REMOTE EQUIPMENT AS DISTRIBUTED REMOTE OBJECTS USING JAVA REMOTE METHOD INVOCATION FOR THE 1.8GEV TSRF SYNCHROTRON RADIATION SOURCE

Noriichi KANAYA, Yutaka TAHARA, Shoji SUZUKI*) and Shigeru SATO*)
Department of Electrical and Electronics, Faculty of Engineering, University of Ibaraki,
Hitachi, Ibaraki, 316-8511, Japan
*) Department of Physics, Graduate School of Science, Tohoku University,
Sendai, 980-8578, Japan

Abstract

High-energy accelerator control systems are usually comprised of distributed computers and equipment such as analogue-digital converters, digital input/output ports and VME CPUs. Remote equipment interface has been implemented using Java Remote Method Invocation (RMI) running under the distributed platforms running under WindowsNT/2000 and Linux on the network. By taking advantage of Java RMI's capability, many remote accesses are carried out easily without paying a lot of efforts for remote communication. Experience with Java Remote Method Invocation for accessing remote equipment/devices as the distributed remote objects is discussed.

1 INTRODUCTION

TSRF (Tohoku-university Synchrotron Radiation Source Facility) is a new third generation synchrotron radiation source that is currently proposed at Tohoku University Japan[1] [2]. TSRF is planned to be constructed at the site of Laboratory of Nuclear Science, Tohoku University, where a 300MeV-Linac and 1.2GeV Stretcher Booster Ring are currently in operation for nuclear physics experiments [3]. By taking advantage of the existing facility, TSRF employs the Stretcher Booster Ring as the injector for the TSRF storage ring. This can greatly reduce construction cost for the TSRF generation synchrotron radiation source.

TSRF is designed to provide VUV-SX synchrotron radiation to the experimental hall where experiments such as VUV experiments, surface physics, soft x-ray lithography, microscopy and crystal structure analysis, will be simultaneously carried out. The high-power wiggler/ undulator beam lines are simultaneously in operation, producing very intense synchrotron radiation beams. The high-power beam lines are distributed along the long circumference of the storage ring. TSRF has a 1.8GeV storage ring with a DBA(double-bend-achromat) type, third-generation storage ring with emittance of 4.9nm·rad, and a circumference of 244m. TSRF has more than ten wigglers/undulators, and thirty beamlines for Soft X-ray and VUV experiments for research.

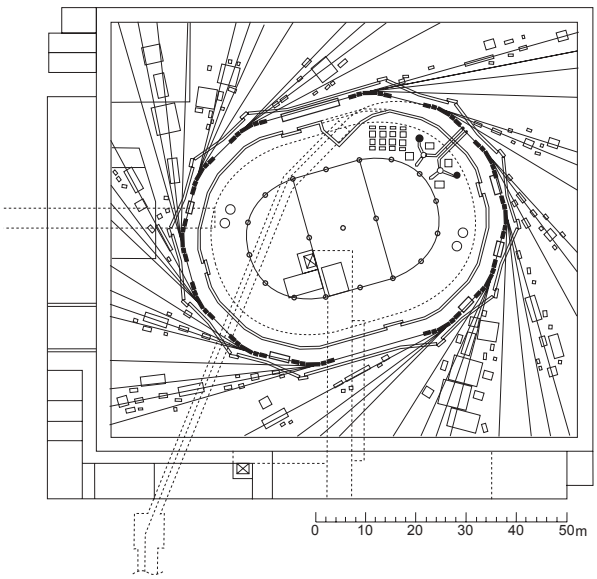

Figure 1: 1.8GeV Synchrotron Radiation Source at TSRF

The control system for the 1.8GeV TSRF synchrotron radiation source controls the storage ring, providing stable synchrotron radiation to users at the experimental hall, and improves the performance of the storage ring [4]. PCs are cost effective equipment as distributed hosts. A remote interface in software is necessary for the control

system to access accelerator components. In this paper, remote equipment as remote objects using JAVA RMI is discussed for the TSRF 1.8GeV storage ring. The remote equipment deals with PCI/VME modules on remote hosts control system. Remote equipment has been implemented using Java Remote Method Invocation (RMI) running under the distributed computers on the network. Remote accesses are carried out easily without paying a lot of efforts for remote communication.

2 SYSTEM CONFIGURATION

The remote equipment is designed with Java RMI running under the distributed PCs on the network. For time critical operation such as a high speed beam feedback, it is implemented in another language in order to avoid any time delay caused by garbage collection.

