Middleware for Distributed Industrial Real-Time Systems on ATM Networks

Ichiro Mizunuma†, Chia Shen‡, and Morikazu Takegaki†

† Industrial Electronics and Systems Lab.,
Mitsubishi Electric Corp.
8-1-1, Tsukaguchi-honmachi, Amagasaki, Hyogo, 661, Japan.
email: mizunuma,takegaki@con.sdl.melco.co.jp

‡ A Mitsubishi Electric Reseach Lab.
201 Broadway, Cambridge, Massachusetts 02139 USA
email: shen@merl.com

Abstract *In this paper we address the problem of middleware design for constructing ATM LAN based distributed industrial plant monitoring and control systems. In particular, we present* a real-time client-server programming model *based on a uniform ATM network. This model is being realized in our middleware called* MidART. *The middleware provides a set of industrial application specific but network transparent programming abstractions and application programming interface (API) that support individual application QoS requirements. In order to achieve* on-demand *transmission of plant data, we have developed a concept called* selective real-time channels *to be supported by* MidART. *We present the design and protocols of* selective real-time channels *and describe how QoS requirements of applications are guaranteed.*

1 Introduction

We address the problem of middleware design for constructing ATM LAN based industrial plant monitoring and control systems. These systems are real-time in nature, requiring the guarantee of hard timing constraints in response to both human operations and sensor notification of plant states. Traditionally, such distributed plant control systems have been built mostly with proprietary network components and host systems, and real-time constraints are met with physical isolation of sub-networks. In order to guarantee the timing constraints of plant control, the workstations supporting the human interface for plant information management often do not reside on the same physical network as the network in the controlled plant sites. Similarly, another separate physical network is used for video transmission from industrial TV cameras in the plant monitoring sites. Consequently, these current distributed plant control systems are inflexible, expensive, difficult to modify and not scalable.

Figure 1 is a typical example of a current industrial control system, similar to those systems from Mitsubishi Electric Corp. and those discussed in [Meinert 95]. The network architecture contains two layers, a *control layer* and an *information layer*. The network bus in the control layer is compliant with FDDI connecting together proprietary subsystems such as programmable logic controllers (PLCs), and operator stations (OPSes). A proprietary protocol, called *cyclic transmission*, is used to achieve guaranteed end-to-end delays in the control layer among the subsystems. Each node (subsystem) in the network contains a dual port memory for cyclic transmission. The contents of each cyclic area are transmitted within some constant period to all the nodes. The information layer employs standard general purpose components for low cost. This layer contains an Ethernet LAN and a number of workstations (EWSes). The EWSes handle plant information management, plant maintenance, and report generation.

This two layer system architecture has some obvious problems:

- Lack of scalability: The contents of the cyclic memory areas are periodically transmitted even if there have been no changes in the data in those areas. This causes network resources to be wasted, thus limiting the amount of communication the network can support. In addition, the scale of the distributed system is limited by the size of the cyclic memory area.
- Fixed QoS (Quality of Service): The QoS is fixed by the cyclic transmission period and cannot be altered according to dynamic application requirements.

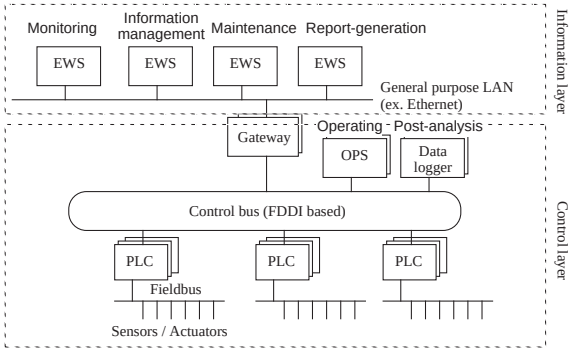

Figure 1: Configuration of current systems.

- Hierarchically layered network: Gateways are employed to connect the control bus and Ethernet. The gateways can be a system bottleneck for performance and reliability.

