RECENT PROGRESS OF STARS

Takashi Kosuge, Yuuki Saito
High Energy Accelerator Research Organization (KEK)

Abstract

STARS (Simple Transmission and Retrieval System)[1], originally developed as an interface program of COACK (Component Oriented Advanced Control Kernel)[2-5] for non-Windows systems, is effective for various control systems. At present, STARS is installed in the beamline control system at the KEK Photon Factory and is reported to be useful by the beamline staff. We are still developing various types of client programs for STARS.

Recently, we have found that STARS is highly effective for developing the next version of COACK.

Here we describe the recent status and progress of STARS and discuss its availability as the core architecture of the COACK system.

OVERVIEW OF STARS

STARS is a message transferring software for small-scale control systems with TCP/IP sockets, which works on various types of operating systems. It was originally developed as an interface program of COACK for non-Windows systems.

STARS consists of client programs (STARS clients) and a server program (STARS server), and each client is connected to the server via a TCP/IP socket. STARS users can upgrade the system by writing client programs, and STARS clients are able to participate in the system at any time without system stoppage.

Node Name and Message Transfer

STARS clients and STARS servers handle only text-based messages. Figure 1 shows an example of message transfer on STARS.

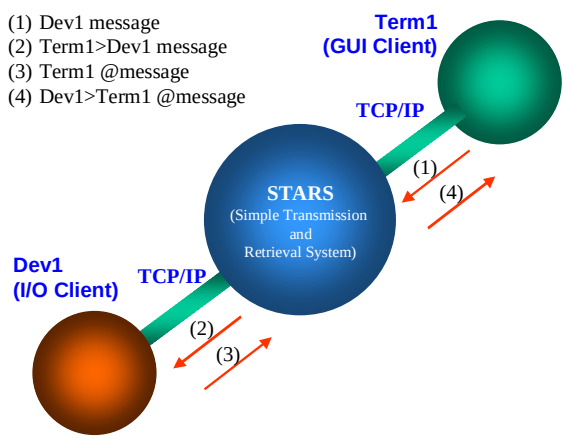

Figure 1: Message transfer on STARS .

Each client program has its own unique node name on STARS and sends messages to the server with the destination node name. In the above example, a client named Term1 sends a message (1) to the server with the destination client name “Dev1”. The server then sends the message to the applicable client named Dev1 (2).

STARS has the following simple rules for messages:

- A message that starts with “@” (e.g., @message) is a reply to a command.
- A message that starts with “_” (e.g., _message) is an event.
- Any other type of message is a command.

Dev1 receives a command from the server in the example. It must send a reply message to the server with the destination node name “Term1” (3) in this case. Finally, the server delivers the message to Term1 (4).

Event Delivery Function

STARS has an event delivery function. Figure 2 shows an example of the message delivery process. An event delivery request is registered by sending a “flgon” command to the STARS server. “System” is used as the destination node name of the STARS server. In this example, Term1 sends an event delivery request related to Dev1 to the server (1). The server then returns a reply message to Term1 indicating that the request has been registered (2).

After the request has been accepted by the server, if Dev1 sends an event message with the destination node name “System” (3) to the server, the request is delivered to Term1 (4).

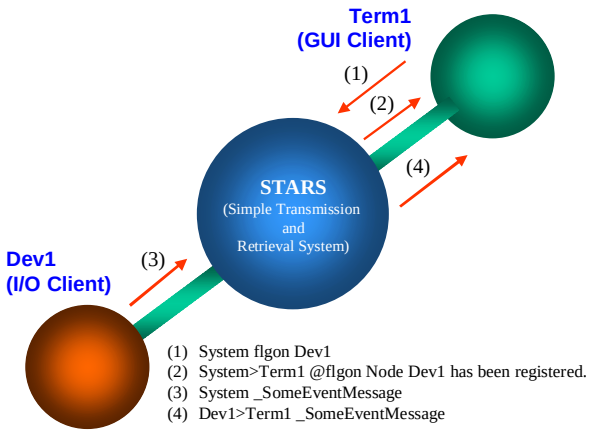

Figure 2: Event delivery function.

Certification on STARS

STARS has a simple certification procedure in three steps as follows:

- Host name certification
- Node name and keyword certification
- Node name and host name certification (option)

Figure 3 shows the certification procedure.

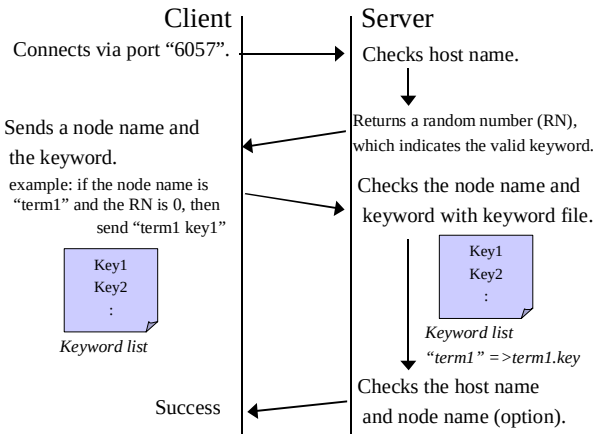

Figure 3: Node name and keyword certification.

