the overall bandwidth nor the available bandwidth for each individual link of the network. It takes a fixed amount of bandwidth for each pair of sites or a fixed total amount of bandwidth from/to a site, which is statically configured in the Bandwidth Table. Note that the fixed bandwidth is allocated by the Utilization Computation Module in our QoS-provisioning system.

Table 4 shows an example of a bandwidth table for pairs of sites. As each call is setup, the sites of source and destination can be known. If there is sufficient bandwidth left for the pair of sites, then the call is admitted and a certain amount of bandwidth is subtracted and will be returned to the pool when the call tears down. Otherwise, the call request is rejected.

Table 4. The Bandwidth Table in SU-CAC

pair of sites	overall bandwidth	available bandwidth
$PairSites_1$	3.0 Mbps	1.6 Mbps
$\vdots$	$\vdots$	$\vdots$
$PairSites_R$	2.0 Mbps	0.8 Mbps

E. Integration with Existing VoIP Systems

1. Methodology in Integration

The Call Admission Control Agent (CACA) is integrated into the existing VoIP systems through the Integration Component (IC). The Integration Component monitors and intercepts call setup signaling, withdraws useful messages (i.e., the bandwidth required by calls, the locations or addresses of callers and callees), passes them to the admission decision making module, and executes the call admission decision. Generally speaking, there are two approaches for this kind of component to intercept the call setup signaling and to execute the call admission decision:

Front-End approach: In this approach, the call setup requests must pass through the agent before reaching the existing call admission decision unit (e.g., CallManager). The call setup responses must also pass through the agent before coming back to the call request endpoint. The agent can directly enforce its call admission decision to the call setup request by adding, modifying or dropping signaling between the endpoint and the existing admission decision unit. There are two basic methods to implement the agent in this approach: proxy method and filter method. To implement proxy method, the agent will be implemented as a signaling proxy. The consistency of the call signalings between the end point and the existing call admission unit can be achieved. However, in most cases, it is complicated to implement a functional proxy, and the integration overhead cannot be neglected since the integration is not transparent to the existing system. To implement the filter method, the agent only intercepts its relevant signaling and is easy to implement. Since the (filter) agent is treated as an IP router or firewall by the existing system, the integration is transparent to the existing system. It may be difficult to achieve the call signaling consistency since there is no direct interaction between the (filter) agent and the existing system.

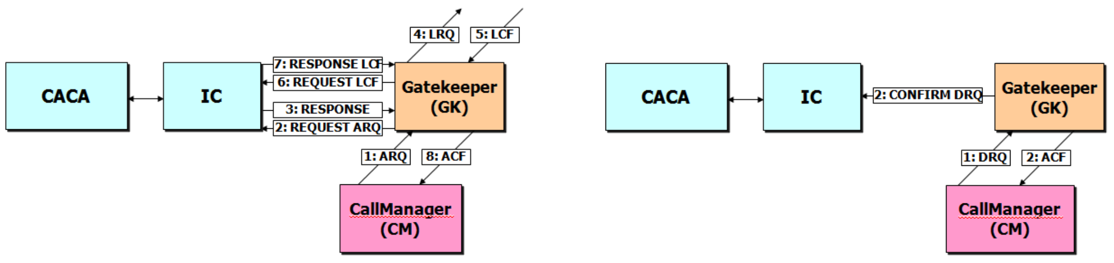
Back-End approach: In this approach, the call setup requests and responses will be forwarded to the agent by the existing call admission decision unit (e.g., Gatekeeper). The agent will indirectly execute its call admission decision to the call setup request by negotiating with the existing call admission decision unit. The Back-End overcomes several problems of the Frond-End approach: 1) The implementation of the agent will not be very complicated since the existing system normally allows the agent to selectively receive and process the signaling; 2) the consistency of the signaling can be easily achieved since the agent directly interacts with call admission decision unit; 3) the integration overhead is little because only the existing admission control unit is aware of the agent. However, the Back-End approach requires the existing system to have the ability that the call setup requests and responses can be redirected to the external application, e.g., our CACA, while the Front-End one does not. In most cases, the Back-End approach has more advantages than the Front-End approach.

2. Case Study

In this section, we would like to use the Cisco VoIP system to illustrate how our QoS provisioning subsystem can be integrated with the existing VoIP system. VoIP is a key part of Cisco's AVVID (Architecture for Voice, Video, and Integrated Data) framework for multiservice networking [69]. The architecture of the Cisco VoIP system is the same as the one shown in Figure 24. It has two major components: Gatekeeper and CallManager. Due to the different design and implementation methodologies of these two components, we adopt the Back-End approach for Cisco Gatekeeper and the Front-End approach with the filter method for Cisco CallManager in the IC, respectively. In the remainder of the section, we will describe the details of how the integration component works with the Cisco Gatekeeper and Cisco CallManager.

a) Integration with Gatekeeper

Cisco Gatekeeper is a built-in feature of Cisco IOS in some Cisco Router series (e.g., 2600, 3600 series) and is a lightweight H.323 gatekeeper. The Registration, Admission, and Status (RAS) signaling that the Cisco Gatekeeper handles is H.323-compatible. Cisco Gatekeeper provides interface for external application servers to offload and supplement its features. The interaction between the Cisco Gatekeeper and the external application is completely transparent to the H.323 endpoint.



