Secure UNIX socket based controlling system for high throughput protein crystallography experiments

Yurii Gaponov,* Noriyuki Igarashi, Masahiko Hiraki, Kumiko Sasajima, Naohiro Matsugaki, Mamoru Suzuki, Takashi Kosuge and Soichi Wakatsuki

Photon Factory, High Energy Accelerator Research Organization (KEK), Japan. E-mail: gaponov@post.kek.jp

A control system for high throughput protein crystallography experiments has been developed based on multilevel secure (SSL v2/v3) UNIX socket under the Linux operating system. Main features of protein crystallography experiments (purification, crystallization, loop preparation, data collecting, data processing) are dealt with by the software. All information necessary to perform protein crystallography experiments is stored (except raw X-ray data, that are stored in Network File Server) in a relational database (MySQL). The system consists of several servers and clients. TCP/IP secure UNIX sockets with four predefined behaviors ((a) listening to a request followed by a reply, (b) sending a request and waiting for a reply, (c) listening to a broadcast message, and (d) sending a broadcast message) support communications between all servers and clients allowing one to control experiments, view data, edit experimental conditions and perform data processing remotely. The usage of the interface software is well suited for developing well-organized control software with a hierarchical structure of different software units (Gaponov et al., 1998), which will pass and receive different types of information. All communication is divided into two parts: low and top levels. Large and complicated control tasks are split into several smaller ones, which can be processed by control clients independently. For communicating with experimental equipment (beamline optical elements, robots, and specialized experimental equipment, etc.), the STARS server, developed at the Photon Factory, is used (Kosuge et al., 2002). The STARS server allows any application with an open socket to be connected with any other clients that control experimental equipment. Majority of the source code is written in C/C++. GUI modules of the system were built mainly using Glade user interface builder for GTK+ and Gnome under Red Hat Linux 7.1 operating system.

Keywords: high throughput SR protein crystallography experiment, TCP/IP UNIX secure socket, database, Linux, control application.

1. Introduction

There are several different tasks one should solve during programming of control software for operating scientific experiments. In general, different types of equipment with different control-device interfaces are used in typical experimental set-ups. One of the main demands of such software is user-friendliness and Graphical User's Interface (GUI) plays a significant role in the control system. Sustaining the capabilities of facilities for protein crystallographic experiments at a synchrotron for an extended period of time demands stringent conditions: it is necessary to keep all parts of the system in reliable conditions and protected from any mistakes or errors due to software/hardware features and human error. The software architecture must be carefully considered for rapid expansion of the scope of the experiments and necessary equipment, which means that the software must be upgradeable and adaptable for new experimental features to be incorporated. Another demand for such experimental control software is the possibility to control and assist during an experiment remotely. This is of considerable importance for high throughput protein crystallography experiments, in which many participants from different institutes participate.

To satisfy the demands described above with one standalone application on one computer is quite problematic. For example, to control several tens of different equipment units makes the application large, heavy, and difficult to modify. Response time of such large application may not be so fast, because everything will be processed in one computer. It is more advantageous to split the large number of tasks to smaller ones, which is indeed a preferred choice in the field of controlling design and large-scale programming (Ohata *et al*., 1998; Pugliese *et al*., 1998). Typically, such a system is based on a local network or a data-bus with a real time operating system of the console computer and CPU board computers are spread (for example, vxWorks with VME-bus equipment). These systems tend to be expensive and quite complicated for extensive programming and modification.

An alternative approach is Internet based distributed systems. Network with communication speed of 100Mb/s is widespread now. With a 1Gb/s network extension, the performance of such network-based system is already comparable with the performance, for example, of VME-based systems. In a distributed system, many different computers running under different operating systems are integrated through the network, which makes it feasible to develop different parts of the system simultaneously by a group of system engineers and computer scientists. Since it will be of modular nature, future upgrade and modification will be manageable. Thus we chose such a network based distributed control system to develop and design a control system of the high-throughput facilities for synchrotron X-ray protein crystallography experiments (Abola *et al*., 2000).

