MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Real-Time Communicating Tasks on COTS-based Distributed Platforms: Task Models and End-to-End Scheduling

Chia Shen, Oscar Gonzalez

TR98-17 December 1998

Abstract

This paper describes our current work on scheduling communicating real-time tasks in a distributed environment. Unique challenges are presented when one tries to build distributed real-time applications using standard off-the-shelf systems which are in common use but are not necessarily designed for real-time systems. In particular, one must deal with (1) mapping application real-time requirements into system schedulable entities, (2) end-to-end scheduling in the face of possible priority inversion, (3) limited real-time scheduling support and limited number of priorities, and (4) integrating real-time and non-real-time tasks in the same platform. Due to space limitations, this paper focuses on solving the first two challenges. The complete solution will be presented in a forthcoming paper. We have implemented these solutions in our network middleware MidART running on PCs with Windows NT operating system over Ethernet LANs.

1998 Real-Time Systems Symposium, Work-In-Progress Proceedings, December 2-6, 1998. Madrid, Spain

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 1998
201 Broadway, Cambridge, Massachusetts 02139

Real-Time Communicating Tasks on COTS-based Distributed Platforms: Task Models and End-to-End Scheduling

Chia Shen
MERL - A Mitsubishi Electric Research Lab.
201 Broadway
Cambridge, MA 02139

Oscar González*
Computer Science Department
University of Massachusetts
Amherst, MA 01003

1 Introduction

It is becoming more and more evident that distributed industrial real-time applications, such as process control, factory automation and plant control systems, will move towards using open, standard, commercially available and more general purpose computers, operating systems and networks. For example, more expensive workstations will be replaced with off-the-shelf PCs, and adoption of Windows NT will allow for increased control at the PC level [5]. There is also a lot of momentum toward making control network IP-based [2]. Moreover, Ethernet offers cost and support advantages over industrial fieldbuses as a control and device-level network [1]. MidART [7, 4, 6] is an ongoing real-time network middleware project to lead this movement towards open and integrated system construction for industrial applications. MidART is a network middleware with a distributed real-time application development software package providing easy-to-use programming interface for real-time data acquisition and communication.

Unique challenges are present when one tries to build distributed real-time applications using standard off-the-shelf systems which are in common use but are not necessarily designed for real-time systems. In particular, one must deal with (1) mapping application real-time requirements into system schedulable entities, (2) end-to-end scheduling in the face of possible priority inversion, (3) limited real-time scheduling support and limited number of priorities, and (4) integrating real-time and non-real-time tasks in the same platform. In this paper, we propose solutions to these challenges, and implement our solutions in MidART running on PCs with Windows NT operating system over UDP/IP and Ethernet LANs. Due to space limitations, this paper focuses on solving the first two challenges. The complete solution will be presented in a forthcoming paper.

*The research has been done during this author's internship at MERL.

2 MidART and Its Application Domain

To understand how we solve the end-to-end scheduling problem, we need to first give a brief overview of MidART and how the application tasks are supported.

2.1 MidART Overview

The MidART middleware provides a set of real-time application specific but network transparent programming abstractions that support individual application data monitoring and control requirements. The focus of our middleware is to support the end-to-end application real-time data transfer requirements with a set of easy-to-use communication service programming interfaces. The two key services provided by MidART are Selective Channels [6] and Real-Time Channel-Based Reflective Memory (RT-CRM) [7].

Selective Channels allow applications to dynamically choose the remote node(s) which data is to be viewed from and sent to at run time. This is accomplished via a set of channel start and stop protocols, and channel bandwidth resource overbooking schemes.