(a) Call Setup (b) Call Tearing-Down

Figure 28. An Illustration of a Successful Call Procedure in IC for Gatekeeper

As shown in Figure 28, the Back-End approach is adopted for the IC to intercept the call signaling between the CACA and the Gatekeeper. The IC handles the H.323 RAS signaling and communicates with the Cisco IOS Gatekeeper. The communication between the Cisco IOS Gatekeeper and the IC is based on Cisco's propriety protocol, Gatekeeper Transaction Message Protocol (GKTMP) [70]. GKTMP provides a set of ASCII RAS request/response messages between Cisco Gatekeeper and the external application over a TCP connection. There are two types of GKTMP messages: 1) GKTMP RAS Messages: It is used to exchange the contents of RAS messages between the Cisco IOS Gatekeeper and the external application. 2) Trigger Registration Messages: It is used by the external application to indicate to the Cisco Gatekeeper which RAS message should be forwarded. If an external application is interested in receiving certain RAS messages, it must register this interest with the Cisco Gatekeeper.

In our implementation, the IC is interested in receiving the following four RAS messages from the Gatekeeper: Admission Request (ARQ), Location Confirm (LCF), Location Reject (LRJ), and Disengage Request (DRQ). LRJ and DRQ is notification-only message to IC. All of the four messages will be automatically registered to Cisco Gatekeeper once the CACA is functional.

Figure 28 (a) illustrates a successful call setup procedure. As an example, we illustrate the detail of a successful call setup procedure as blow:

1) A CallManager as a H.323 gateway sends an ARQ message to Gatekeeper.
2) The Gatekeeper searches its trigger condition and finds a match to the IC. It patches the message ARQ to message named "REQUEST ARQ" and sends it to the IC.
3) The IC processes the "REQUEST ARQ", e.g. recording down the bandwidth of the call, conference ID, address of the caller etc. and sends back a "RESPONSE ARQ" to let Gatekeeper continue normal processing.
4) The Gatekeeper sends a Location Request (LRQ) message to request address translation.
5) The Gatekeeper receives a message LCF, patches it to message "REQUEST LCF" and sends it to the IC.
6) The IC withdraws the address of the caller from the message "REQUEST LCF". In addition to the previous buffered information from message ARQ, it sends collected information to the Admission Decision Making Module of the CACA.
7) The Admission Decision Making Module of the CACA sends a confirmation to IC to accept the call if the network resource is available. The IC sends a message "RESPONSE LCF" back to the Gatekeeper. In the meantime, the Admission Decision Making Module updates the status of the network resource and the call.

8) The Gatekeeper sends message Admission Confirm (ACF) to grant call permission to the CallManager.

Figure 28 (b) illustrates a simple tearing down procedure, where the IC will update the status of network resource once receiving the message DRQ.

b) Integration with CallManager

Cisco CallManager is a comprehensive and heavyweight VoIP processing application, which runs on the Microsoft Windows platform. It can interact with endpoints using multiple protocols, e.g., Skinny Client Control Protocol (SCCP), H.323, and Session Initiation Protocol (SIP), etc. In this work, we implement SCCP, a popular signaling protocol in Cisco VoIP system, in the IC for CallManager.

To the best of our knowledge, Cisco CallManager does not provide an interface for external applications to supplement its call admission control mechanism as a Gatekeeper does. In this case, only the Front-End approach can be adopted to intercept the Call Signaling of a CallManager. As we mentioned above, there are two basic methods, proxy and filter, to implement Front-End approach. By the proxy method, if the integration of the CACA to the current VoIP system is not transparent to the endpoints, the integration will affect thousands of the endpoints. The overhead and interruption caused by the integration to an operational environment is a realistic problem when using the proxy method. Considering that, we use the filter method in the current design and implementation of the IC for CallManager. Figure 29 shows the basic idea of this method.

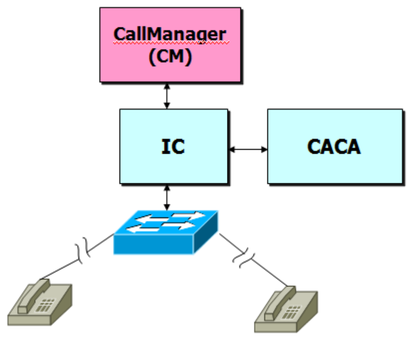


Figure 29. An Illustration of Communication Protocol in IC for CallManager

Since the basic idea and the procedure of the signaling process in both the IC for CallManager and the IC for Gatekeeper are similar, we would like to highlight the difference in intercepting the SCCP using the IC for CallManager. The CallManager is unware of the IC. It directly sends the implicit grant permission message (i.e.,message "StartMediaTransmission") to the endpoints of the admitted call. However, in case the CACA makes a decision to deny the call because of the lack of available bandwidth, it should not let the message "StartMediaTransmission" be received by the endpoints. One of the approaches is to continue dropping the message from the CallManager until the CallManager terminates the TCP connection after a finite timeout. There are two problems: 1) The caller does not get any indication whether the call is accepted or not. 2) The timeout is about 60 seconds. To compensate for the above two problems, the IC can explicitly indicate caller by sending the busy tone message, "StartToneMessage," to the endpoint and prevent the CallManager from sending a message to the endpoint by sending the call terminating message, "OnHookMessage," to the CallManager. However, additional messages from the IC would interfere with the synchronization of the TCP connection between the endpoint and the CallManager. To speed-up the resynchronization of the TCP connection, and limit the impact on the CallManager, the IC will send a TCP RESET packet to the endpoint in place of the CallManager.

