VERSATILE NETWORK STREAM CAPTURE TOOL USING JAVA FOR HIGH ENERGY ACCELERATOR CONTROL SYSTEMS

Shintaro Mori, Akihiko Shikanai, Takuya Saito, Hideyuki Hayami, Noriichi Kanaya,
Graduate School of Science and Engineering,
University of Ibaraki, Hitachi, Ibaraki, 316-8511, Japan

Abstract

A network stream capture tool has been developed to monitor and capture control data and information encapsulated in network stream for distributed control systems for high energy accelerators. The tool allows capturing data streams between specific computers, and dumping the stream data into a file. The data can be browsed using graphic user interface (GUI), either in binary, ASCII, and hexadecimal format to analyze and debug communication protocol employed among computers for the control systems. The tool has been implemented using Java, and thus ported to various platforms, including Linux, Solaris and Windows, providing versatile functionality necessary for multi-computer control systems. This paper describes design and implementation of the network stream capture tool in detail.

INTRODUCTION

Control systems for high energy accelerators are mostly based on client/server model employing many computers and network in order to access components of the accelerators [1],[2],[3]. Obviously a high energy accelerator has a large number of components that are remotely distributed along the circular vacuum chamber of the doughnut. The clients and servers of the control system are usually running under not the same operating systems or platforms. Without monitoring the network stream between the clients and servers on the network it tends to be difficult to debug such clients/remote-servers along the large accelerator. Thus, a packet capturing tool is necessary to develop the distributed control system. Furthermore, unlike a freely available tool for a specific platform, we need the tool that can operate under multi-platforms, such as Linux (with different distributions), Solaris and Windows (for console). There is also the case that embedded interface systems, that provide interconnection capabilities between computers and accelerator components, have no operating system but UDP stacks [4].

To achieve this goal, a versatile network stream capture tool has been implemented for the purpose of debugging the client/server systems, i.e., control systems for accelerators. The tool must have the following principal functions:(1)capturing packet stream, (2)viewing filter capability, (3) data-dump, (4) GUI, and (5) it has to be able to run under different platforms. The design and implementations of the tool is described.