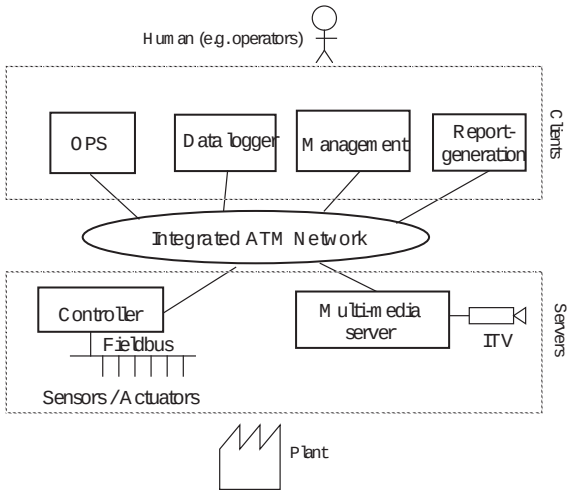

Figure 2: New system architecture with ATM network.

We envision that the four key technical advantages of ATM, i.e., *high bandwidth, end-to-end individual QoS, virtual connections* and *flexible topology*, will enable us to build future distributed plant control systems on open standard network technology with integrated communication services (including data, images, video and audio). This should lead to cost reduction, more flexibility and more dynamics in applications. Figure 2 represents a new, flexible system architecture of a plant monitoring and control system with an ATM network. The advantages of such a uniform network architeture include:

- Flexible QoS can be supported on per application basis. Traffic that needs deterministic guarantee and traffic that can tolerate larger transmission delay variation can potentially co-exist in the same network. For distributed real-time applications, we can readily utilize the concept of real-time channels [Ferrari 90].

- The configuration of the system can be easily modified by adding or removing subsystems on a plug-and-play basis, since we can employ standard platforms now.

- With a uniform network technology for the entire system, we have eliminated bottlenecks such as gateways which pose performance and reliability problems [1].

- Not all the real-time data need to be periodically transmitted — on-demand transmission of plant data or video can be accommodated.

- By integrating video and audio, new services can be provided. For example, alarm video can be implemented in which abnormal plant behavior can now be shown using video to operators on their workstations.

In order to obtain these advantages in a plant monitoring and control system, a new style of programming to construct distributed real-time systems is needed. In particular, each application must be able to request and acquire its own individual QoS. This paper presents a *real-time client-server programming model* for distributed real-time systems based on a uniform ATM network. This model is being realized in our middleware called *MidART* (*Mid*dleware and network *A*rchitec ture for *R*eal-*T*ime industrial applications). This middleware provides a set of industrial application specific but network transparent programming abstractions and application programming interface (API), as well as a set of QoS mapping schemes and protocols that translate application level real-time requirements into ATM QoS parameters and traffic descriptors, including schemes to ensure that this mapping maintains the timing guarantees at both the sender and the receiver. In addition, to achieve on-demand transmission of plant data, we have also developed a concept of *selective real-time channels*.

[1] Current commercially available ATM switches already support hot-swappable duplex switch boards (in the case of faults in primary board, the second board stands for immediately) for reliability and continuous operation.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the *MidART* architecture. In Section 3, we present our real-time client-server programing model and show programming abstractions on the model. In Section 4, we present the concept of selective real-time channels which provides plant monitoring and control applications the necessary abstraction to obtain data on demand, and enables the efficient usage of the network. Section 5 shows how to guarantee application-to-application QoS. We compare our work presented in this paper with other related work in Section 6, and the paper concludes with Section 7.

2 A Brief Overview of *MidART*

Figure 3 depicts the architecture of *MidART* with a two-node system, a client node and a server node. Each host in the network contains one copy of *MidART*. *MidART* includes three components, a RT-API, a RT-ATM, and a LAC. RT-ATM contains a set of QoS mapping schemes and protocols, and is responsible for translating application's QoS parameters into ATM service traffic descriptors and QoS parameters, multiplexing application traffic and establishing ATM virtual connections. [Shen 96] gives examples of how this translation and multiplexing can be accomplished. RT-API provides a set of programming abstractions and API as library routines to support application development. It also contains application task management facilities specific to industrial plant control application domain. LAC is responsible for checking the schedulability of tasks with respect to the resources in the end host systems.