F. Performance Evaluation

We consider two metrics in the performance evaluation: 1) the introduced overhead to the admission and 2) the overall bandwidth utilization. Correspondingly, we choose two measurement metrics: admission latency and admission probability. Admission latency is used to measure the overhead of admission. Admission probability is the ratio of the number of admissions to the number of overall requests, which is a well-known metric to measure the overall bandwidth utilization. The higher the admission probability, the higher the overall bandwidth utilization achieved.

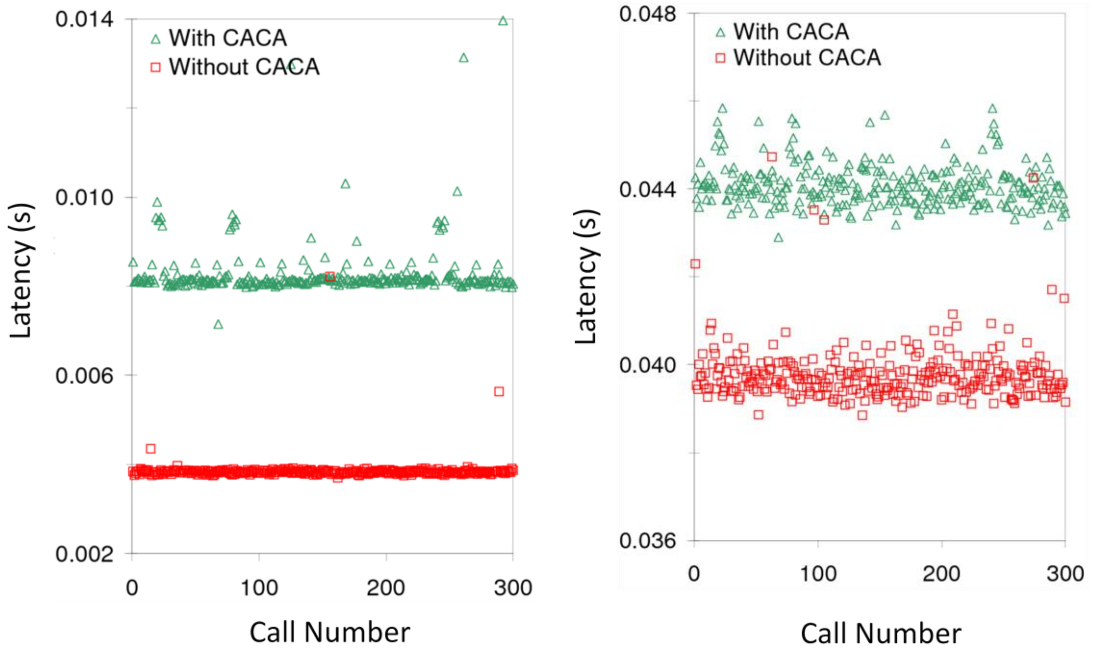
1. Admission Latency

In this section, we run a suite of experiments to evaluate the admission latency in two VoIP systems: 1) the one with our CACA and 2) the one without our designed CACA. Due to the different design and implementation methodology of CACA for CallManager and Gatekeeper, we run two experiments for both cases. The experiments are run in the Internet2 Voice Over IP Testbed at Texas A&M University.

a) Call Admission Control Agent (CACA) for Cisco Gatekeeper

In the experiment, we tried 300 calls for each CAC mechanism. The call signaling crosses two Cisco CallMangers and two Cisco Gatekeepers from a Cisco IP phone in

Texas A&M University to another IP phone in Indiana University. To show the introduced overhead by our designed QoS-provisioning system, we have two sets of data: local admission latency and round-trip admission latency.



(a) Local Admission Latency (b) Round-trip Admission Latency

Figure 30. The Distribution of Local and Round-trip Admission Latency for Gatekeeper

Figure 30 (a) shows the distribution of local admission latency between receiving ARQ and sending out LRQ by Gatekeeper. Figure 30 (b) shows the distribution of round-trip admission latency between receiving ARQ and sending ACF out by Gatekeeper. Table 5 gives us the summary of the distribution of admission latency for each case in terms of the mean value and standard deviation. The local admission latency excludes the network latency and the processing latency on the other side. It shows a more accurate latency introduced by our designed QoS-provisioning system, which is shown by the standard deviation of the latency distribution in Table 5. The round-trip admission latency gives us the view of the overall admission latency.

Table 5. The Mean Value and Standard Deviation of Latency Distribution for Gatekeeper

	local admission latency (ms)		round-trip admission latency (ms)	
	mean value	standard deviation	mean value	standard deviation
with CACA	8.286	0.863	44.302	2.665
without CACA	3.850	0.277	39.870	1.530