Protein crystallography experiments are complicated in that it is a multi stage experiment. There are several key stages: over expression of a protein, purification, crystallization, harvesting and mounting the crystal in cryo-loops, data collection, data analysis and structure determination. To perform tasks in these stages in a high-throughput mode, it is necessary to integrate all the systems, responsible for the different experimental stages, into one control system with a database to store all the experimental parameters and results of the experiments.

The main goal of the current project is to develop a control system with a unified database for protein overexpression, purification and crystallization, automated harvesting and mounting of protein crystals in cryo-loops, X-ray data acquisition, data analysis and structure determination.

2. UNIX TCP/IP secure socket client/server scheme

We chose the UNIX TCP/IP socket client/server as the principal communication scheme. To make communication secure we chose OpenSSL, an open-source implementation of the SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols, which is one of the most popular protocols used in Internet programming. This scheme is reliable and well designed under all operating systems for inter-process/program asynchronous communication (Pugliese *et al*., 1998; Ohata *et al*., 1998; Sweet *et al*., 2001). It allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before any data will be transmitted between server and client. After establishing the secure connection with a server, the client is able to communicate with the server in both directions: to send and to receive information.

There are four types of software (communication interface) behaviors during sending and receiving the information (Fig. 1): (a) a Send/Receive (S/R) type of interface performs a sequence of two operations sending a request followed by waiting/expecting to receive a reply (in this case, the interface is responsible to receive the reply from the other side to which it sent the request), (b) a Receive/Send (R/S) type of interface performs a sequence of two operations receiving a requires (in a listening mode) followed by a sending a reply (the interface is responsible to send a reply to the other side from which it received the request), (c) a Send (S) type of interface only sends a request or message without waiting for /expecting a reply, (d) a Receive (R) type of interface receives a request or message (in the listening mode) without sending any replies. A combination of the first two types of interfaces allows one

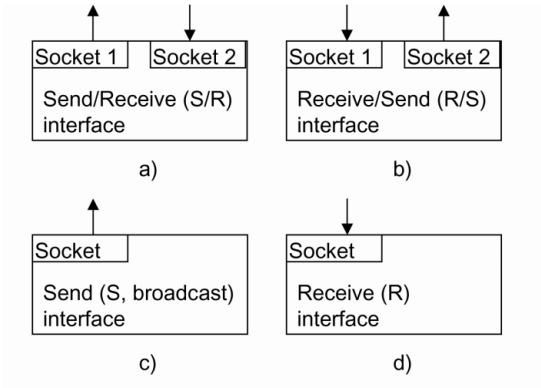

Figure 1

Different types of socket communication interfaces: a) sending a request and waiting for a reply; b) waiting for a request and sending a reply, c) sending a request or message without waiting for a reply (is used in a broadcast mode), d) receiving a request or message without sending a reply.

to build network communication with a high reliability. On the other hand, a combination of the last two types allows for network communication with a high communication speed, a preferred choice for broadcast systems. A mixture of the two combinations can be made optimized for the best performance of our control system. The usage of the interface software allows one to develop well-organized control software with a hierarchical structure of different software units (Gaponov *et al*., 1998), which will pass and receive different types of information (different levels of commands in control applications), to avoid the burden of keeping track of all the low level details of information. This hierarchical organization of servers and clients is best achieved by the use of multiple sockets for each interface, which ensures the high fidelity of communication.

In principle, all connections between different clients can be established through one communication server (Abola *et al*., 2000). It simplifies the control software for a multiple set of experimental equipment. Indeed, it is not necessary to create the control software as a server, because it can be connected to the main communication server. It will be a control client application with different types of communication interfaces mentioned above and it will wait for requests or some messages (broadcast) or it can generate its own requests or messages (broadcast). STARS, (Simple Transmission and Retrieval System) was developed at the Photon Factory, KEK, Tsukuba, Japan as such a multi socket server. It allows any clients with open sockets to be connected to STARS with predefined names. One client can be connected to STARS through several sockets. Any combinations of communications between clients connected to STARS are performed with the use of predefined names. The STARS server has a broadcast mode of operation. Therefore, all the types of communication interfaces mentioned above can be used with this server. Fig.2 shows a simple case of control for the X-ray area detector and a goniometer using the STARS server.

