Extensible Real-Time Data Dissemination on Channel-Based Reflective Memory

Kazuya Kitsunai
Dept. of Information and Computer Science
Keio University
Yokohama, Kanagawa, 223-8522, Japan

Hidenori Kobayashi
School of Science for Open and Environmental Systems
Keio University
Yokohama, Kanagawa, 223-8522, Japan

Nobuyuki Yamasaki
Dept. of Information and Computer Science
Keio University
Yokohama, Kanagawa, 223-8522, Japan

Yuichiro Anzai
Dept. of Information and Computer Science
Keio University
Yokohama, Kanagawa, 223-8522, Japan

Abstract

In this paper, we propose real-time communication middleware which supports data dissemination with proxy extension. To fulfill stringent QoS requirement such as timing constraint, we employ push-based data delivery approach and real-time channel. Furthermore, by making the most of connected clients, we could permit the increased number of clients and extend the scalability. Since it is also desirable for distributed real-time system to support real-time communication specifications by hardware, we used *Responsive Processor* as experimental platform.

1 Introduction

Recently, time criticality of data dissemination is required for real-time network-base application such as distributed monitoring and control system, video transmission and video conferencing. However, such as HTTP in World Wide Web, server-client model has employed *client pull-based data dissemination* where a client sends request packets to a server and receives request packets. Since arrival time of data in the client side cannot be bounded, client pull model is not suitable for distributed real-time application.

Many researches focus on *server push-based data dissemination*[1], [2] that correspond to stringent time criticality. In RT-CRM[5], dedicated reflective memory area is reflected via channel through a network. By combining both reflective memory model and channel properties, it can fulfill time constraint with flexibility where server is able to push data to any node.

However they do not focus on access locality of servers. If so many clients access to limited numbers of server, these servers suffer excessive overhead and then they cannot offer any more service. Therefore, access limitation has been low. To fulfill more client's requests, load balancing of convergent server is needed.

In this paper, we address load balancing of real-time data dissemination application with stringent timing constraints and try to support the increased number of clients. Our proposed model is based on the idea that there are so many clients who have the same replica when the server suffers excessive overhead from numbers of clients. We use these clients as temporary proxy server which can serve the same data in server. The rest of this paper is organized as follows. In Section 2, we indicate our target system architecture overview and design of reflective memory model for it. In Section 3, we showed the details of *Responsive Processor* which we use to construct our experimental distributed real-time system. In Section 4, concluding remarks.

2 Design

2.1 System Overview

There's been a shift from centralized to parallel/distributed in real-time system. The development and increasing bandwidth of data delivery enables us to construct distributed network-based control system. Although the availability of high-bandwidth advances, it is much more important for communication middleware to support real-time QoS guarantees in order to construct distributed network-based control system.

In Fig.1, we showed the example of distributed network-based system which we assumed in this paper. This type of system connects the numbers of nodes such as terminal, multimedia server for video transmission, system surveillance and etc, sensors and actuators. Each type of node has different communication properties. The data size of multimedia server is

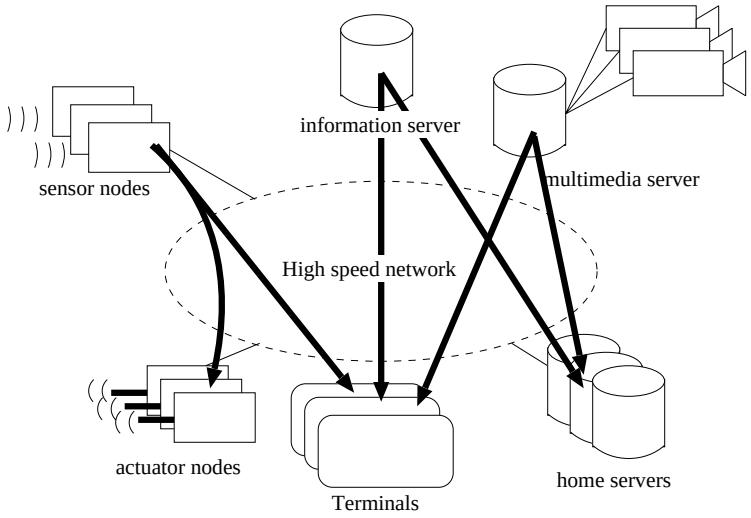

Figure 1: Distributed network-based control system

commonly large, but communication is bearable with soft real-time guarantee. In contrast, the data size of sensor and actuator nodes is small, but for hard real-time guarantee, it is not desirable to control the system.

The number of nodes in the system we assumed is not limited. To apply our model on the common Internet, the number of clients(terminal) and servers is generally unexpected. Also, we assumed the system topology is multihop network(i.e. a message pass through multiple intermediate nodes). In order to fulfill the timing constrained in such system, utilizing the proxy in our model is acceptable.