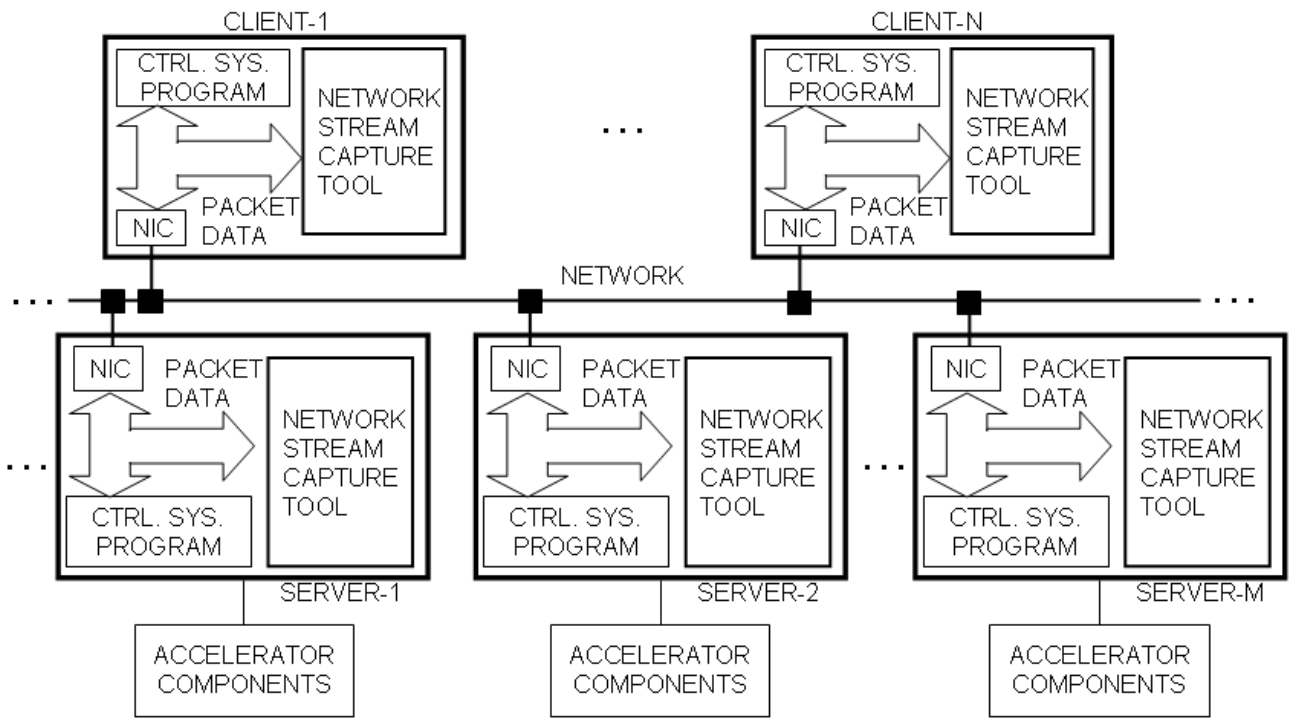

Figure 1: Network stream capture tools tapping the incoming/outgoing packet stream through NICs on various platforms.

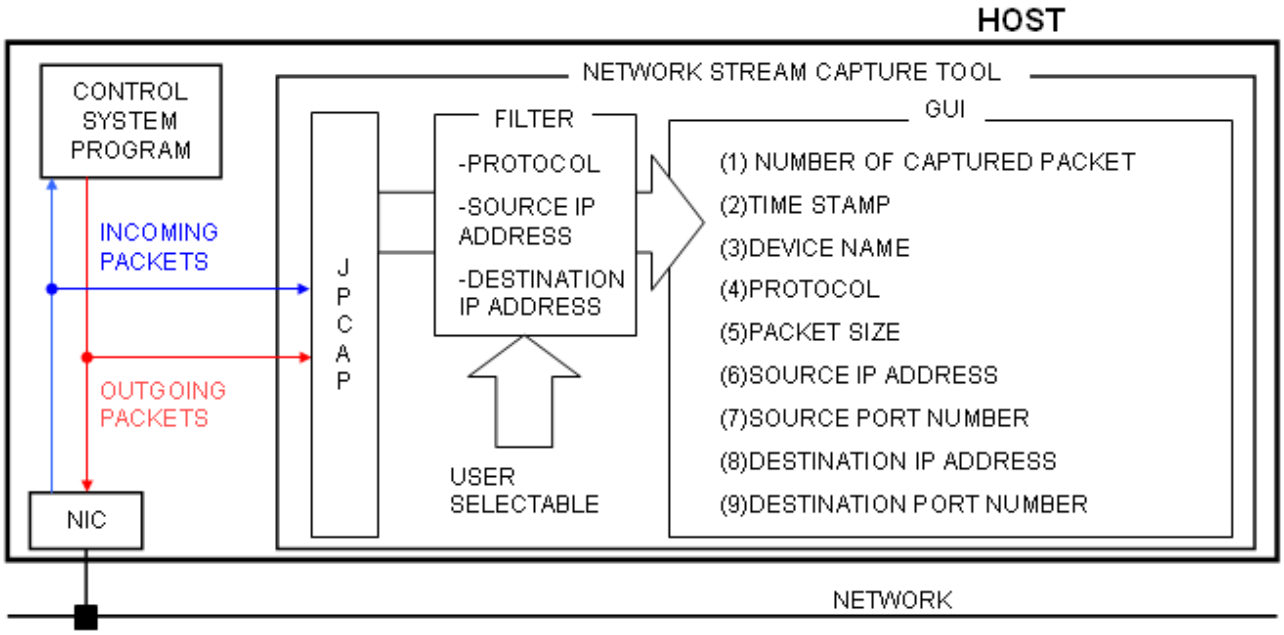

Figure 2: Block diagram of the network stream capture tool in the host computer.