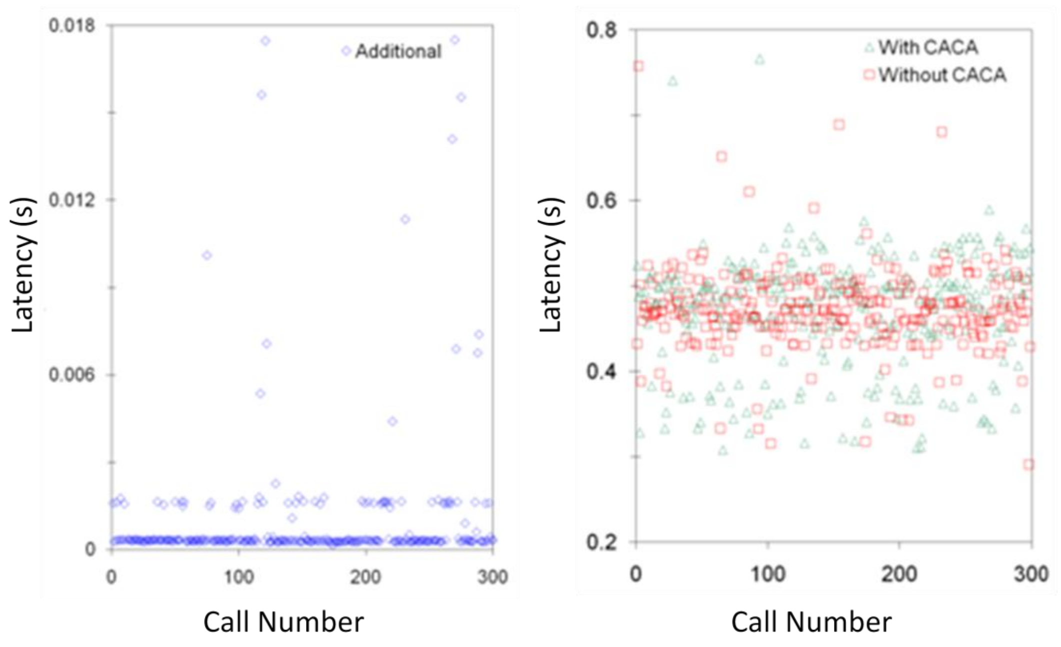
The admission latency in the VoIP system with CACA is around 44.3 ms. The admission latency in the VoIP system without CACA is around 39.8 ms. With CACA, the introduced latency is about 8.3 – 3.9 = 4.4 ms. The overall latency is very acceptable and the introduced latency is pretty small. To measure the introduced latency, we measured the admission latency between receiving ARQ and sending out LRQ from the Gatekeeper, not the additional latency from the CACA directly. Here, the additional admission latency includes not only the admission latency introduced in CACA, but also the additional latency in Gatekeeper caused by interaction between Gatekeeper and CACA, which cannot be measured directly.

b) Call Admission Control Agent (CACA) for Cisco CallManager

In the experiment, we also initiated 300 calls for each CAC mechanism. The call signaling crosses one Cisco CallManger between two Cisco IP phones at Texas A&M

University. To show the introduced overhead by our designed QoS provisioning system, we have two sets of data: local admission latency and round-trip admission latency.

Figure 31 (a) shows the distribution of local additional admission latency which is introduced by CACA in processing one call signaling message. Figure 31 (b) shows the distribution of round-trip admission latency.



(a) Local Admission Latency (b) Round-trip Admission Latency

Figure 31. The Distribution of Local and Round-trip Admission Latency for CallManager

Table 6 gives us the summary of the distribution of admission latency for each case in terms of the mean value and standard deviation.

Table 6. The Mean Value and Standard Deviation of Latency Distribution for CallManager

	latency (ms)	
	mean value	standard deviation
with CACA	476.002	94.796
without CACA	479.367	92.114
additional	1.202	3.080

The admission latency in the VoIP system with CACA is around 476 ms. The admission latency in the VoIP system without CACA is around 479.3 ms. With CACA, the introduced latency is about 1.2 ms (i.e., additional latency). The overall latency is acceptable and the introduced latency is quite small[9].

2. Admission Probability

To make the data convincing, the measure of admission probability requires a high volume of calls in the VoIP system. However, it is not feasible or realistic to produce a high volume of calls in the VoIP system: First, our designed QoS-provisioning system is

[9] As shown in Table 6: 1) The average latency with CACA is less than the one without CACA. It is because the latency varies on the status of the network and VoIP system, and the additional latency introduced by CACA is a small proportion of the round-trip latency. 2) The average latency with CACA for CallManager is much larger than the one for Gatekeeper. It is because the SCCP signaling is more comprehensive than RAS signaling, and the call admission requires tens of round-trip SCCP messages between the CallManager and endpoint for each call request.

only deployed in Internet2 Voice Over IP Testbed in Texas A&M University, where simultaneous calls from many sites are not available. Second, even in the fully-deployed VoIP system, a high volume of calls for the experiment will affect the operation of VoIP heavily. Admission probability can only be measured by simulation. In this section, we run a suite of simulation to evaluate the admission probability for the LU-CAC mechanism and the SU-CAC mechanism, respectively.

Traditionally, call arrivals follow a Poisson distribution and call lifetimes are exponentially distributed. This call mode can approximate the realistic call mode very well. In our simulation, we use this call mode to simulate calls by Mesquite CSIM 19 toolkits [71] for simulation and modeling. In the simulation, overall requests for call establishment in the network form a Poisson process with rate λ, while call lifetimes are exponentially distributed with an average lifetime of $\mu = 180$ seconds for each call. All calls are duplex (bidirectional) and use G.711 codec, which has a fixed packet length of (160 + 40) bytes (RTP, UDP, IP headers, and two voice frames) and a call flow rate of 80 Kbps (including 64 Kbps payload and other header).