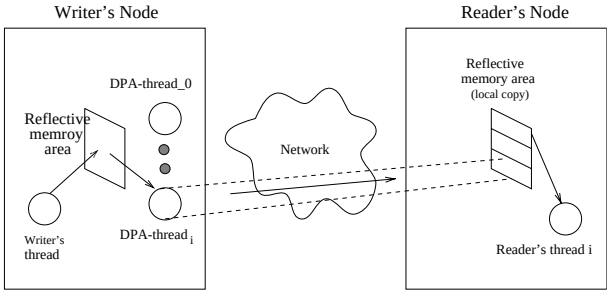

Figure 1: RT-CRM High Level Architecture

Figure 1 depicts the high level architecture of RT-CRM. RT-CRM is an association between a writer's memory and a reader's memory on two different nodes in a network with a set of protocols for memory channel establishment and data update transfer. A writer has a memory area where it stores its current data (e.g., a PLC stores all the sensor data), while a reader

establishes a corresponding memory area on its own local node to receive the data reflected from the writer (e.g., an operator station receives and displays monitoring data). Data reflection is accomplished by a data push agent thread, a DPA-thread, residing on the writer's node and sharing the writer's memory area. This agent thread represents the reader's QoS and data reflection requirements. A virtual channel is established between the agent thread and the reader's memory area, through which the writer's data is actively transmitted and written into the reader's local memory area. In this architecture, we support the following features:

- A reader memory area may be connected to multiple remote writer memory areas simultaneously. However, at any moment only one writer is permitted to write into the reader's memory area via the associated agent thread. The selection of the particular writer at any time is done via Selective Channel protocols.
- A writer memory area may be connected to many remote reader memory areas simultaneously. Data is pushed to each reader according to their individual requirements.

In particular, given a reflective memory area D, since a DPA-thread is a separate thread of control from the writer's application thread, RT-CRM can support the following types of data push operations and data reception operations:

- Data Push Operations:
  - *Synchronous Data Push*: Pushes are triggered by application writes.
  - *Asynchronous Data Push*: Pushes are performed periodically, with independent timing from that of the writer's application.
- Data Reception Operations:
  - *Blocking Read*: Application reads block while awaiting the arrival of a data update message from the writer's node. When the message is received, the reader application thread is signalled.
  - *Non-Blocking Read*: Application reads return the current contents of the reflective memory area. That is, the reader's application will not be notified upon the arrival of data update messages.

For more details of Selective Channels and RT-CRM, readers are refered to [6] and [7]. The uniqueness of MidART lies in the simplicity of services provided and the flexibility of data reflection models. This simplicity leads to ease of understanding and ease of use by application builders, while its flexiblity sufficiently serves the needs of the class of real-time applications MidART is designed for.

2.2 Application Characteristics

To design approriate scheduling algorithms, we must first understand what is the application domain at hand. For industrial applications, we have derived the following characterization of the most common types of distributed application tasks:

- Command and control: Sporadic activities which need immediate data transmission and delay bounded data reception.
- Video/audio: Periodic transmission, requiring low jitter display.
- Plant monitoring: Sporadic data collection and transmission, including alarms, and immediate data display upon reception.
- Trend graph: Periodic data collection and transmission, and periodic data display upon reception.
- Background: Non-real-time activities such as logging data to disk, reviewing video segments, and sending email. Best effort data transmission and reception.

With the set of data push and reception operation modes provided by MidART, we can support the above application requirements with many combinations of operation modes. Table 1 lists two possible such combinations – SB and AN.

Modes	Data Type	Deadline	Application
SB	Sporadic	G	Command issuing
AN	Periodic	G	Trend graph
SB	Sporadic	G	Plant data
AN	Periodic	G	Video/Audio
SB	Sporadic	NG	Background

Table 1: S = Synch., A = Async., B = Blocking, N = Non-blocking, G = Deadline Guaranteed, NG = No Guarantee.

In the rest of this paper, we will concentrate on developing solutions for the challenges listed in the Introduction, i.e., mapping of application requirements into schedulable entities, end-to-end scheduling, limited number of priorities and accommodating both real-time and non-real-time application tasks in the same system, on COTS-based platforms.

3 Integrated End-to-End Scheduling

It is clear from the previous section that any scheduling solutions for the industrial application domain must provide support for:

- Periodic and sporadic real-time tasks with end-to-end deadlines. For example, an operator command-and-control task usually requires a specific delay bound from the command issuing time to the command actuation time.

- Non-real-time tasks which can be either dynamic or periodic.
- Jitter control.
- Communications with bounded delay.