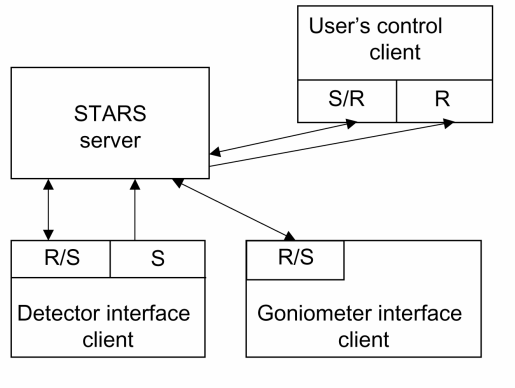

Figure 2

Simple control system with the STARS server. User's control client sends a request to Goniometer interface client to set the necessary angle and checks the completion of the operation. Then User's control client sends a request to Detector interface client to collect data, checks the completion of the operation and finally reads the data into storage media. In an emergency case, Detector interface client sends the emergency message to User's control client. R/S – Receive/Send type, S/R – Send/Receive type, S – Send type, R - Receive type of communication socket.

3. Unified database

To keep all the details of complicated protein crystallography experiments requires an extensive use of a database linked to the control system. The main goal in designing such database is to allow systematic analyses of the information such as target selection, protein overexpression, purification, crystallization, crystal harvesting and handling, data collection and analysis and structure determination. Such analyses will be essential in the high throughput protein crystallography experiments to avoid human errors and mistakes. We started with a concept of unified database, which can oversee all the aspects of the experiments and have developed a relational database and implemented it as part of the control system.

Fig.3 shows the database layout. The main object in the database is Crystal Table, which contains information on a protein crystal mounted in a cryo-loop (which in turn is described in the CrystalLoop table). Crystal loops are stored in a loop tray (LoopTray table). Protein crystals are harvested from crystallization drops (CrystallizationDrop table), which are stored in a crystallization tray (DropTray table). A protein drop is prepared from a purification batch (PurificationBatch table), which is part of a large set of experiments defined in a protein project (Project table). Protein crystals are grouped into sets of protein crystals (SetOfCrystals table) according to scientific task or methodology. The details of X-ray experiments are stored in experimental conditions (ExperimentCondition table), which are associated with experimental schedule (ExperimentalSchedule table). Concerning X-ray data analysis, the details of data indexing and integrating, and data reduction procedures are stored in indexing and integrating conditions and reduction conditions (IndexingAndIntegratingCondition and ReductionCondition tables respectively), which are associated with indexing and integrating schedule and reduction schedule (IndexingAndIntegratingSchedule and ReductionSchedule tables respectively).