When a client connects to the STARS server via port 6057, the server first checks the host name of the client machine. If the host name is not on the list of the server, the connection is immediately cut by the server. If the host name exists, the server returns a random number. This number indicates a valid keyword, and the client must next send its node name and a corresponding keyword with the number. For example, if the node name of the client is "term1" and the server returns "0", the client must send "term1 key1" to the server. The server then checks the node name and keyword against the keyword file. Finally, the server checks the host name and node name (if necessary), and if every procedure has been performed correctly a connection is established.

Hierarchical Node Name

A hierarchically structured system can be developed with the hierarchical node name of STARS.

A period (".") is used for the separator and the STARS server uses the first part of the node name as the destination. For example, if the destination node is specified as "Br1.Dev1" or "Br1.Dev2", both messages are delivered to Br1 (Figure 4).

A client program like Br1 must be able to handle hierarchical node names in a real hierarchically structured system. This type of client program is called a bridge.

Access to Virtual Machine on COACK

A bridge client that can handle hierarchical node names was developed as a COACK interface. COACK is able to handle a virtual image of the hardware (accelerators, beamlines, etc.), which is called the COACK virtual machine and has a hierarchical structure. The hierarchical node names of STARS can be accommodated by the COACK virtual machine.

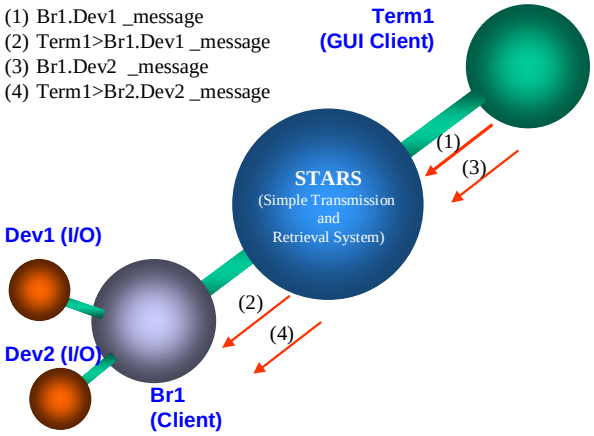

Figure 4: Hierarchical namespace.

APPLICATION OF STARS

Recently, STARS has been operating stably in BL-6A, NW-12, BL-5, BL-16A, the beamline interlock system, the access control system for experimental halls, and the monitoring system for beam transports. Next it will be installed in BL-1A, BL-1B, BL-14A, BL-20A, BL-28, and NW-14.

The control system for BL-16A, which operates with STARS, is described below as an example.

BL-16A

BL-16A at the Photon Factory is an X-ray beamline of the synchrotron light source. It has two experimental stations: A1 and A2. Optical devices of the beamline such as the monochromator and mirrors are controlled from these two experimental stations, but they use different control systems (A1: application software with Visual Basic and LabVIEW; A2: SPEC). We installed STARS in the beamline and prepared an interface for LabVIEW and SPEC.

Hardware and Network

The composition of the hardware and network of BL-16A is shown in Figure 5. A PC for the users' terminal of A1 (Windows XP), a PC for the STARS server (FreeBSD), and an Ethernet/RS-232-C converter are installed inside a firewall. A PC for A2 (Linux) is connected to the KEK-LAN directly.

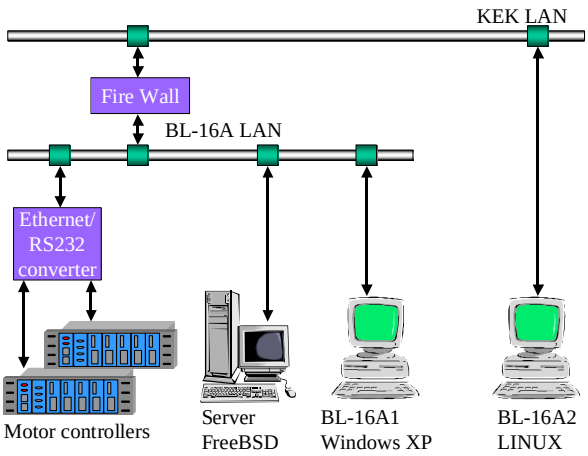

Figure 5: BL-16A hardware and network.

Software

Figure 6 shows the layout of the STARS clients and server of BL-16A. A STARS server and motor driver clients (written in Perl) are running on the STARS server PC (FreeBSD). STARS clients (written in Visual Basic) and LabVIEW are running on the users' terminal PC and are connected to the STARS server directly. SPEC (a UNIX-based software package for control and data acquisition), which runs on the PC for A2, is connected to the server through a SPEC interface program of STARS.