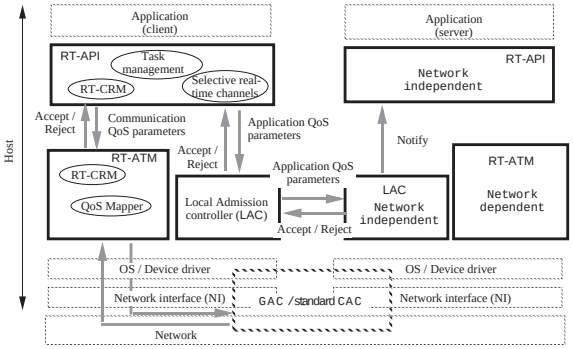

Figure 3: *MidART* architecture.

Application-to-application (or human-to-plant in our particular application domain) QoS guarantee includes end-to-end (i.e., network interface to network interface) guarantee and network interface to application process guarantee. To provide application-to-application QoS guarantee in a distributed real-time system, we must ensure that (a) sufficient resources (e.g., CPU, buffer space, network bandwidth, etc.) are reserved for the services that a user task needs in both the client and the server, and (b) that each service does not use more of the resources than what has been reserved for it. Thus we need admission control for the former and run time scheduling for the latter in both the host systems and the network. Systems implemented with *MidART* achieve this application-to-application QoS guarantee with the combination of (1) specifying application QoS with RT-API, (2) application to network QoS mapping by RT-ATM, (3) local admission control by LAC, (4) network admission control by GAC, and (5) real-time operating system task scheduling using commercially available real-time operating systems.

MidART prototype is being built on an ATM LAN based platform. The prototype system set up includes an ASX200BX ATM switch from FORE Systems, Inc. [FORE 95], and a number of HP 734rt computers with the HP-RT real-time operating system connected to the ATM switch using ATM adaptors from Interphase Inc[2]. The HP 734rt computers implement a Multimedia Server, an Operator Station and a set of sensors/actuators.

3 Programming Model and Programming Abstractions

3.1 Programming model

Figure 4 represents our real-time client-server programming model for industrial applications. In a distributed plant monitoring and control application, the most common physical components include sensors, actuators, (I/O or programmable logic) controllers, operator stations (OPSes), I/O devices, and database storage systems. Our real-time client-server programming model is designed to best exploit the advantages of ATM high speed networks for these applications and to facilitate ease of programming in this application domain.

Each subsystem in the model is classified as a client or a server. A client is a subsystem which is used by humans directly (e.g. OPS, information management, report-generation) or indirectly (e.g. database) via other subsystems. A server is a subsystem which monitors the plant with sensors or industrial TVs (ITVs), and controls the plant directly by actuators. For example, PLCs (programmable logic controllers) and a

[2]We are also investigating the possibility of using PCs with the LynxOS from Lynx Real-Time Systems, Inc as host systems. This, however, must depend on the availability of LynxOS drivers for ATM adaptors.

multimedia server with ITVs can be servers. Each client can request a server to send plant data or video of the plant and can request a server to issue commands to the plant specifying the QoS needed. The behavior of the servers is required to be as simple as possible such that it is predictable. More complicated operations and functions are usually done in clients. The servers are required to be reliable and must run continuously for a long time.

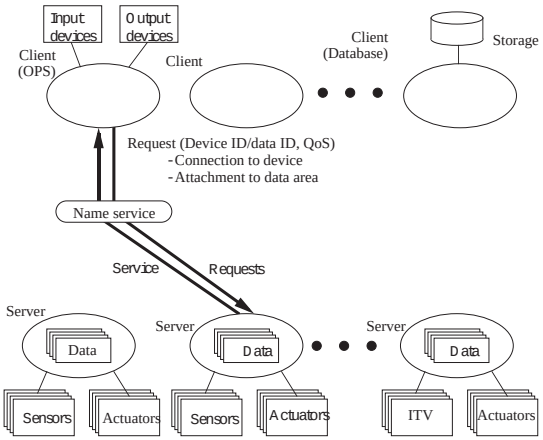

Figure 4: Real-time client-server programming model.

A server can be thought of as a certain kind of I/O controller with limited processing power (e.g., loop control only) and storage (memory) capacity. A server is usually connected to various input devices (e.g. sensors and ITVs) and output devices (e.g. actuators) via Fieldbus or other kinds of low-level bus. The services provided by a server may include sampling data from the input device, storing it as a byte stream, and sending control data to the actuator. The server has *data areas* in its address space. Data sampled from the input devices and set-up values for loop control can be stored in the data areas, for example.