The features and problems often found in contemporary operating systems, such as Windows NT, IRIX and Solaris, include:

- Features:

  Preemptive priority-based scheduling and/or round-robin scheduling.

  Non-degradable priorities.

  Mechanisms for priority adjustment (e.g., set priority of a process or thread to a different level).

  Periodic timers to trigger periodic events and release of thread execution.

- Problems:

  No priority inheritance among processes/threads, and no priority tracking from user threads to system/network protocol stack threads – This makes real-time end-to-end scheduling with network communication very difficult. For example, the execution of socket level calls do not necessarily follow the priorities assigned to the threads which make the socket calls.

  Limited number of priorities – This implies that we cannot use a unique priority for all the real-time tasks.

  No specific support for guaranteeing timing constraints for tasks besides basic priority-based scheduling.

3.1 Model Precedence Constrained Tasks with Release Jitter

In general, when tasks need to communicate over IP across a LAN, the end-to-end computation and communication entities include the application threads, socket level send and receive, network interface and network transmission. In particular, Figure 2 depicts the tasks involved in an end-to-end scenario[1]. The Application writes and reads are application processes/threads that include MidART library calls. ReMA is the Reflective Memory Area, set up as shared memory between the applications and MidART. DPA and DRA are Data Push and Data Receive Agent threads in MidART, both of them use sockets for communication over UDP/IP. *mbuf* is the memory used by the sockets. Since we have no control of the network interface and network transmission, these two have been merged into the black box of Network. C_x is the worst case computation/communication time of the respective task entity x.

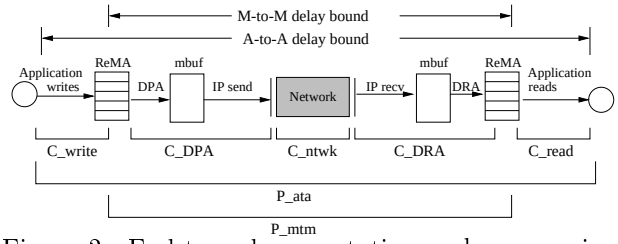

Figure 2: End-to-end computation and communication.

In [7], we have classified end-to-end into application-to-application, and memory-to-memory. Some of the application activities, such as command-and-control, require application to application (A-to-A) delay bound guaranee, while the others, such as trend graphs, only require memory to memory (M-to-M) guarantee. To support A-to-A, we must schedule all the tasks from application writes to application reads as a precedence constrained task graph. On the other hand, to support M-to-M, we can schedule application writes and reads independently, while treating all the rest of the tasks as one precedence constrained task set. Therefore, in Figure 2 P_ata and P_mtm are the periods of an A-to-A task set and a M-to-M task set respectively. Note that the synchronous data push as well as the blocking read modes impose precedence constraints between the entities involved by definition, while the asynchronous data push and the non-blocking read modes support independent task representation naturally.

To enable the end-to-end scheduling (for both A-to-A and M-to-M) on any COTS-based platform with only priority-based scheduling support, we model precedence constrained tasks as independent periodic tasks with release jitter as follows.

- Let J_y be the jitter of task entity y.
- In the case of A-to-A:

  $J_{write} = 0$, $J_{DPA} = C_{write}$,
  $J_{DRA} = J_{DPA} + C_{DPA} + C_{ntwk}$,
  $J_{read} = J_{DRA} + C_{DRA}$.

- In the case of M-to-M:

  $J_{write} = 0$, $J_{read} = 0$,
  $J_{DPA} = 0$, $J_{DRA} = C_{DPA} + C_{ntwk}$.

To map application timing requirements into our system schedulable components, we take the following approach:

- Application writers and readers specify their periods/delay bound, synchronous or asynchronous data reception, as well as blocking or non-blocking data retrieval semantics.

[1] This is a simple pictorial representation. The computation time and memory size are not drawn to scale.

- With reference to Table 1, timing requirements are mapped as follows:

  (1) SB mode: The writer's period and delay bound are used for all the tasks end-to-end, and the release jitters of tasks are calculated according to the A-to-A case as above.