CONFIGURATION

Figure 1 shows the simplified diagram of the network stream capture tool. Since Java has no direct API to access the physical NIC(s), Jpcap library, which has JNI(Java Native Interface) layer, is employed[5]. Jpcap is a primitive protocol parser that allows application program to fetch data stream i.e., packet, from the NIC and decomposes its header into the control fields associated with the packet. With the aid of Jpcap, the network stream capture tool (NSCT) has been implemented in Java.

It allows operating under different platforms for accelerator control systems. NSCT can 'tap' and capture incoming and outgoing packet-stream through the NIC of the computer as shown in Fig.2. Unlike the network analyzer there is no need to capture packet stream outside the computers but through the NIC since we focus on only information in terms of data flow and packets between the client and the server. Thus there must be an NSCT on each client (or server) side to acquire the data stream. This greatly reduces overhead of the tool. NSCT has been implemented by integrating the following functionality:

(1)Capturing packet stream and viewing-filter capability,

NSCT extracts the following information specified by user and depicts on the display (Fig.3) which shows protocols, source/destination addresses, and port ids. By analyzing the control fields of the packet, the contents of the packets are displayed in accordance with combination of the following protocols: TCP, UDP, ICMP and ARP. By default, it depicts all possible protocols in packet stream. On top half of the screen, a sequence number and time-stamp both added by the tool are shown to identify the packets. The device name of NIC is provided as a vender's ID, and it always the same since the tool recognizes only one NIC at the same time.

From left to right, protocols, data-length or size in bytes of the packets are depicted. Showing the data-length of the packets gives a simple clue to debug the system when the packet is unexpected surplus data caused by bug or network failure. One can make sure that the packet is exactly designated from the source and destination IP addresses with the specified port ids just by browsing the GUI of the tool.

By default, NSCT shows the contents of all packets. However, one can chose specific source address or destination address to capture such data stream (Fig.2).

(2) Dump function,

An error may occurs in a long time interval when the control system is in operation during debugging stage. It is difficult to find out such inherent bug in erroneous communication program. The tool can dump data stream into a file for a specific duration, for hours, so that one can analyze and inspect the packets in the dump file later. The tool has a GUI to provide similar screen after analyzing the contents of the dump file.

(3) And user friendly GUI with remote-viewing function.

As shown in Figure 3, the GUI of the tool depicts the data stream captured. Binary data stream captured from a NIC of server is depicted on the left and its ASCII printable characters on the right while unprintable characters are indicated by period. It allows identifying what command characters were sent to the server from the client.

For front-end interfaces, such as reconfigurable embedded interface systems, they usually have no NIC but stacks for UDP or TCP/IP communication [4]. The tool has a remote-viewing function useful