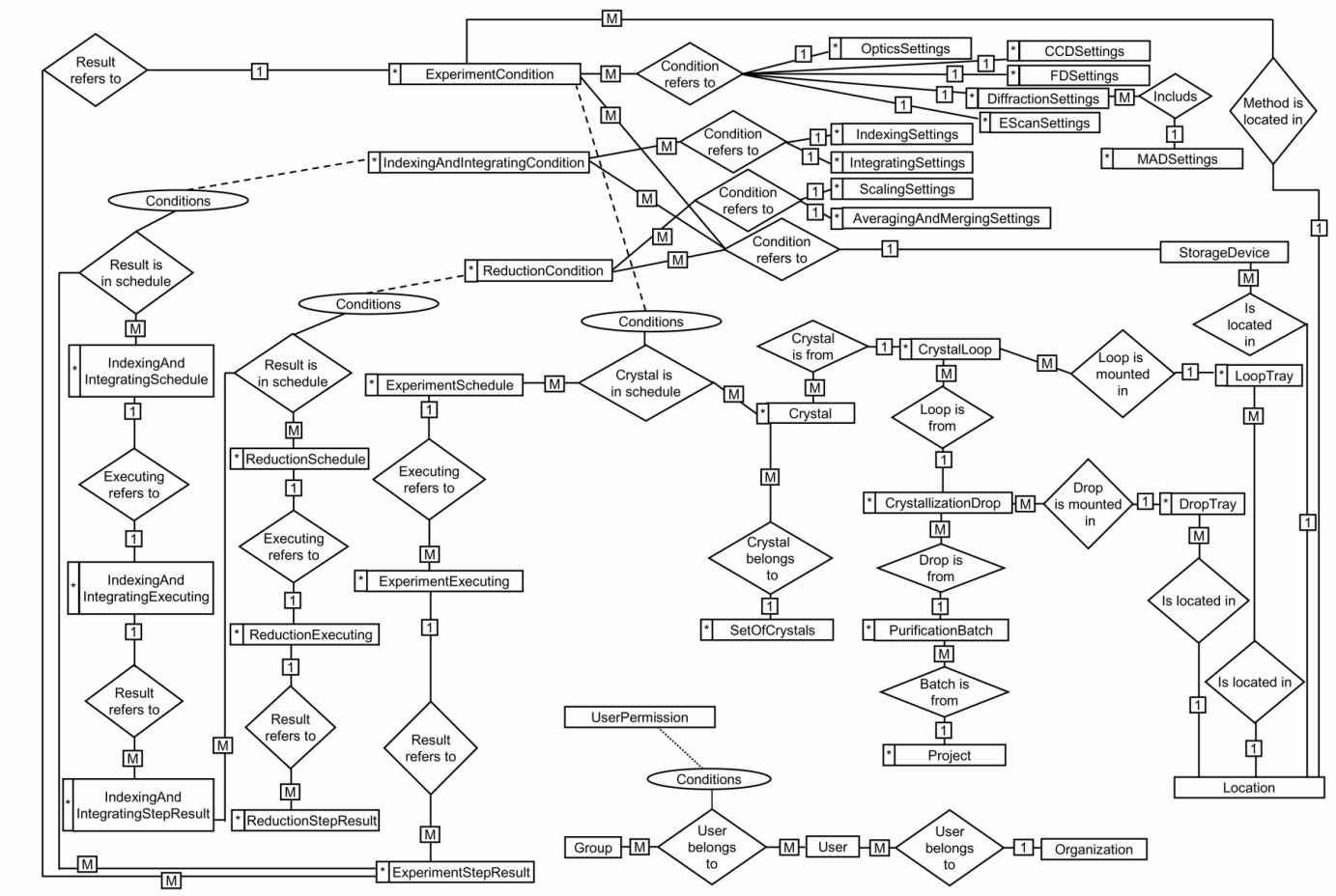

Figure 3

Relational database layout. The symbol in the square box signifies a type of relation: "1-" one record in the table is related with; "M –" many records in the table are related with. In this database there are all types of relations: 'one' to 'one', 'one' to 'many', 'many' to 'many'. An asterisk * indicates a table, whose records belong to different users.

4. Two-level control system

A control system for complicated multi-stage protein crystallography experiments needs to handle an extremely large number of tasks in sequence and simultaneously. In such a case, it is advantageous to divide the tasks to logically connected smaller ones. These smaller tasks can be made responsible for a subset of functions in the control system or a part of equipment control. In the chosen client/server model, this means to prepare client software for a set of logically connected equipment, for example, a beam stopper, a fluorescence detector, and a collimator, that all need to move in a synchronized way. The client server will be connected to STARS, thus it is possible to communicate with other control clients responsible for other subsets of equipment. This way of distributing multiple tasks to several control clients simplifies the control system enormously because it is not necessary to control all the details of the equipments.