  (2) AN mode: Reader and Writer will have their independent periods/delay bounds. Computation and communication tasks C_DPA, C_ntwk and C_DRA will use reader's period and delay bound, and their respective release jitter will be calculated according to the M-to-M case as above.
- All the C_x values are derived via a MidART profiler. Due to space limitation, we will not discuss this further.

3.2 Priority Inversion Prevention

In order to prevent or minimize priority inversion at the socket level and at the network, We deviced three schemes – rate control, uniform message size, and asynchronous DPA server for all asynchronous data push operations. Through a MidART profiler, we identify the optimal message size S^o for any particular network setup. The message size will serve as the non-preemptive unit of computation time and transmission time (Thereafter, we will use S^o as both the unit message size and the time needed to send and receive a message of this size.). All data to be pushed is divided into messages of this unit size. To ensure that a higher priority data push request only suffers blocking time due to priority inversion for at most S^o time units, instead of using a DPA thread for each corresponding remote reader application, we employ a DPA server (ADS) that pushes data on behalf of all the asynchronous data push requests. Since asynchronous data pushes are all periodic with possible release jitters, ADS implements a modified dual priority scheduling algorithm [3]. The ADS acts as the central scheduler and rate controller for all the asynchronous data pushes that have been admitted into the system.

3.3 Mapping Applications to System Priorities

Here we present how we deal with the problem of limited number of priority levels. In Section 2.2, we described five types of application activities. They represent tasks with real-time constraints, as well as task without real-time requirements. The real-time activities fall into synchronous data pushes and asynchronous data pushes. In general, the synchronous data pushes are more urgent, time critical tasks, while the asynchronous ones are periodic in nature. The non-real-time activities can be serviced with best effort. With this characterization, we have developed the following scheme for mapping application activities into system priorities:

- All end-to-end tasks requesting mode SB will be assigned the highest priority $P^{highest}$. Usually, there are only one or two such application entities in a system.
- All the application entities requesting mode AN with timing constraints will be serviced by ADS according to their respective periodicity and data size. As described in the last subsection, ADS implements a dual priority algorithm such that each communication request is not serviced until its promotion time. Thus, in essense, the ADS resides in a lowest priority P^{low} until the promotion time of some task, at which time ADS is promoted to a high priority $P^{high} < P^{highest}$.
- All the non-real-time activities are assigned a priority that is between P^{high} and P^{low}.

4 Status and Future Work

With limited space, we have very briefly described our current work on end-to-end scheduling of real-time communiating tasks on COTS-based distributed systems. We are developing our scheduling scheme on a PC-based Windows NT platform with Ethernet as the LAN. We have already obtained initial experimental results to show how our server-based solution alleviates the socket level priority inversion problem. We are also developing analysis to show the quantitative bound of our end-to-end dual priority scheduling algorithm with release jitters. In a forthcoming paper, we will demonstrate the full potential of our task modeling and scheduling approach.

References

[1] Dick Caro and Rich Mullen. Ethernet as a Control Network. *CONTROL Magazine, Putman Publishing Co*, February 1998.

[2] Deborah Claymon. Control Freaks: Control networks will regulate every factory, house, and office . *The Red Herring*, March 1998.

[3] R. Davis and A. Wellings. Dual Priority Scheduling. In *IEEE Real-Time Systems Symposium*, December 1995.

[4] O. Gonzalez, C. Shen, I. Mizunuma, and M. Takegaki. Implementation and Performance of MidART. In *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*, December 1997.

[5] John A. Hill. 10th Annual Control Market Outlook: Software Taking Control. *InTech: The International Journal for Measurement and Control*, January 1997.

[6] I. Mizunuma, C. Shen, and M. Takegaki. Middleware for Distributed Industrial Real-Time Systems on ATM Networks. In *17th IEEE Real-Time Systems Symposium*, December 1996.

[7] C. Shen and I. Mizunuma. RT-CRM: Real-Time Channel-Based Reflective Memory. In *IEEE Real-Time Technology and Applications Symposium*, June 1997.