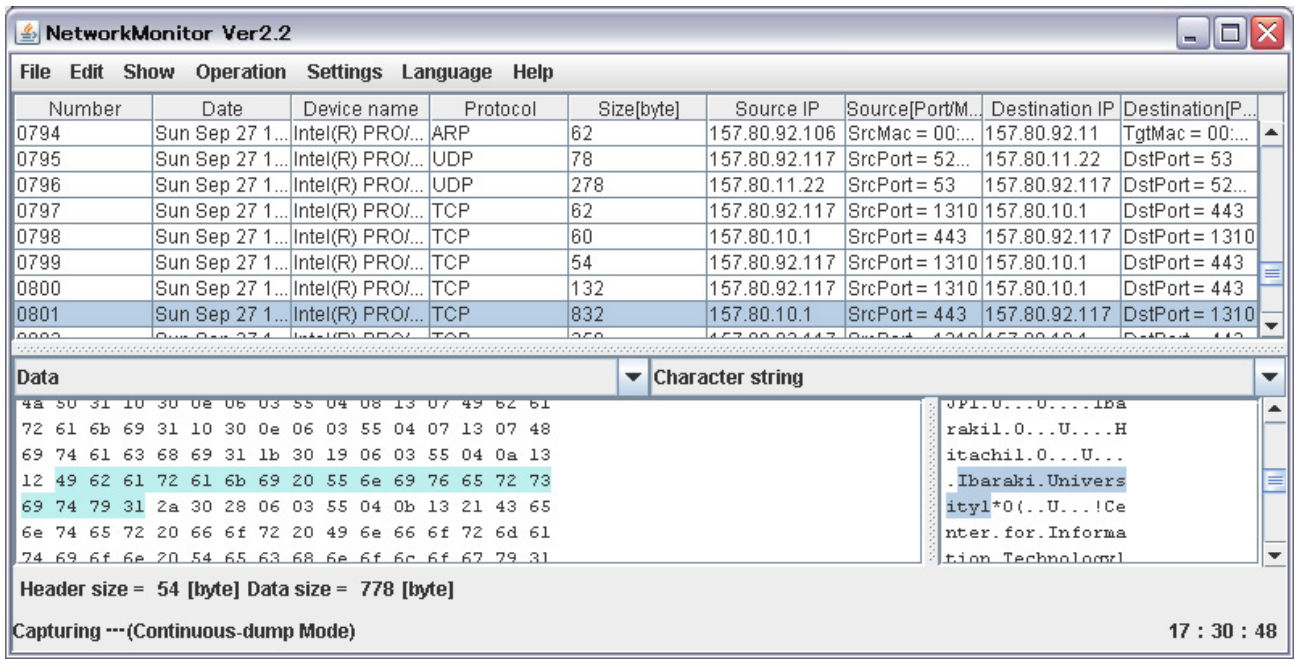

Figure 3: GUI of the network stream capture tool.

for debugging purpose. Once the tool running on the server side captures and dumps data streams, into the file, between the server and the embedded interface systems on the network. The tool running on another host has a remote-viewing function that can fetch the file containing the dumped stream data on the server. This is done by accessing that server through the network. Then the tool running at the remote host provides the analyzed data onto the screen as if the tool were residing on the embedded interface systems. This tool provides important information in developing such front-end interfaces for accelerators although the tool captures only data stream on the server side but not exact data stream received at the embedded interface system.

The tool is applicable even to network communication system where command is not implemented by combination of characters but binary as in Java remote method invocation (RMI). Using RMI, arguments are passed by serializing argument-data into a byte-stream data at a server-side, and then are transmitted to the client requested.

CONCLUSION

The network stream capture tool provides important functionality for analyzing and debugging communication packets among computers for accelerator control systems. It is capable of running at cross-platform environments, allowing to show details of data stream in the visible forms among clients/servers. The tool is found to be quite useful for even debugging front-end interface systems, for accelerator components, that have no network interface controller but simple UDP stacks.

ACKNOWLEDGEMENT

The authors wish to express their gratitude to Prof. Emeritus Shigeru Sato, and Prof. Shoji Suzuki, Tohoku University for their valuable discussions.

REFERENCES

[1] N.Kanaya, Y.Tahara, N.Kobayashi, S.Suzuki, and S.Sato, “Asynchronous Remote Event Notification Using a Distributed Object Model for Heterogeneous Remote Monitoring System and Control System at the 1.8-GeV Tohoku Synchrotron Radiation Source,” IEEE Trans. Nucl. Sci., Vo1.53, No.5, Oct., 2006.

[2] Y.Tahara, N.Kanaya, and S.Suzuki, “Design of The Remote Data Acquisition System Using Java JINI for The 1.8-GeV Synchrotron Radiation Beamlines at TSRF,” Proceedings of EPAC, France, 2002.

[3] N.Kanaya, N.Kobayashi, Y.Tahara, S.Suzuki, and S.Sato, “Distributed Control System Using a Remote Distributed Object Model for 1.8GeV Synchrotron Radiation Beamlines at TSRF,” IEEE Trans. Nucl. Sci., vol.52, No.1, Feb., 2005.

[4] M.Ariff.B.Mohtar, T.Saito, S.Mori, N.Kanaya, and K.Furukawa, “Reconfigurable Embedded Interface System for High Energy Accelerators,” in these proceedings.

[5] http://netresearch.ics.uci.edu/kfujii/jpcap/doc/ (2009.07.21).