Figure 2 shows the block diagram of the remote equipment. Java programs run on the Virtual Machine (VM) that provides homogeneous environment on different platforms independent of their operating systems and hardware architecture. Although the VM runs under any operating systems such as Windows, Linux and Solaris, the VM has no direct interface to physical devices which are tightly implemented upon specific operating systems. Thus those physical devices are not accessible to control applications. The interfaces to physical accelerator components such as digital I/Os, ADCs and DACs connected to magnets and beam position monitors, are platform-dependent upon each specific operating system. A thin definition interface for PCI modules was written in C for the Java Native Interface (JNI) [5] through which Java classes can access the physical devices.

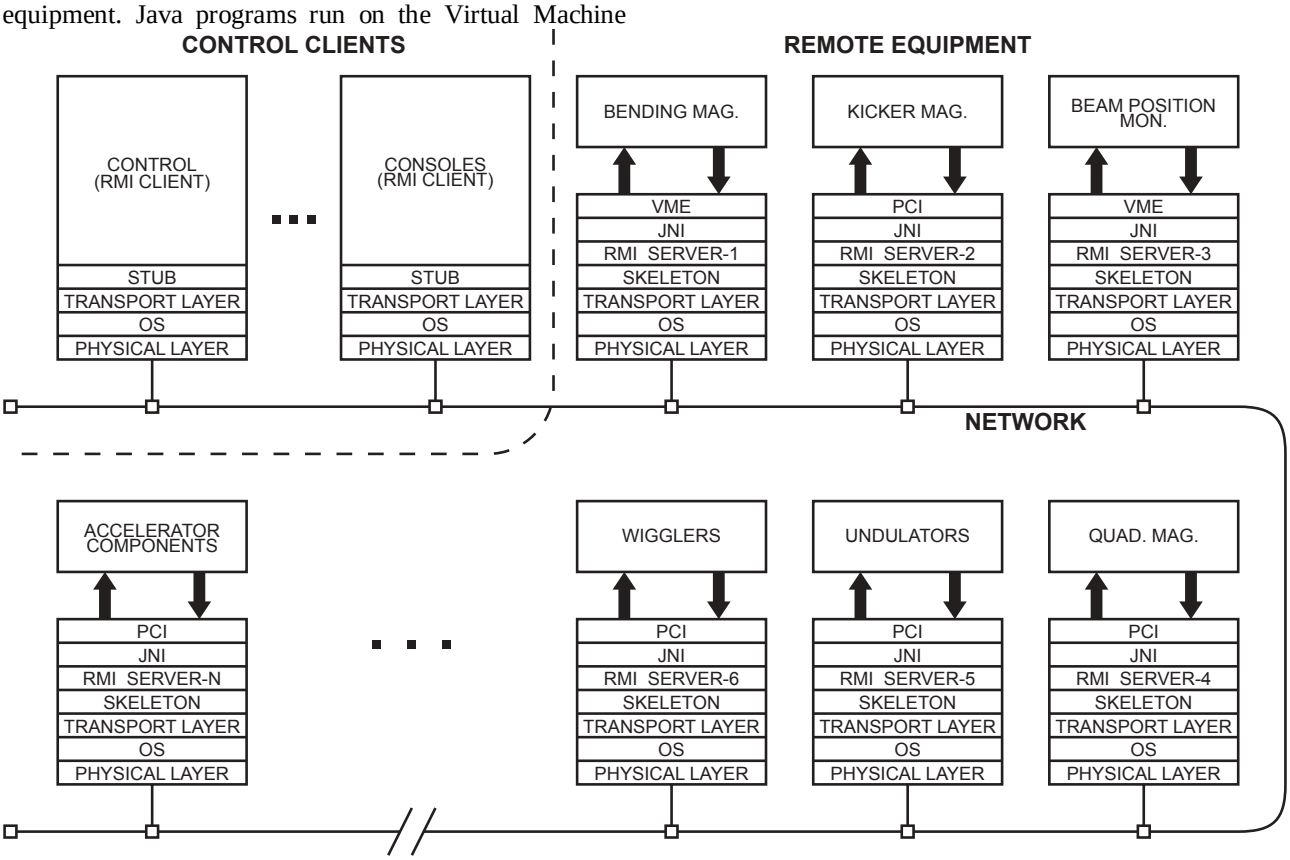

Figure 2: Remote Equipment using RMI for the 1.8GeV Synchrotron Radiation Source at TSRF