3.2 Programming abstractions

In the distributed industrial plant control application domain, application processes often need to access remote devices in a timely fashion, or to share data with certain real-time update semantics and timing constraints. With the real-time client-server programming model described in section 3.1, we briefly introduce two new programming abstractions, *remote device access* (RDA) and *real-time channel-based reflective memory* (RT-CRM). These programming abstractions differ from similar general purpose ones in that they provide specific real-time support. In particular, QoS requirements such as data update rates or message deadlines can be specified and guaranteed.

RDA is a real-time unidirectional connection (the direction from the server to the client for an input device and the opposite direction for an output device) between a task in a client and a server task controlling a device. The task specifies the QoS parameters of the connection, an inter-arrival time of messages and their deadline. Once an RDA is established, the client task can receive messages from the input device or can send messages to the output device with the library functions. The rate control in the middleware on the sender side guarantees the inter-arrival time. RDA can be used to implement alarms and control functions.

RT-CRM is a software-based unidirectional reflective memory. It provides data reflection with guaranteed timliness, while allowing applications to specify how and when data are reflected. This is in contrast with the current commercially available hardware supported reflective memory in which each write is reflected to all the nodes in the network immediately. Data reflection can be done per period or upon a write. RT-CRM employs a *writer-push* data reflection model. The writer of a reflective memory area in RT-CRM is responsible for the creation of the memory area. After the creation, any remote reader application can *attach* itself to this reflective memory area for reading, and can specify its QoS parameters in terms of data reflection period, or delay bounds of data reflection upon writes.

4 Selective Real-Time Channels

4.1 Concept

At any instance, an OPS usually only need the plant data and video stream sampled by a small subset of the sensors or ITVs. The traditional cyclic transmission, however, transmits *all* of the sampled data at regular intervals limiting the amount of real-time transmission that can be supported in a network.

Assume that client C and m servers $S_1, S_2, \ldots, S_m$ are connected to each other via an ATM network with a switch. If we implement a cyclic transmission on the network, we need a bandwidth of $m \times B$ at the link between a client and the switch (figure 5). Here, suppose that the transmission rate from a server is constant and let it be B. It is obvious that a large m can easily saturate the bandwidth of the link.

In the case where a client actually needs data from only one server, for example, we need a bandwidth of only B at the link between the client and the switch. To achieve this, one obvious solution is to establish channels dynamically when a client requires data transmission from a specific server or ITV, and to employ the point-to-point real-time channels [Kandlur 94]. With this approach, however, it is difficult

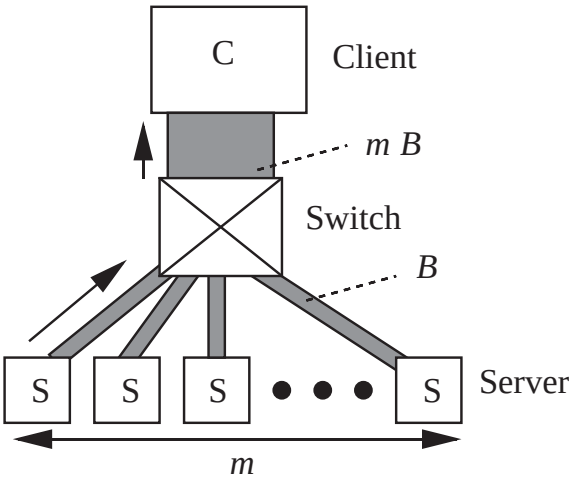

Figure 5: Bandwidth used with cyclic transmission.

to bound the time of channel establishment and it is not guaranteed that the channel with its QoS requirements can be always established successfully. In the case of OPS, response times to an operator's request to change the sensor or ITV for monitoring must be bounded, e.g., less than or equal to a few tens of milliseconds.

To solve the above problems and to utilize network resources efficiently, our *MidART* middleware manages network resources as follows:

Scheme BA (bandwidth allocation)
If a client requires data transmitted from at most k ($k << m$) (m is the number of servers) servers at the same time, assign bandwidth $k \times B$ at the link between the client and the switch SW_1 that the client is physically connected to, and assign bandwidth B at the link between each server and the switch SW_e that the server is connected to. For all the intermediate network links between SW_1 and SW_e, we assign a bandwidth of $min\{k \times B, s \times B\}$ to each of them, where s is the number of servers that actually go through the link.