Two different network topologies are chosen for the simulation: Internet2 backbone network and a campus network. Gatekeeper and CallManager are configured to perform call admission control in the Internet2 environment and in the campus environment, respectively.

a) Internet2 Backbone Network

Abilene is an advanced backbone network that supports the development and deployment of the new applications being developed within the Internet2 community. Figure 32 shows the core map of the Abilene network (September 2003) used in our simulation. There are 11 core node routers, each located in a different geographical area. All backbone links are either OC48 (2.4 Gbps) or OC192 (9.6 Gbps). The call route will be chosen uniformly randomly from the set of all pairs of core node routers. Suppose that the end-to-end deadline for queueing is 10 ms. The maximum utilization is 0.195 for deterministic service, i.e., under the condition that about 19.5 percent link bandwidth is used for voice traffic, the end-to-end delay for any voice packet can meet the deadline requirement. The maximum utilization is 0.307 for statistical service with deadline violation probability 10^{-6}, i.e., under the condition that about 30.7 percent link bandwidth is used for voice traffic, any voice packets may miss the deadline with small probability 10^{-6} at most. λ changes from 100.0 to 1000.0.

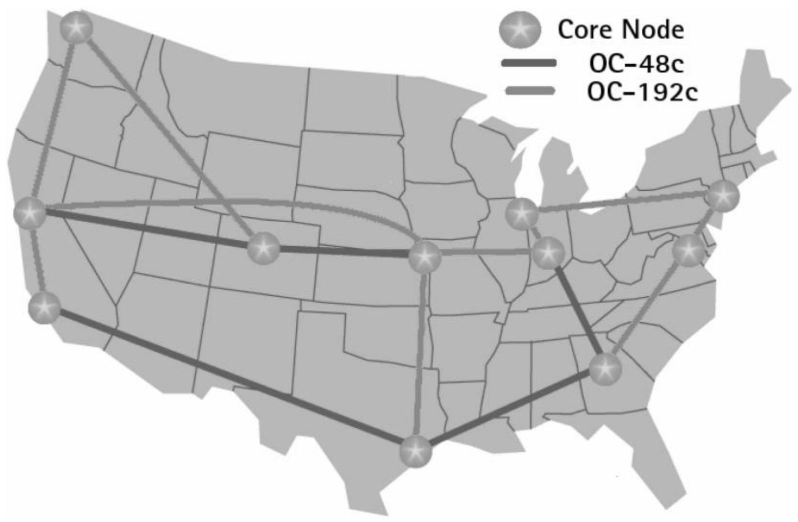


Figure 32. The Core Map of Abilene Network (September 2003)

Figure 33 shows the admission probabilities for the voice call in the two CAC mechanism as a function of arrival rates. We find that the LU-CAC mechanism can achieve a much higher admission probability than SU-CAC for both deterministic service and statistical service. Statistical service can achieve a much higher admission probability than deterministic service, as expected.

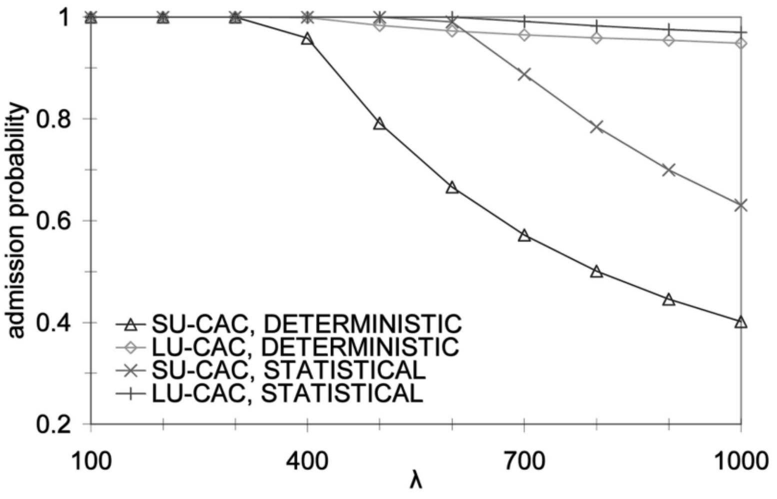


Figure 33. The Admission Probability in Abilene Network

b) Campus Network

Figure 34 shows the campus network topology used in our simulation. The link bandwidth is either 100 Mbps or 155 Mbps. The call route will be chosen uniformly randomly from the set of all pairs of sites (0, 1, . . . , 18). Suppose that the end-to-end deadline is 10 ms for queueing. The maximum utilization is 0.208 for deterministic service, i.e., under the condition that about 20.8 percent link bandwidth is used for voice traffic, the end-to-end delay for any voice packet can meet the deadline requirement. The maximum utilization is 0.332 for statistical service with deadline violation probability 10^{-6}, i.e., under the condition that about 33.2 percent link bandwidth is used for voice traffic, any voice packets may miss the deadline with small probability 10^{-6} at most. λ changes from 1.0 to 10.0.

