Inter-application communication during LHD consecutive short pulse discharge experiment

Inter-Application Communication during LHD Consecutive Short Pulse Discharge Experiment

Masahiko Emoto[a], Hideya Nakanishi[a], Masanobu Yoshida[a], Masaki Ohsuna[a], Hideki Ogawa[a], Setsuo Imazu[a] , Hiroya Maeno[a], Tatsuki Ito[a], Mitsuhiro Yokota[a] , Miki Nonomura[a], and Miwa Aoyagi[a]
[a]*National Institute for Fusion Science, Toki, Japan*

LHD short pulse experiments are executed every three minutes. After the end of the discharge, the scientists must collect, analyze, visualize the last acquired data of the discharge, and prepare for the next discharge. From the beginning, the computer environment of the LHD (Large Helical Device) experiment has been built as a network distributed system, and various computers have been used for data acquisition or physical analysis. When one program is finished on one computer, that computer must send the results in order to the other computers to run programs. Smooth communication is required in order to finish all the tasks before the next discharge. To exchange the information among the applications running on the different computers, the authors have tried various methods, such as a commercial software to share the memory over the network, simple network file sharing method, IP multicast, web interfaces, and others. The purpose of this paper is to share our experiences of trial and error to build the network distributed systems for the consecutive plasma discharge experiments.

Keywords: data handling, communication, experiment sequence

1. Introduction

For the LHD experiment, more than 100 diagnostics devices are collecting the experiment data. Furthermore, because the computer system has been developed as the network distributed system, many computer systems coexist [1-2]. Most of the data is acquired by the LABCOM systems [3]. Generally, the raw signal data is referred to only by the scientists who are responsible for the diagnostics because other people do not know how to convert the signal data into physical data. Therefore, they usually refer to the physical data managed by the Analyzed Data Server [4]. Therefore, most of the experiment data can be referred to by interfaces of the LABCOM system and the Analyzed Data Server system. However, there are other systems that acquire raw signal data or device information data in their own way. The authors are often asked to register these data into the Analyzed Data Server system. In order to do this, communication between their system and the registration programs is required.

The most common operation of the LHD is the short pulse experiments. In this operation mode, plasma discharge experiments are executed every three minutes. During the short break between the discharges, they must collect data in order to analyze the previous discharge and prepare for the next discharge. In order to execute such tasks in a short amount of time, smooth communication among the computers engaged in the experiments is required. For historical reasons, various communication methods are used for the LHD experiment. In the next section, the authors discuss these methods, and discuss their advantages and disadvantages.

2. Methods of Communication

For the LHD project, various methods are used to communicate among the heterogeneous computers. Fig.1 shows the communications among the major components. When the shot number is updated, the user computer retrieves the raw signal data from the LABCOM system, and converts it into physical data referring to the experimental information in the database server. Then, the physical data is registered in the Analyzed Data Server. The physical data is stored as a comma separated text file, and it can describe multi-dimensional data. Because most of the experiment participant cannot understand the raw signal data, they usually refer to the physical data provided by the Analyzed Data Server. Therefore, in principle, all of the physical data related to the LHD project, such as time history of NBI injection power (1-D data), time history of radial distribution of electron temperature (2-D data), and others, are served by the server.

When the new data is registered in the Analyzed Data Server, an IP multicast packet, in which the shot number and the diagnostics name of the new data are recorded, is broadcasted to the LAN. Receiving this packet, the Automatic Analysis Server [5] runs the analysis programs that require the registered data as input source. Lastshot.py plots the latest data retrieved from the LABCOM system and the Analyzed Data Server. Lastshot.py also uses the IP multicast packet to know that the new data is registered, and updates the screen to draw the latest data. After the update is finished, it sends the screen image as a PostScript file to the virtual printer. Receiving the PostScript file, the virtual printer converts the PostScript file into a PDF file provided by the web server.

Table 1 categorizes the technologies used for the LHD project. The technologies can be roughly divided into two categories, HIGH level and LOW level. The HIGH level category uses OS or Application to share the contents.

Generally, HIGH level does not need programming and requires little effort to share the data. On the other hand, the LOW category must write more codes. Also, LOW can be divided into five types, socket programming, web technology, remote procedure call (RPC), network shared object, and UDP. The following subsection shows each method in detail.

.