We call a set of real-time channels between a client and m servers *selective real-time channels* if at most k ($k < m$) of them are used at the same time and the combination of k channels can be changed within a constant time (changed in real-time). Figure 6 represents the concept of selective real-time channels ($n = 1, k = 1$).

We show how selective real-time channels are realized with programming abstractions described in section 3.2.

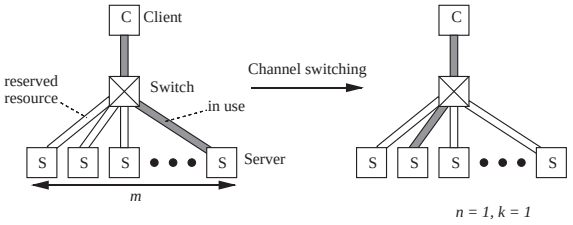

Figure 6: Concept of selective real-time channels.

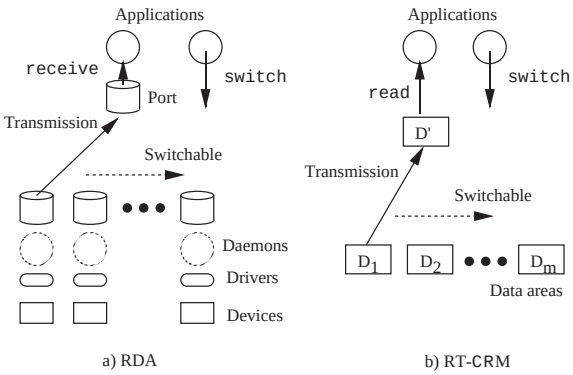

Figure 7: Programming abstractions with selective real-time channels.

Figure 7 a) shows an example of the selective real-time channels for RDA-based communications, called Switchable RDA. An application task in a client can communicate messages to application tasks in servers and vice versa via ports in the client and the servers.

Figure 7 b) conceptually represents the API of selective real-time channels for RT-CRM. Suppose that a data area D' in a client is shared with data areas $D_1, D_2, \ldots, D_m$ in the same server or in different servers. Data area D' and one of $D_1, D_2, \ldots, D_m$ can perform the one-way transmission or copy of their contents at any instance. A task in the client can read the content of D'. If the transmission is in the direction of the client to the servers, a task in the client writes the content. A task in the client can request the change or switch of the partner of data area D'. *MidART* guarantees that the time from the request to the finish of switching is bounded. In other words, a task in the client can read/write the contents of $D_1, D_2, \ldots, D_m$ in a bounded time including the switching times.

4.2 Channel Switching Algorithm

A set of selective real-time channels is established by the following algorithm (suppose that all of the

routes from clients to servers have been decided).

1. Establish a multicast real-time channel from each client to all servers (This channel is used by the client to send control messages to the servers).

2. Reserve bandwidth on each link of the routes from servers to the client according to **Scheme BA**.

Channel switching from server S_i to server $S_{i'}$ according to a request of an application in client C_j is performed by the following algorithm.

1. Client C_j sends all servers 'STOP' requests with the name of the server to be stopped (S_i). Then, the client waits for an 'ACK' message from the server S_i.

2. On receiving a STOP request, each server checks whether the server name in the request message matches its name. If it is, the server stops the transmission to client C_j and sends it an ACK message. Otherwise, the server ignores the message.

3. On receiving an ACK request, client C_j sends all servers 'START' requests with the name of the server to start transmission ($S_{i'}$).

4. On receiving a START request, each server checks whether the server name in the request message matches its name. If it is, the server starts the transmission to client C_j. Otherwise, the server ignores the message.

The algorithm above is one for the case of $k = 1$. It will be easy to extend the algorithm for the case of $k > 1$ by repeating the algorithm k times.

To reduce the bandwidth reserved for these messages at each link on the route, we use a multicast channel for control messages from clients to servers instead of using point-to-point real-time channels.

5 QoS Guarantee (Implementation)

In this section, we describe the mechanism which guarantees QoS, as a delivery delay and a transmission rate, in a system based on a real-time client-server model.