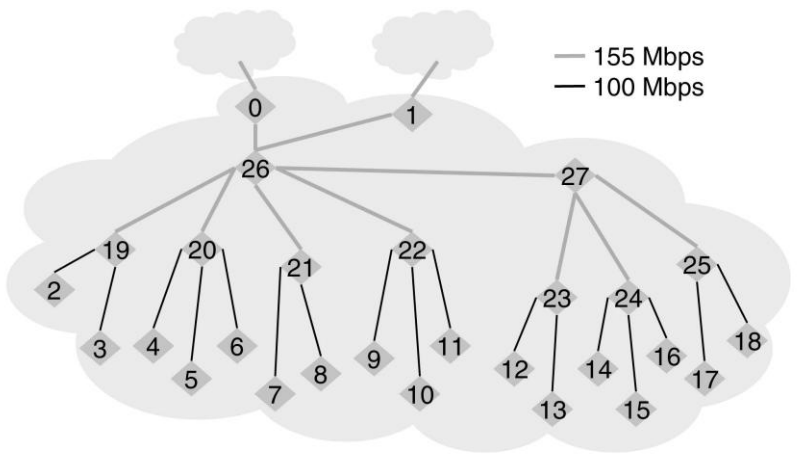


Figure 34. A Campus Network Topology

Figure 35 shows the admission probabilities for the voice call in the two CAC mechanisms as a function of arrival rates. The admission probabilities in the two call admission control mechanism are different. Similar to the observation made in the Internet2 Backbone network, the LU-CAC mechanism can achieve much higher admission probability than SU-CAC for both deterministic service and statistical service. Statistical service can achieve a much higher admission probability than deterministic service, as expected.

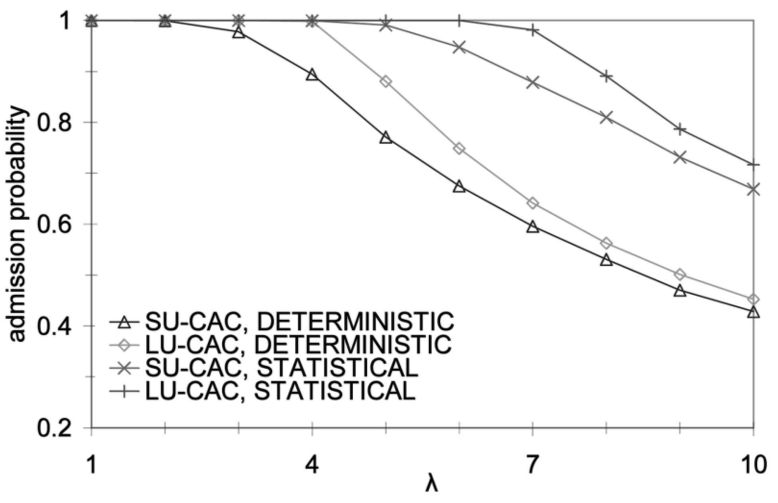


Figure 35. The Admission Probability in the Campus Network

G. Deployment

We have successfully realized our system in the Internet2 Voice Over IP Testbed at Texas A&M University. We systematically evaluate our proposed QoS-provisioning system in terms of admission delay and admission probability. Our data show that, if a VoIP system is designed with our new system model and design methodology, the overall system can achieve high resource utilization while invoking relatively low overhead. A portion of the research results have been published in [68], [72], [73].

CHAPTER VI

SUMMARY

This research work provides analysis and addresses issues related to the design and realization of the next generation of mission-critical application systems. We pay special attention to two important aspects of these systems: reliability and real-time guarantees. We narrow our focus to three most important software design scenarios, single node mission-critical applications, cluster environment mission-critical applications, and wide area network environment mission-critical applications. For these three types of applications, we proposed novel system models, design methodologies to improve system reliability and provide real-time guarantees. Based on these novel model and design methodology, we build demonstrative or real world applications and through detailed analysis and performance evaluation, we show that our model and design methodology can indeed improve the reliability and provide real-time guarantees for these different mission-critical applications.

REFERENCES

[1] "Definition of Mission Critical," http://en.wikipedia.org/wiki/Mission-critical, 2010.

[2] National Hurricane Center, "November 2005 Atlantic Tropical Weather Summary," http://www.nhc.noaa.gov/archive/2005/tws/MIATWSAT_nov_final.shtml, 2005.

[3] E. V. Larson, and J. E. Peters, "Preparing the U.S. Army for Homeland Security: Concepts, Issues, and Options," RAND Corporation, 2001.

[4] E. N. Rappaport, J. L. Franklin, L. A. Avila, S. R. Baig, J. L. Beven II, and et al., "Advances and Challenges at the National Hurricane Center," http://www.nhc.noaa.gov/pdf/NHC_WAF_Advances_Challenges_200904.pdf, April 2009.

[5] Carolina Emergency Management Division (SCEMD), "2009 South Carolina Hurricane Plan," http://www.scemd.org/Plans, 2009.

[6] C. Szyperski, D. Gruntz, and S. Murer. *Component Software: Beyond Object-Oriented Programming*, 2nd Edition, Addison-Wesley Professional, 2002.

[7] A. Pasetti and W. Pree. "The Component Software Challenge for Real-time Systems," *Proc. of the First International Workshop on Real-Time Mission Critical Systems*, Scottsdale, AZ, Nov./Dec. 1999.

[8] Object Management Group, "Realtime CORBA joint revised submission," OMG Document orbos/99-02-12 ed., March 1999.

[9] D. C. Schmidt and F. Kuhns. "An Overview of the Real-time CORBA Specification," *IEEE Computer*, vol. 33, no. 6, pp.56–63, 2000.

[10] D. C. Schmidt, "Real-time CORBA with TAO," http://www.cs.wustl.edu/~schmidt/TAO.html, 2010.

