A SOFTWARE FRAMEWORK TO CONTROL A NETWORK-CONNECTED EQUIPMENT AS A PSEUDO DEVICE

M. Ishii, T. Fukui, T. Masuda, T. Ohata
SPring-8, Hyogo 679-5198, Japan

Abstract

Recently network-connected measurement instruments and controllers have increased. Today, Ethernet could be understood as a kind of a field bus for the front-end subsystems. The network-connected equipment uses the socket communication as a common software interface. The socket interface requires different handling from bus attached device access, that is, control application programmers have to understand the socket communication protocol in detail. It is inconvenient for them to develop the software efficiently. We developed a software framework, Device Masquerade, which handles a network-connected device as a pseudo device. The socket interface can be controlled through common equipment access libraries like bus-attached devices. The Device Masquerade consists of three parts; 1) API functions to access pseudo device, 2) a communication client software to access the server running on the remote equipment, 3) a pseudo device driver to interconnect between application programs linking the API library and the communication client. The pseudo device driver implements the exclusive access control. It is possible to replace the socket with other protocol. We applied the Device Masquerade for the installation of motor control units in the linac control system.

INTRODUCTION

The measurement instruments and controllers with network interface have increased in the field of accelerator and radiation research. Ethernet is recognized as a field bus for the front-end subsystem that provides high-speed communication. IP protocol is popular and implemented as an external communication interface of various control systems.

We needed a new software framework in order to introduce the network-connected equipment in the SPring-8 standard software framework [1], MADOCA, (Message And Database Oriented Control Architecture). The software framework handles network communication in the same manner as bus-attached devices. Equipment experts develop application. Understanding the details of the network protocols is inconvenient.

In the MADOCA framework, software processes control the same devices at the same time. So we need exclusive access control to the device. It seems that many measurement instruments and controllers with network connectivity are designed to be controlled via a single software process. It is important to implement exclusive access control in the MADOCA framework.

We developed a new software framework, named Device Masquerade, to control network-connected devices as pseudo devices. The Device Masquerade uses UNIX device files, so that it can be adapted not only to the MADOCA but also to other control frameworks.

This paper reports the design and performance of the new software framework, and explains about an application to the motor control units (MCU) in the linac control system [2].

DESIGN

In order to control the network-connected equipment in the MADOCA framework, our requirements were as follows.

- It can control network-connected equipment in the same handling as bus-attached devices.
- It implements the exclusive access control.
- It is a general-purpose software framework.

To meet these requirements, we designed the Device Masquerade as follows.

- Introduce a client process named Communication client (ComC) to be responsible for communication with a network-connected device. Encapsulates the differences of the network- connected devices in the ComC.
- Introduce Universal Pseudo Device (UPD) to hide the network-connected device from the application programs.
- Implement exclusive access control with lock/unlock mechanism in the UPD.
- Communicate between application programs and the ComC with the UPD by passing the control messages.

Figure 1 shows the schema of the Device Masquerade.

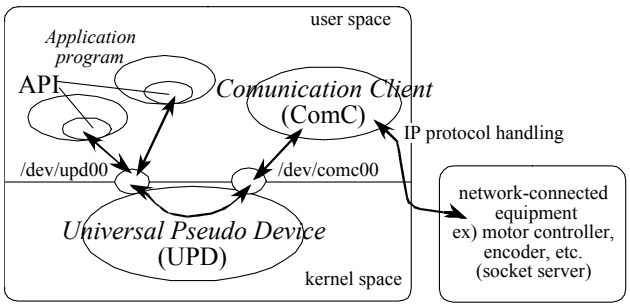


Figure 1: Schema of the Device Masquerade

IMPLEMENTATION

The UPD provides two kinds of device files. One, /dev/upd00 in Figure 1, is for the application programs. Another, /dev/comc00 in Figure 1, is for the ComC. These device files and a ComC is one set to control a network-connected device.

The ComC supports IP protocols such as TCP socket, UDP socket and FTP service. The ComC is possible to replace the IP protocol with other protocol. The ComC obtains the information such as server host name, port and a kind of protocol from start-up argument and configuration file with ASCII format, therefore, the set - up of the ComC is very easy.

The sequence of the Device Masquerade is as follow.

1) The ComC accesses to the UPD, and sleeps until receiving a message.
2) Application program sets a lock flag for exclusive access.
3) Application program sends a message to the UPD.
4) The UPD passes the message to the ComC, and the ComC wakes up.
5) The ComC communicates with a network-connected device.
6) The ComC accesses to the UPD with return message, and sleeps.
7) The UPD passes the message to application program.
8) Application program gets the return message.
9) Application program releases a lock flag.

PERFORMANCE

Test Environments

We measured the performance of the Device Masquerade in Red Hat7.2, Solaris8 and HP-RT on single CPU. Table 1 shows the specification of the platform on the performance measurements.