5.1 Application-to-application delay

This subsection shows that what kinds of primitive delays an application-to-application delivery delay consists of, then shows how to obtain their bounds. Let us treat two applications which communicate with each other via RT-CRM. It is obvious that the delays using RDA is simpler. Figure 8 and table 1 shows the primitive delays included in an application-to-application delay. Suppose that the deamon tasks in middleware are executed at higher priorities than those of applications. The transmission between the two hosts in this example is periodic.

Table 1: Primitive delays

D_{write}	The time from an interrupt of the input device to the activation of the application task in the sender host.
C_{write}	The time that the application task needs to write the sampled data into the data area PD.
T_{trans}	The period of the deamon task in the middleware in the sender host.
C_{mid_send}	The execution time of the deamon task.
C_{os_send}	The time that the OS needs to transmits the packets to the NIC.
D_{nic_send}	The time from the completion of the transmission of NIC until all packets have been transmited into the network.
$D_{network}$	The time that a cell needs to travel from the NIC in the sender to the one in the receiver.
D_{nic_recv}	The time from the arrival of the last cell to the NIC in the receiver to when all packets have been reassembled.
D_{os_recv}	The time that the OS needs to load all packets which have been reassembled by the NIC into application memory.
C_{mid_recv}	The time from when the deamon task in the middleware is activated by the OS to when the task finishes all required operations.
T_{recv}	The period of the application task in the receiver host.
D_{read}	The time from the start of the period of the task to that when the task is actually activated. This time occurs because of the preemption of tasks with higher priorities.
C_{read}	The execution time of the application task.

If we can obtain the maximum values of the all primitive delays above, we can estimate the application-to-application delay D_{atoa} as the sum of the primitive delays in table 1. Here, suppose that the time of interrupts and the time of context switch can be neglected. We can estimate the maximum values of all the delays as follows.

D_{write} and D_{read} are waiting times for resources such as CPUs and semaphores. Their maximum values are calculated using rate monotonic algorithm (RMA) [Liu 73] [Lehoczky 89] with its extensions for resource contention [Rajkumar 91] by LACs in the hosts. C_{write}, C_{os_send}, and C_{read} are estimated as linear functions of the data size S. C_{mid_send}, C_{mid_recv}, T_{trans}, T_{recv}, and D_{os_recv} are constant. D_{nic_send} and D_{nic_recv} depend on the cell scheduling algorithm of the NICs and traffic characters such as minimal inter-arrival times of other channels in the NICs. Here, we assume that the scheduling algorithm is FIFO or fixed priority based and the minimal inter-arrival times of cells in all channels are known. With this assumption, one can calculate D_{nic_send} and D_{nic_recv} based on a usual RMA. How to calculate $D_{network}$ is described in subsection 5.2.

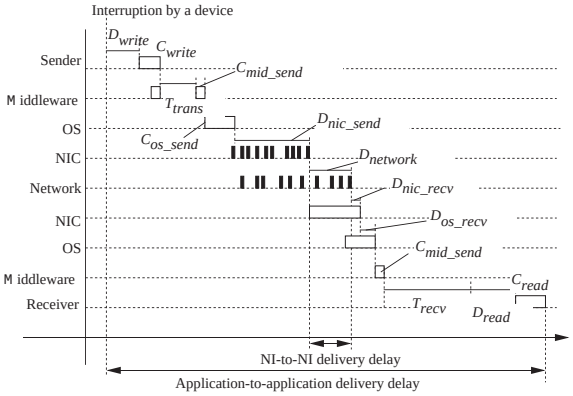

Figure 8: Application-to-application delivery delay.

5.2 Network delay and bandwidth

As delays in the NICs, the delay in the network, $D_{network}$, depends on the scheduling algorithm of the switches and the traffic characters of the other channels going though the switches.

Signalling and CAC algorithms of current commercial ATM products mainly support point-to-point or point-to-multipoint connections, not multipoint-to-point connections which are needed for selective real-time channels. For data transmissions in selective real-time channels, we use PVC connections and FORE ASX-200 BX's functions of bandwidth and buffer overbooking for VBRs [FORE 95]. This means that current selective real-time channels are able to be established only statically, not dynamically.