2.2 Reflective Memory Model

To achieve various QoS requirement from clients, we use real-time channel[3] where resources will be reserved for them after the connection establishment operation, as well as freed during connection disestablishment. The route of a channel will be chosen at the time of its establishment. Although there are many routing and resource management algorithms, it is out of domain that we deal with in this paper. Our goal is to construct real-time communication middleware on real-time channel. So we assumed that there might be real-time channel between server nodes and client nodes.

Our channel-based reflective memory model is based on [5] where Reflective Memory Server (RMS) is in both server node and client node. RMS manages Reflective Memory Table(RMT), Data Push Agent(DPA) attributes(i.e. data pushing period, starting/stopping DPA, periodic/aperiodic/sporadic data push, etc) and connection requests from client nodes.

In Fig.2, we showed our modified model of it. In our model, the writer thread(e.g. sensing data management, graphics transmitting) in server node creates Reflective Memory Data(RMD) and registers it on RMT managed by RMS. RMT can be globally referenced by all threads in its node. In RMT, Reflective Memory Identifier(RMID) is defined which is unique in the system. RMID is pair of (original server node ID, memory area ID). Because of this pair, we can respectively manage RMD in every node of the system. When a reader thread(e.g. terminal) in a client node would like to receive the server's RMD and connect to it, it sends a request packet to RMS thread in the server node. When the server's RMS thread receives the request from it, it tries the admission test. If passes, the RMS thread creates a DPA thread to reflect(send) RMD to it. The DPA thread manages sending data on real-time channel along the reader's QoS requirement. The reader thread can start/stop its dedicated DPA thread to flexibly change the receiving data it wants.

2.3 Proxy Extension

On the other hand, client node can register its receiving RMID if it allows to become a proxy node for

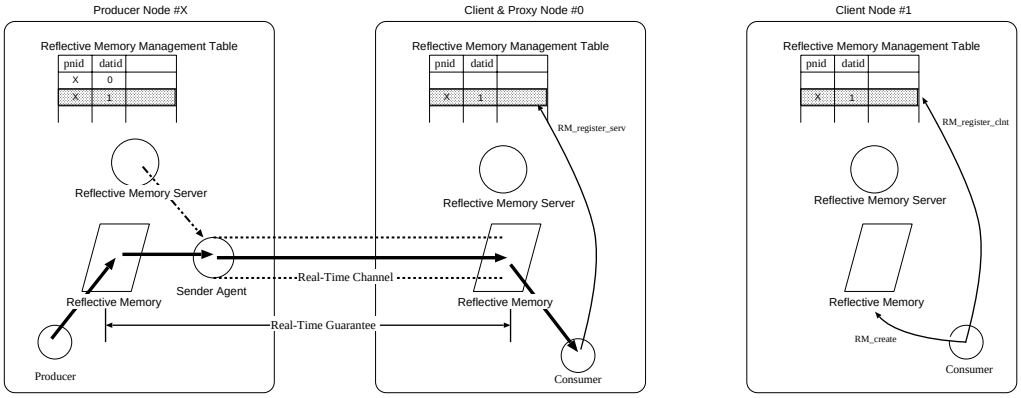

Figure 2: RM(X,1) is reflected to client #0

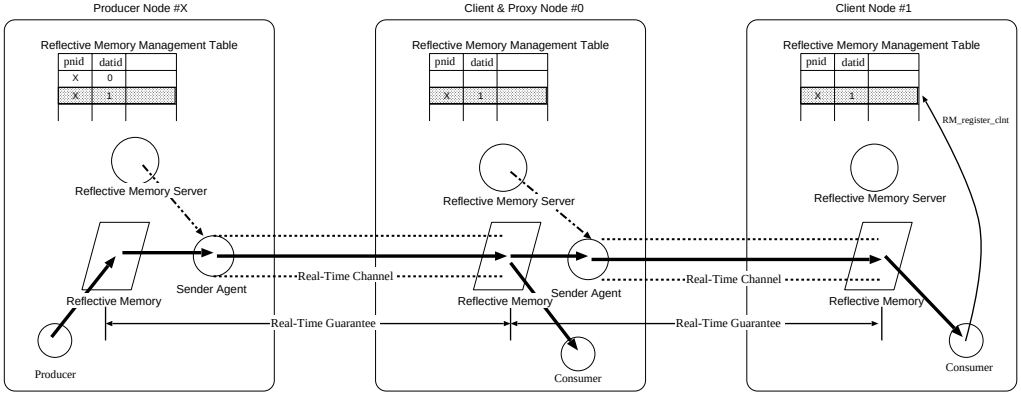

Figure 3: RM(X,1) is reflected to client #1 via proxy(client #0)

future clients. With registering RMID, another clients whose request was rejected from the server can connect to the client which has the same RMID in the original server node.