In the RPC client-sever model, it is complicated to pass and return arguments, using the XDR filters, among different platforms since the data structure of the arguments and their length are platform-dependent. It also involves difficulties caused by the byte-order and/or Big/Little Endian architectures. These arguments have to be exchanged among clients/ servers across the network. This results in costly implementation procedures. In the RMI, however, Java provides homogeneous platform environment. For remote-equipment arguments on a server are simply passed by serializing argument-data into a byte-stream data, and then transmitted it to the client requested. Transmitting a set of control messages from a client to a server is also done in the same way. The stub is the proxy allowing control programs (clients) to invoke remote methods and carries out marshaling of their arguments. The skeleton waits calls from a client, and marshals parameters and finally passes them to the

associated methods of a server. The transport layer establishes connections and deals with data from/to remote servers.

We have implemented the remote equipment on PCs running under Windows NT/2000. PCs are also utilized as RMI clients under Windows NT/2000 for controlling applications, including consoles with a large display, and a pointing device. Man-machine interfaces, including graphic status displays providing menu driven interface are also coded in rich Java graphics class libraries. There are Linux clients on PCs implemented on exactly the same classes as those implemented on Windows. There is no need to recompile the client source code on Windows NT in order to port it to a Linux machine since poring codes is done by duplicating the original classes on Windows NT to a Linux side. It means that the exactly the same client code can reside on Linux without any modification.

There is a registry server (not shown in Fig.2) that is running as a background process, and it allows control programs (clients) to locate a remote server. During bootstrap, a server asks the registry server to register or bind its name (in the URL convention) in order to make the methods of the server available to all control programs. This name is queried by a client to locate the server. Once the control program locates where the remote equipment and its related methods are, it can access the remote equipment. Then the control program carries out the remote method as if it were a local method to control the remote device to be concerned. Thus any accelerator equipment on the network is transparent to the control programs. For example, once a magnet control program obtains the reference of the BendingMagnetServer on a remote host computer, it can invoke the remote setCurrent(i) method: BendingMagnet.setCurrent(i).

When the client executes a remote method of a remote server, its action is delivered through the network to the skeleton of the server to be concerned. The skeleton marshals parameters to the actual method in the server. And finally the actual remote method is executed at the server side. In addition, a callback mechanism has been implemented for a PCI parallel digital I/O on Window NT. A client does not need to suspend until a remote method completed at the server side. Upon completion of the remote method, the client is notified of it from the server and receives the result.

As shown in Fig. 2, PCI/VME modules are connected to the bending magnets, quadruple magnets, beam position monitors, wigglers, undulators and other accelerator components. PCs are connected to a 100-Mbps network during implementation and test phases. During commissioning phase for the TSRF synchrotron radiation source, the remote equipment will be ported and implemented for a FDDI (Fiber Distributed Data Interface) with token-passing, dual-ring network using a fiber-optic link suitable for exchanging control messages. The remote equipment is also employed for the remote data acquisition system for the beamlines [6].

3 CONCLUSION

The remote equipment as remote objects using JAVA RMI is discussed for the TSRF 1.8GeV storage ring. The remote equipment deals with PCI modules on remote Linux hosts control system. Porting the remote equipment servers to Linux server on a Sparc chip is now under way.

4 ACKNOWLEDGEMENT

The authors wish to express their gratitude to Mr.T.Watahiki, University of Ibaraki for his valuable suggestions, and to N.Kobayashi, Y.Uesugi and K.Murase for their assistance.

5 REFERENCES

[1] S.Suzuki, M.Katoh, S.Sato and M.Watanabe, "Design of a Storage Ring Light Source at Tohoku University," Nucl. Instrum. Meth. Vol.A467-468,pp.72-75,2001.

[2] M.Katoh, S.Sato, S.Suzuki and T.Yamakawa, "Lattice Design of the Synchrotron Radiation Source at Tohoku University," Proc. of the 5th European Particle Accelerator Conference,Spain,June,1996.

[3] M.Oyamada et al., Proc. of 10th Symp. on Accel. Sci. Tech. p463, 1995 (in Japanese).

[4] N.Kanaya, S.Suzuki and S.Sato, "Present Status of the Distributed Computer Control System for the 1.8GeV Synchrotron Radiation Source TSRF at Tohoku University," in these proceedings.

[5] *Java Native Interface Specification*, Sun Microsystems, www.javasoft.com.

[6] Y.Tahara, N.Kanaya, S.Suzuki and S.Sato, "Design of Remote Data Acquisition System for the 1.8GeV Synchrotron Radiation Beamlines using Java Jini at TSRF ," in these proceedings.