We show how to realize selective real-time channels with PVC and BX's overbooking. Suppose that we have a client and m servers and the bandwidth from server $S_i (i = 1, \ldots, m)$ to the client is B_i. In a case where at most $k (k << m)$ servers send data at the same time, we can give an integer value of $\lfloor \frac{\sum_i B_i}{k \cdot \max_i B_i} \times 100 \rfloor$ as the parameter `vbrob` at the input link of the client, for example. Here, the switch assign a VBR channel bandwidth of $100 \times 100/\text{vbrob}\%$ of that required to assign. For transmissions of control messages from the client to the servers, we can use a CBR multicast channel. [3]

If we can obtain an ATM platform which supports multipoint-to-point connections in future, GAC will map application requirement for communication (C, T, D) onto VBR parameters with calculating $D_{network}$ by the method proposed in [Zheng 94a] or that in [Mizunuma 95], for example.

[3] We can also use CBR unicast channels for the transmission. This approach lacks scalability in the number of servers, but bandwidth assigned for control messages is very narrow and we believe that we have no problem with at most one hundred servers in an actual application.

6 Related Work

There have been a couple of research efforts in experimenting with introducing early ATM network systems into industrial systems. The *Mercuri* ATM testbed [Guha 95] [Pavan 95] is a process monitoring system with first generation standard ATM platforms. Although *Mercuri* also provides a client-server model to applications, the objective of the *Mercuri* testbed differs from ours in that their focus is mostly to experiment with multimedia data (video and audio) on ATM networks. Thus message throughput and round trip delay, as well as video frame jitter are their main concerns for performance measurements, rather than application-to-application deterministic QoS guarantee. On the other hand, we are focusing on providing programming models and mechanisms for QoS specification and guarantee.

[Pokam 95] also proposed using ATM networks to incorporate multimedia in process control systems. Their system, however, is not scalable due to their ring topology. In addition, their approach needs special hardware for communication interface (STRAM communication boards) rather than using standard platforms.

Task scheduling and QoS management in middleware or operating systems for multimedia end systems have been the focus of numerous research efforts recently [Mercer 94], [Nishio 94], [Kawachiya 95], [Jeffay 95]. Although most of them are not dealing with industrial plant control systems with deterministic guarantee requirements, we can still find similarities in our respective projects.

The approaches we are taking for our host system admission control in LAC and the application to operating system QoS mapping supported in RT-ATM are similar to the reservation concept in [Mercer 94] and [Jeffay 95]. One difference lies in that we use commercially available real-time operating systems (i.e., HP-RT or LynxOS) which are monolithic, rather than micro-kernel based technology.

A middleware for continuous media applications is presented in [Nishio 94] and [Kawachiya 95]. A conductor/performer architecture is used to guarantee QoS of continuous media and to provide dynamic QoS control. The middleware is a set of system servers on the RT-Mach micro kernel. The focus of this work is very different from ours — dynamic QoS control among multiple sessions in one host system for continuous stream of video data.

There have been a lot of studies lately on real-time

communication with end-to-end (network interface-to-network interface) QoS guarantees in a packet switching network or an ATM network [Kandlur 94], [Raha 96], [Zheng 94a], [Zheng 94b] and [Mizunuma 95]. Most of these work are based on the cencept of real-time channels [Ferrari 90] and derive delay bounds given packet scheduling algorithms and traffic source specifications. These are important results that we expect the next generation ATM networks can make use of.

7 Conclusion

In this paper, we have given an overview of our middleware *MidART* (*Mid*dleware and network *A*rchitecture for *R*eal-*T*ime industrial applications), and presented the details of our new real-time client-server programming model for distributed industrial plant control applications. For ease of use and predictability, our programming model and API has stressed simplicity and uniformity. *MidART* is one of the first efforts in introducing the emerging ATM technology into distributed industrial plant control applications. We are currently implementing and evaluating our middleware on an ATM LAN platform.

MidART is an on-going effort. Besides the evaluation of its real-time performance, some of the open issues that we are continuing to address include signalling algorithms at the ATM layer to facilitate the efficient implememtation of selective real-time channels , as well as effective QoS mapping algorithms for the RT-ATM component of *MidART*.

Acknowledgements

The authors would like to thank Dr. Satoshi Horiike of IES Lab., Mitsubishi Electric Corp. and Dr. Qin Zheng of MERL for their valuable discussions during the course of the research reported in this paper.