[11] J. A. Stankovic, R. Zhu, R. Poornalingam, C. Lu, Z. Yu, M. Humphrey, and B. Ellis. "VEST: An Aspect-based Composition Tool for Real-time Systems," *Proc. the IEEE Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canada, May 2003.

[12] "SURA Coastal Ocean Observing and Prediction (SCOOP) Program," www.csc.noaa.gov/cots/accomp_reports/SCOOP.pdf, 2005.

[13] "Cloud Computing," http://en.wikipedia.org/wiki/Cloud_computing, 2010.

[14] "One-Way Transmission Time (Recommendation G. 114)," *International Telecommunication Union (ITU)*, http://www.itu.int/itudoc/itu-t/aap/sg12aap/history/g.114/g114.html, 1996.

[15] Object Management Group, "CORBA specifications," http://www.omg.org, 1995.

[16] Microsoft, "COM: Delivering on the Promises of Component Technology," http://www.microsoft.com/com, 2004.

[17] Sun Microsystems, "Enterprise JavaBeans Technology," http://java.sun.com/products/ejb, 2005.

[18] National Oceanic & Atmospheric Administration, http://www.noaa.org, 2006.

[19] M. Kaminski, R. Weaver, M. Marquis, and W. Meier, "Cryospheric Products from Earth Observing System Satellites at the National Snow and Ice Data Center," *IEEE Int. Proc. Geoscience and Remote Sensing Symposium (IGARSS)*, Anchorage, AK, September 2004.

[20] L. A. Hunt, "MOPITT Data and Tools Available from the Atmospheric Sciences Data Center," *IEEE Int. Proc. Geoscience and Remote Sensing Symposium (IGARSS)*, Anchorage, AK, September 2004.

[21] I. A. Newman, "Distributed Data and Metadata Management in the GENIE Project," *IEE Colloquium on Distributed Databases*, London, UK, December 1992.

[22] J. Thomson, D. Adams, P. J. Cowley, and K. Walker, "Metadata's Role in a Scientific Archive," *IEEE Computer*, vol. 36, no. 12, pp. 27-34, 2003.

[23] S. Testa, W. Chou, "The Distributed Data Center: Front-end Solutions," *IEEE Journal on IT Professional*, vol. 6, no. 3, pp. 26-32, 2004.

[24] S. Graupner, V. Kotov, and H. Trinks, "Resource-sharing and Service Deployment in Virtual Data Centers," *IEEE Int. Conference Distributed Computing Systems Workshops (ICDCSW)*, Los Alamitos, CA, July 2002.

[25] D. A. Reed, "Grids, the TeraGrid and Beyond," *IEEE Journal on Computer*, vol. 36, no. 1, pp. 62-68, January 2003.

[26] J. Altmann, "A Model for Resource Sharing for Internet Data Center Providers within the Grid," *Proc. of the First IEEE International Workshop on Grid Economics and Business Models (GECON)*, Seoul, South-Korea, April 2004.

[27] "Sun Grid," http://www.sun.com/service/sungrid/, 2005.

[28] R. Buyya and S. Vazhkudai, "Compute Power Market: Towards a Market-oriented Grid," *Proc. of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, May 2001.

[29] C. Kenyon and G. Cheliotis, "Grid Resource Commercialization: Economic Engineering and Delivery Scenarios," *Grid Resource Management: State of the Art and Research Issues*, Chapter 26, pp. 465-478, Kluwer Academic Publisher, 2004.

[30] B. Plale, D. Gannon, D. Reed, S. Graves, K. Droegemeier, B. Wilhelmson, and M. Ramamurthy, "Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD," *ICCS Workshop on Dynamic Data Driven Applications*, Atlanta, GA, 2005.

[31] "The Geosciences Network (GEON)," [online] http://www.geongrid.org/, 2005.

[32] V. Sanjeepan, A. Matsunaga, L. Zhu, H. Lam, and J. Fortes, "A Service-Oriented, Scalable Appraoch to Grid-Enabling of Legacy Scientific Applications," *Proc. of the IEEE International Conference on Web Services (ICWS)*, Orlando, Florida, July 2005, pp. 553-560

[33] "Nortel Networks: Voice over IP Solutions—Succession Enterprise," http://www.nortelnetworks.com, 2004.

[34] "Avaya IP Telephony Solutions," http://www.avaya.com, 2004.

[35] "Alcatel IP Telephony," http://www.alcatel.com, 2004.

[36] J. Davidson, *Deploying Cisco Voice over IP Solutions*, Cisco Press, 2002.

[37] "3Com IP Telephony Solution," http://www.3com.com, 2004.

[38] S. Jamin, S. Shenker, and P. Danzig, "Comparison of Measurement-Based Admission Control Algorithms for Controller-Load Service," *Proc. IEEE Infocom*, Kobe, Japan, Apr. 1997.

[39] D. Tse and M. Grossglauser, "Measurement-Based Call Admission Control: Analysis and Simulation," *Proc. IEEE Infocom*, Kobe, Japan, Apr. 1997.

[40] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Providing Absolute Differentiated Services for Real-Time Applications in Static-Priority Scheduling Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 326-339, 2004.

[41] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing Synchronous Message Deadlines with the Timed Token Protocol," *Proc. IEEE Int'l Conf. Distributed Computing Systems*, Yokohama, Japan, June 1992.