Fig.4 shows the overall scheme of the control system. There are two levels of control. On the top level, the main server module, PCCServer, allows user's client applications to be connected in a multi-user mode. This server has a socket connection with the MySQL database, which stores all the top-level information. The server is able to execute applications for indexing, integration and scaling of the experimental data. On the top-level, control commands reflect the modes of operation of the experimental equipment, for example, data collection, direct beam measurement, beam stopper alignment, etc.

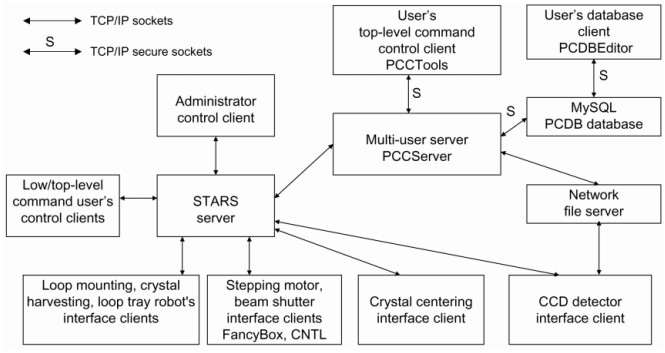

Figure 4

The overall scheme of the multi-level UNIX socket based controlling system.

On the low-level all units of experimental equipment are grouped into several sets, for example, (1) optical elements, (2) goniometer including the one-circle goniometer with an XYZ stage for crystal alignment, (3) the area of the goniometer-head (which we call FancyBox) including a collimator, fluorescent and X-ray detectors and a beam stopper, (4) the area detector control and data

X-ray data collection, and (5) manipulation/crystallization robots. An interface control clients connected to STARS through a predefined communication interface are responsible for the control of each set of equipment. Commands from the top-level control software are transformed into several ones for different sets of experimental equipment. For example, the FancyBox interface control client has several communication interfaces: (1) R/S-type of interface to receive commands mainly from PCCServer (2) S/R-type to communicate with the security control client, and (3) S-type to send out information such as the status of the equipment or an emergency case.

5. Operating system and programming environment

The main operating system is Linux RedHat 7.1. The multi-user server, user's control client and database client, PCCServer, PCCTools and PCDBEditor all operate under the Linux.

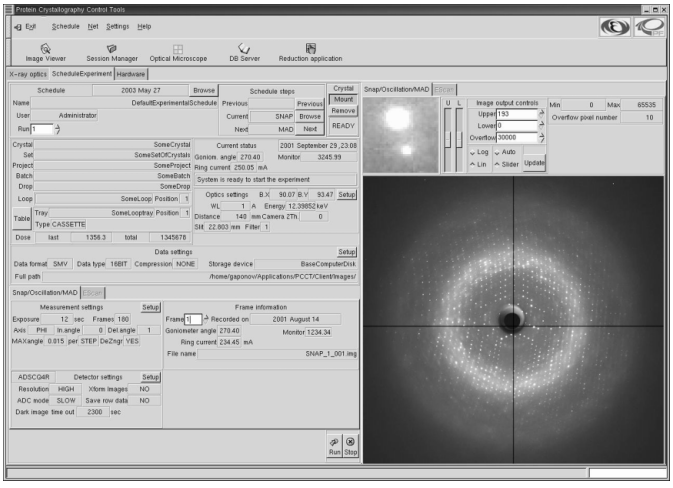

Figure 5

The main window of the user's control client, PCCTools.