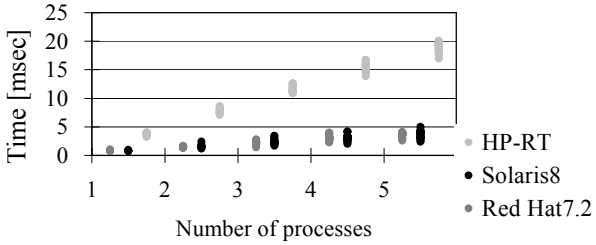
Table 1. Specification of the platform on the performance measurements for the Device Masquerade

Maker	Hewlett-Packard	Densan[3]	Dell
Product	743rt	DVE 686/50	Optiplex GX240
CPU	PA7100LC	Pentium 3	Pentium 4
Clock	64MHz	800MHz	1.8GHz
OS	HP-RT A.2.21	Solaris8	Linux (Red Hat 7.2)

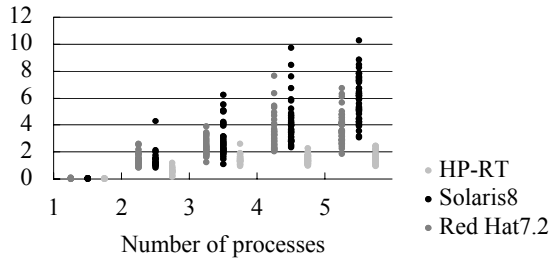
We set up the TCP socket server software on Solaris2.6 to emulate a network-connected equipment. We made the test program including API on each platform. That sequence of test program is 1) get a lock, 2) send message, 3) receive message, 4) release a lock. The message length is 30bytes. The response time is the time taking from 1) to 4). We used *gettimeofday()* standard system call for time measurement. Under multiple processes environment, we repeated this sequence 10000 times, measured average and maximum response time. And we repeated the measurement 10 times.

Response Time

Figure 2 a) shows the average response time. The total time to send and receive is constant regardless of number of processes. It is 0.7msec in Red Hat7.2 and Solaris8 and 4msec in HP-RT. The response time mostly depends on to get a lock and to release a lock. Figure 2 b) shows the maximum response time. HP-RT clearly shows the characteristic of real-time operating system (OS) and good performance. The OS scheduler of Linux and Solaris doesn't re-schedule process in wait queue.



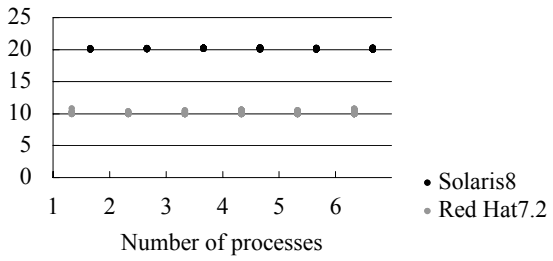
a) Average response time



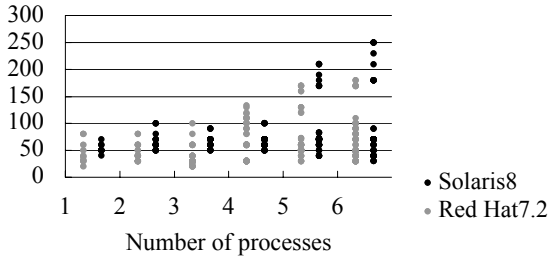
b) Maximum response time

Figure 2: Response time

The average response time is enough performance for our control system, but the maximum response time has problem under multi processes environment. The response time takes an order of seconds, it affects on control system. To solve this problem, we forced to release CPU by sleeping for one tick interval after release a lock. Figure 3 a) shows the average response time with one tick interval in API. The average response time is constant regardless of increasing processes. Figure 3 b) shows the maximum response time with one tick interval in API. The maximum response time is improved remarkably.



a) Average response time

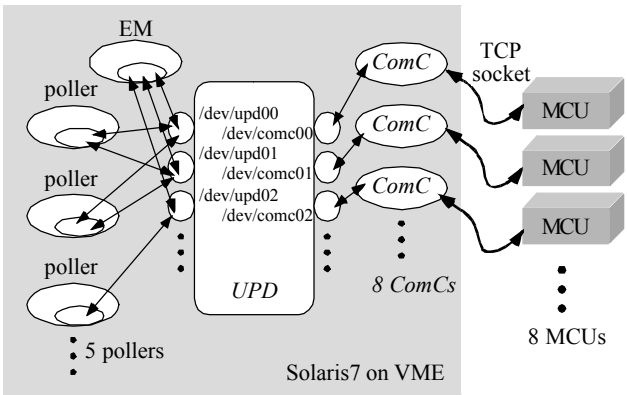


b) Maximum response time

Figure 3: Response time with one tick interval in API: one tick is 10msec.

APPLICATIONS

In the summer of 2003, we applied the Device Masquerade to control the MCUs for linac control system. Figure 4 shows the schematic view of the MCU control using the Device Masquerade.



EM : device control software
poller : data polling software
MCU : intelligent Motor Controller Unit

Figure 4: Schematic view of the MCU control using the Device Masquerade for linac system

In linac control system, the data polling cycle is 5sec. Three applications, an EM (equipment manager, device control software) and two pollers (data polling software), access a MCU at the same time. A VME CPU controls eight MCUs. Equipment experts developed application programs without a network programming.

SUMMARY

We developed the software framework, the Device Masquerade, to control a network-connected device as a pseudo device. We applied the Device Masquerade to control the MCUs for linac control system. The Device Masquerade hides the network-connected devices from the application programs. The Device Masquerade implements exclusive access control for multiple accesses environment. The Device Masquerade is possible to adapt not only to the MADOCA but also to other control frameworks. Our measurements showed poor performance under the non real-time OS. To solve this problem, we forced to release CPU in API. The performance is improved remarkably.

REFERENCES

[1] R. Tanaka *et al.,* "Control System of the Spring-8 Storage Ring", ICALEPCS'95, Chicago, USA, (1995) p.201.
[2] T. Masuda *et al.,* "Upgrade of Spring-8 Linac Control by Re-engineering the VME systems for Maximizing Availability", ICALEPCS'03, Gyeongju, Korea, (2003) in these proceedings.
[3] Densan Inc.: www.densan.co.jp