[42] S. Chong and S. Li, "(s, ß)-Characterization Based Connection Control for Guaranteed Services in High Speed Networks," *Proc. IEEE Infocom*, Boston, MA, Apr. 1995.

[43] J. Liu. *Real-Time Systems*, Prentice Hall, 2000.

[44] F. P. Kelly, "Blocking Probabilities in Large Circuit-switched Networks," *Advance in Applied Probability*, vol. 18, pp.473–505, 1986.

[45] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan, "ATM Network Design and Optimization: A Multirate Loss Network Framework," *IEEE/ACM Trans. Networking*, vol. 4, no. 4, pp.531–543, 1996.

[46] "JBoss EJB Application Server," http://www.jboss.org, 2004.

[47] Sun Microsystems, "Java Management Extensions," http://java.sun.com/products-/JavaManagement, 2003.

[48] M. Fleury and F. Reverbel. "The JBoss Extensible Server," *Proc. of ACM/IFIP/USENIX International Middleware Conference (Middleware 2003)*, Rio de Janeiro, Brazil, June 2003.

[49] S. Stark and The JBoss Group, "JBoss Administration and Development Documentation"– ebook – 3.2.1. http://www.jboss.org/docs, 2003.

[50] TimeSys. "Real-time Specification for Java Reference Implementation," http://www.timesys.com, 2001.

[51] Sun Microsystems. "Java Remote Method Invocation specification," http://java.sun.com/j2se/1.3/docs/guide/rmi, 2003.

[52] J. Wu, "General Schedulability Bound Analysis and Its Applications in Real-Time Systems," Dissertation, Texas A&M University, May 2006.

[53] "Portable Batch Scheduling System (PBS)," http://www.pbsworks.com/, 2004.

[54] "Local Data Manager (LDM)," http://www.unidata.ucar.edu/software/ldm/, 2005.

[55] "Apache Struts," http://struts.apache.org/, 2004.

[56] Sun Microsystems, "Java Authentication and Authorization Service (JAAS) Reference Guide," http://java.sun.com/developer/technicalArticles/Security/ jaasv2/, 2003.

[57] N.Goodarzi, "Develop JAAS Security on JBoss Application Server," http://www.javadev.org/files/JAAS-JBoss.pdf, 2008.

[58] "Security on JBoss", http://docs.jboss.com/jbossas/admindevel326/html, 2004.

[59] D. Heimbigner and D. McLeod, "A Federated Architecture for Information Management," *ACM Transactions on Office Information Systems (TOIS)*, vol. 3, no. 3, pp. 253-278, 1985.

[60] "Create, Read, Update and Delete (CRUD)," http://en.wikipedia.org/wiki/Create,_read,_update_and_delete, 2010.

[61] "Extract, Transform, Load (ETL)," http://en.wikipedia.org/wiki/Extract,_transform,_load, 2010.

[62] D. Cheng, Z. Mai, J. Wu, and W. Zhao, "A Service-Oriented Integrating Practice for Data Modeling, Analysis and Visualization," *Proc. of the Second IEEE International Symposium on Service-Oriented Applications, Integration and Collaboration (SOAIC)*, Shanghai, China, October 2006.

[63] "Registration, Admission and Status Signaling (Recommendation H. 225/RAS)," *International Telecommunication Union (ITU)*, 1998.

[64] D. Collins, *Carrier Grade Voice over IP*, McGraw-Hill Professional Publishing, 2001.

[65] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Providing Absolute Differentiated Services for Real-Time Applications in Static-Priority Scheduling Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 326-339, 2004.

[66] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Differentiated Services with Statistical Real-Time Guarantees in Static-Priority Scheduling Networks," *Proc. IEEE Real-Time Systems Symp.*, London, UK, Dec. 2001.

[67] V. Paxson, "End-to-End Routing Behavior in the Internet," *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1996.

[68] S. Wang, Z. Mai, D. Xuan, and W. Zhao, "Design and Implementation of QoS-Provisioning System for Voice over IP," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 17, no. 3, pp. 276-288, 2006.

[69] J. Alexander, C. Pearce, A. Smith, and D. Whetten, *Cisco CallManager Fundamentals*, Cisco Press, 2002.

[70] Cisco, "Gatekeeper External Interface Reference, Version 3.1," http://www.cisco.com/en/US/docs/ios/12_2/gktmp/gktmpv3_1/h323intr.html, 2001.

[71] Mesquite Software, "Mesquite csim 19," http://www.mesquite.com, 2004.

[72] Z.Mai, S.Wang, D.Xuan, and W. Zhao, “Delay Performance and Management of VoIP System,” *VoIP Handbook: Applications, Technologies, Reliability, and Security*, CRC Press, Chapter 10, pp. 169-186, 2008.

[73] S. Wang, Z. Mai, W. Magnussen, D. Xuan, and W. Zhao, “Implementation of QoS-Provisioning System for Voice over IP,” *Proc. of Real-Time and Embedded Technology and Applications Symposium*, San Jose, CA, September, 2002.

VITA

Zhibin Mai was born in Guandong, the People's Republic of China. He received his B.S. and M.S. degress in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 1992 and 1995 respectively. His research interests include design and implementation of real-time application systems, high performance computing and data center. He can be reached care of Dr. Wei Zhao, Department of Computer Science, Texas A&M University, TAMU 3112 College Station, TX, 77843-3112. His email address is: zbmai@yahoo.com.