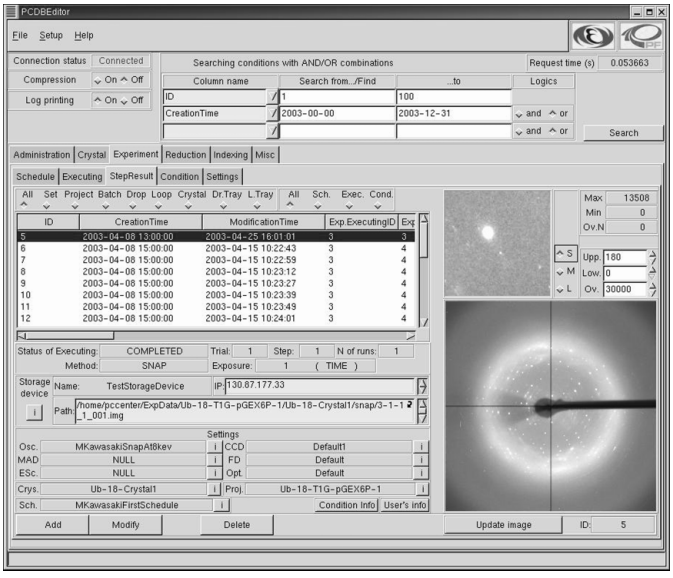

Figure 6

The main window of the user's database client, PCDBEditor.

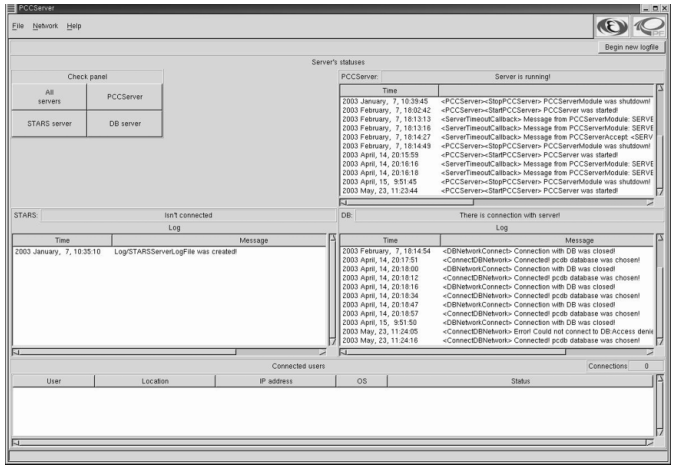

Figure 7

The main window of the interface control client, PCCServer.

The operating systems for other interface control clients are not fixed so far– it may be a combination of in-house development and third-party software. For example, interface control client to operate with FancyBox is realized under Linux, user's low-/top-level client CNTL (to operate with different stepping motors, beam shutter) – under Windows 2000. MySQL database is operating under Linux.

Majority of the source code is written in C/C++ (GNU C/C++ compiler v.2.97). GUI modules of the system were built mainly using Glade user interface builder for GTK+ and Gnome under Red Hat Linux 7.1 operating system. STARS server is written in Perl. Fig.5, 6, and 7 represent the main windows of the control clients PCCTools, PCDBEditor, and server PCCServer.

This work was supported in part by Grants-in Aid for Scientific research from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan, Special Coordination Funds for Promoting Science and Technology, and Protein 3000 Project of the MEXT.

References

Abola, E., Kuhn, P., Earnest, T. & Stevens, R.C. (2000). Nature. Struct. Biol., **7**, 973-977.

Gaponov, Yu. A., Ito, K. & Amemiya, Y. (1998). *J. Synchrotron Rad*, **5**, 593-595.

Kosuge, T., Saito, Y., Nigorikawa, K., Katagiri, H., Shirakawa, A., Nakajima, H., Ito, K., Kishiro, J. & Kurokawa, S. (2002). Oral talk on 4th International Workshop on Personal Computers and Particle Accelerator Controls, 14-17 October, 2002, Frascati (RM), Italy, http://www.lnf.infn.it/conference/pcapac2002/TALK/WE-03/WE-03_talk.pdf.

Ohata, T., Konishi, H., Kimura, H., Furukawa, Y., Tamasaki, K., Nakatani, T., Tanabe, T., Matsumoto, N., Ishi, M. & Ishikawa, T. (1998). *J. Synchrotron Rad.* **5**, 590-592.

Pugliese, R., Gregoratti, L., Krempska, R., Bille, F., Krempasky, J., Marsi, M. & Abrami, A. (1998). *J. Synchrotron Rad*, **5**, 587-589.

Sweet, R. M., Skinner, J.M. & Cowan, M. (2001). Synchrotron Radiation News, **3**(14), 5-11.