Fig.2 and Fig.3 depict our proxy extension scheme discussed above. In Fig.2, the client #0 connected to the server #X and registers its RMID in its RMT. Its RMID must be identical with the server's RMID. With registering its RMID and fulfilling client #1's QoS requirement, the client #0 can serve as a substitute(proxy server) for the original server #X. When proxy's RMS receives request from another client, proxy's RMS tries the admission test and connect to the requested clients as well as the original server does.

Also, it is worth noting that with unique RMID in the system showed in the last section, client nodes can search elective RMD without incurring server node. In other words, it is not necessary for client nodes to send requests to server node.

3 Experimental Environment

To evaluate our model, we implemented it on *Responsive Processor*[6]. *Responsive Processor* is a system-on-a-chip in which RISC processing core(SPARClite 120.0MHz), four Responsive Link for real-time communication, various computer peripherals (SDRAM I/Fs, PCI, USB, DMAC, timers, SIO, PIO, etc.) and various control peripherals (A/DC, D/AC, pulse counters, PWM, etc.) into single chip.

Responsive Processor is designed for parallel/distributed control in real-time. Therefore it has real-time communication architecture called *Responsive Link*, in which a line for data communication(data link) is separated from a line for event communication(event link). A link is point-to-point and full-duplex. Generally, if packet size is large, it offers better bandwidth, on the other hand, if packet size is small, it guarantees the latency. The data link realizes soft real-time communication. Therefore its packet

size is fixed length and large(64B). To the contrary, the event link is for hard real-time communication, packet size of which is fixed length and small(16B). The *Responsive Link* also performs overtaking lower priority packet when a packet collision occurs. The priority of a packet is added on packet header section, in which source/destination address is appended to. The priority attachment and replacement on a packet are conducted by software. The routing table of the event/data link can be separately set up, so that the event/data link can be routed independently.

Furthermore, we developed parallel/distributed real-time operating system called *RT-Frontier* on it. To utilize the Responsive Link properties, it supports three communication paradigm which is individually called event message, state message and signal. Event message is transmitted on the data link and notified to receiver thread when message arrives. State message is, meanwhile, uses the data link as well as event message and is not notified to receiver thread. In our model, we uses state message for memory reflection, event message for communication with remote reflective memory server. At the last message type is signal which is carried on the event link and has almost the same features of UNIX signal. To manipulate(starts, stops for example) Data Push Agent thread in our model, we use signal.

4 Experiment and Result

In this section, we evaluate data arrival jitter of our communication model. For this evaluation, we assumed the case that client requests as Reflection period:50msec, Memory-to-Memory Delay:51msec, Data size:256 byte, Distance between server and client:1 hop. Figure 4 shows data arrival time

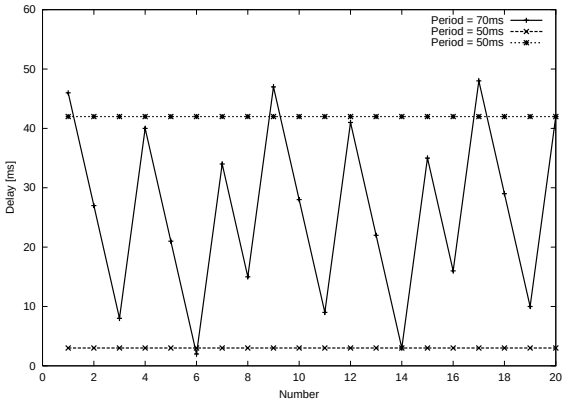

Figure 4: The result of arrival delay

5 Concluding Remarks

In this paper, we proposed data dissemination middleware on channel-based reflective memory model with proxy extension. With this properties, we can alleviate the access concentricity from clients with stringent QoS requirement.

Acknowledgements

This study was performed through Special Coordination Funds of the Ministry of Education, Culture, Sports, Science and Technology of the Japanese Government.

References

[1] Swarup Acharya, Michael Franklin, and Stanley Zdonik. Balancing Push and Pull for Data Broadcast. In *Proceedings of the ACM SIGMOD Conference*, pp. 183–194, May 1997.

[2] Manish Bhide, Pavan Deolasee, Amol Katkar, Ankur Panchbudhe, Krithi Ramamritham, and Prashant J. Shenoy. Adaptive Push-Pull: Disseminating Dynamic Web Data. *IEEE Transactions on Computers*, Vol. 51, No. 6, pp. 652–668, June 2002.

[3] Domenico Ferrari and Dinesh C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, pp. 368–379, 1990.

[4] Michael Franklin and Stan Zdonik. Data In Your Face: Push Technology in Perspective. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 516–519, June 1998.

[5] Chia Shen and Ichiro Mizunuma. RT-CRM: Real-Time Channel-Based Reflective Memory. *IEEE Transactions on Computers*, Vol. 49, No. 11, pp. 1202–1214, November 2000.

[6] Nobuyuki Yamasaki. Design and Implementation of Responsive Processor for Parallel/Distributed Control and Its Development Environments. *Journal of Robotics and Mechatronics*, Vol. 13, No. 2, pp. 125–133, 2001.