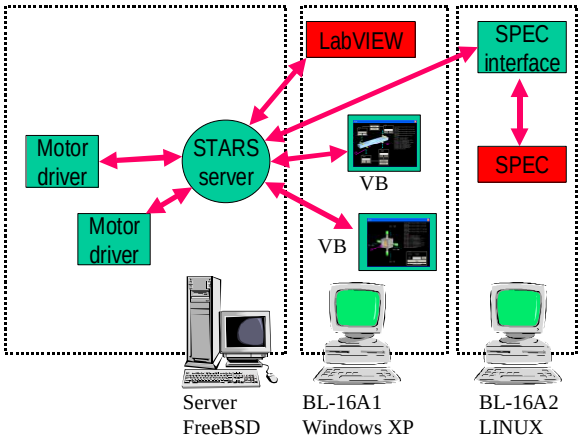

Figure 6: Layout of STARS clients and server at BL-16A.

STARS CLIENTS AND UTILITIES

Connection of STARS is easy, and STARS users can develop an effective system by writing simple client programs. The STARS interface library and multipurpose STARS clients provide high-performance development resources for STARS users. Recently, STARS has been equipped with several library modules (a Perl interface module, C interface library, and ActiveX control for VB6) and multipurpose STARS clients (a logger, logreader, and sequencer).

Perl Interface Module

The Perl interface module of STARS provides simple connection to STARS. An example of its usage is shown below.

```
#! /usr/bin/perl

use stars;

#my node name.
$NodeName = 'sample_client';

#Hostname of STARS server.
$Server = 'localhost';

#Connect to the STARS server.
$tak = stars->new($NodeName, $Server)
  or die Could not connect Stars server";

#Send a message to a client named "Dev1".
$tak->Send("Dev1 getdata");

#Receive a message from "Dev1".
$rt = $tak->Read();

#Print result
print "$rt\n";

#Exit, processes of closing connection, etc. are
# performed automatically.
exit(0)
```

Sequencer

The sequencer is one of the multipurpose clients of STARS. The sequencer executes scripts upon request from other STARS clients.

If a client sends a command which requires the execution of a script, the sequencer reads the corresponding script file and executes it. The sequencer accesses I/O clients, if necessary. Finally, the sequencer returns the result to the client. Figure 7 shows an image of message transfer between STARS clients and a sequencer. Script commands are definable by users and script files can have a hierarchical structure.

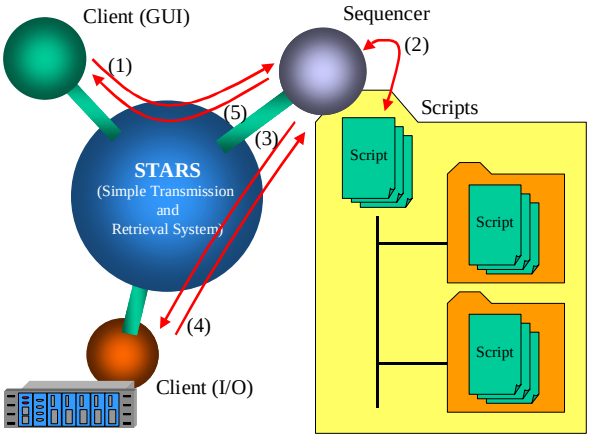

Figure 7: Sequencer.

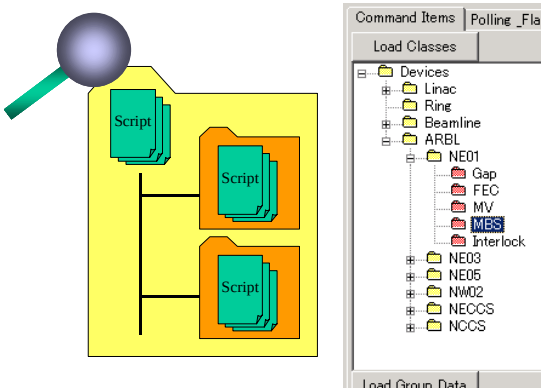

Figure 8: Hierarchical structure of sequencer and COACK virtual machine.

FEEDBACK TO COACK

The hierarchical structure of script commands on the sequencer can be accommodated by the COACK virtual machine, which has a hierarchical structure (Figure 8), and various functions can be added by editing script files. Then the sequencer is able to have one of the functions of COACK (the COACK virtual machine).

CONCLUSION

Recently, STARS has been running stably on various control systems, and the development and maintenance of its clients and server are continuing. STARS is effective for the development of the next version of COACK.

REFERENCES

[1] T. Kosuge, et al., “COACK Multi-Server System with STARS”, PCaPAC2002, Frascati, 2002.

[2] T. Kosuge, et al., “COACK Application for the Beamline Interlock System at the Photon Factory”, PCaPAC2000, Hamburg, 2000.

[3] I. Abe, et al., “COACK-II Project on Accelerator Control Kernel Development”, ICALEPCS '99, Trieste, 1999.

[4] T. Kosuge, et al., “Application Server and Pushing Technology on COACK-II”, ICALEPCS '99, Trieste, 1999.

[5] I. Abe, et al., “Recent status on COACK project”, PCaPAC2000, Hamburg, 2000.