References

[ATM Forum TM 4.0] ATM Forum Technical Committee. Traffic Management Specification 4.0 April 1996.

[ATM UNI 3.1] ATM Forum Technical Committee. ATM User-Network Interface (UNI) Specificcation Version 3.1 Sept. 1994.

[Ferrari 90] D. Ferrari and D. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. IEEE Journal on Selected Areas in Communications, 8(3):368–379, April 1990.

[FORE 95] FORE Systems, Inc. *ForeRunner*^{TM} ASX-200BX ATM Switch User's Manual, MANU0026 - Rev. D. September 1995.

[Guha 95] A. Guha, A. Pavan, J. Liu, A. Rastogi, and T. Steeves. Supporting Real-Time and Multimedia Applications on the Mercuri ATM Testbed. The IEEE Journal on Selected Areas in Communications, Vol.13, No.4, pp. 749–763, 1995.

[Jeffay 95] K. Jeffay and D. Bennett. A Rate-Based Execution Abstraction for Multimedia Computing. The Fifth International Workshop on Network and Operating System Support for Digital Audio and Video. pp 67-78. April 1995.

[Kandlur 94] D. D. Kandlur, K. G. Shin, and D. Ferrari. Real-time Communication in Multihop Networks. IEEE Transactions on parallel and distributed systems, Vol. 5, No. 10, pp. 1044–1056, Oct. 1994.

[Kawachiya 95] K. Kawachiya, M. Ogata, N. Nishio, and H. Tokuda. Evaluation of QOS-Control Servers on Real-Time Mach. The Fifth International Workshop on Network and Operating System Support for Digital Audio and Video. pp 123-126. April 1995.

[Lehoczky 89] J. P. Lehoczky, L. Sha, and Y. Ding. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. IEEE RTSS '89, pp. 166–171, December 1989.

[Liu 73] C. L. Liu and J. W. Layland. Scheduling Algorithm For Multiprogramming in a Hard-Real-Time Environment. JACM, Vol. 20, No. 1, pp. 46–61, January 1973.

[Meinert 95] G. Meinert. Openness for Automation Networks. *In*Tech, pp 29-32. Oct. 1995.

[Mercer 94] G. Mercer, S. Savage and H. Tokuda. Processor Capacity Reserves: Operating System Support for Multimedia Applications. IEEE International Conference on Multimedia Computing and Systems. pp 90-99. May 1994.

[Mizunuma 95] I. Mizunuma, S. Horiike, and M. Takegaki. Real-Time Communication in Plant Monitoring / Controlling Systems with ATM Networks. RTCSA'95, pp. 243–250, Oct. 1995.

[Nishio 94] N. Nishio and H. Tokuda. Conductor-Performer: A Middle Ware Architecture for Continuous Media Applications. 1st International Workshop on Real-Time Computing Systems and Applications, pp.122-131, 1994.

[Pavan 95] A. Pavan, J. Liu, A. Guha, J. Pugaczewski, and M. Midani. Experimental Evaluation of Real-Time Support on the Mercuri Wide Area ATM Testbed. LCN '95, pp. 82–91, 1995.

[Pokam 95] M. R. Pokam, J. Guillaud, and G. Michel. ATM Networks Allow Multimedia in Process Control Systems. IEEE ICCCN '95, pp. 324–331, 1995.

[Raha 96] A. Raha, S. Kamat, and W. Zhao. Admission Control for Hard Real-Time Connections in ATM LANs . IEEE INFOCOM '96, March 1996.

[Rajkumar 91] R. Rajkumar Synchronization in Real-Time Systems: A Priority Inheritance Approach. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061 USA. 1991.

[Shen 96] C. Shen. On ATM Support for Distributed Real-Time Applications. IEEE RTAS '96, pp. 70–78, June 1996.

[Zheng 94a] Q. Zheng, K. Shin, and C. Shen. Real-Time Communication in ATM Networks. LCN '94, pp. 156–164, Oct. 1994.

[Zheng 94b] Q. Zheng and K. Shin. On the Ability of Establising Real-Time Channels in Point-to-point Packet-switched Networks. IEEE Transactions of Communication, Vol. 42, No. 4, pp. 1095–1105, 1994.