2.1 High Level

2.1.1 File Sharing

This file sharing method is used for placing the file on the network shared folder, such as Windows file sharing or NFS, in order to share files with several computers. This is one of the easiest methods, and is often proposed by scientists. This is because modification of the existing program is small; the scientists simply add more code to write a shared file. However, this method has many disadvantages, such as 1) the number of connections is restricted by the license; 2) the file is locked by one server and others cannot read the file; 3) the completion of the file writing is not known; and 4) the ambiguity of the file format, and others reasons.

Applications in NIFS

This file sharing method is widely used for the LHD experiment because of its simplicity. The files are written in the shared folder of Windows or Samba services provided by Linux servers to share among multiple PCs. For example, the data acquired by systems other than the LABCOM system provide their data by this method.

However, there are a few systems that previously used this method, but subsequently abandoned file sharing. For example, the current shot number previously was written in a shared file. The applications that wanted to know the current shot number read the file. However, because of the poor behavior of some applications the file was often locked. Thus the other applications could not read the file. Currently, the shot number is provided by the PHP server, and IP multicast, which will be introduced below.

Another example is a virtual printer. The virtual printer is a system that provides the PDF files of the summary graph. The virtual printer received PostScript files from the client and converted the PostScript files into the PDF files. The virtual printer also used the shared file method. The virtual printer waited for the “close” system call to detect the completion of file writing, and began file conversion. However, if the size of the PostScript was large, the application did not write the file at once, but repeats “open” and “close” until the writing was completed. The virtual printer system often read the incomplete PostScript file and failed to convert. Therefore, the authors abandoned this method, and began to use CGI instead.

2.1.2 Database

Because the database server is to be used by multiple clients simultaneously, one can avoid problems that occur during file sharing, such as file locking, incomplete reading, and ambiguity of the data format. However, using a database, the program must use the database API, and must convert the original data into the database types.

Applications in NIFS

In the LHD projects, various databases are working. For example, the experimental configuration data are directly inserted into the database, and the user can search the data using the database languages. Furthermore, many services use open-source relational databases, such as PostgreSQL, MySQL, and SQLite in the background to store the various information, such as attributes of the experimental data, configurations of applications, and others. Fig.2 shows a part of the tables stored in the database for the Analyzed Data Server. The Analyzed Data Server uses PostgreSQL to store the location of the physical data and other experimental information.

2.2 Low level

2.2.1 Direct Socket Programming

IP is a basic protocol of the Internet. There are two major types of IP connections, TCP and UDP. TCP is a connection based protocol, and it is more reliable than UDP. Therefore, it is commonly used for various communications for the LHD experiments. Using socket API is the most basic way to use TCP. For example, the LABCOM system uses socket interface to transfer data between the server and the clients.

Using socket API directly we may expect good performance and flexibility, but the program becomes complex and is error prone. Therefore, other technology to facilitate the TCP program, such as Web technologies, RPC, network shared object, and others, are used for the LHD project.

2.2.2 Web technology

Generally, the communication between two computers is not symmetrical. The communications are usually done among a small number of servers and many clients. The programmers must take special care when writing server-side programs to handle many requests from clients, or they will easily cause problems such as memory leaks. Using a web server, it becomes easier to build a server program. Fig.3 shows the sequential diagram of the communication between a client program and a web server using CGI. The web server runs the CGI program each time it receives a request. Because HTTP is the most popular protocol in the Internet, one can use a well-proven server to avoid server-side problems, and one can use various products for the web server, such as load balancers and proxy servers. The disadvantages are that this method requires extra code for session management and that the performance is poor.

Applications in NIFS

For the LHD experiment, the following services are using web technologies. 1) Shot number notification service, which is the replacement of the previous service using the shared file, and it is provided by the PHP server. 2) Virtual Printer. This is the server to convert PostScript files received from the clients into PDF files. The converted files are served by the web server as well as being sent to the page printers. Fig. 4 shows the summary graph drawn by Lastshot.py displayed on the screen while discharge experiments are running, and it is sent to the virtual printer to create PDF files. 3) Experiment scheduler, daily reports, and simple data viewer, and other services are developed by Ruby on Rails, which is a web application framework written in Ruby.

2.2.3 Remote Procedure Call

In order to ease the difficulties of implementing complex client-server protocol, Remote Procedure Call (RPC) is often used. Fig. 5 shows the sequence diagram of RPC. The communication between server and client is hidden by stubs. The protocol is usually defined by IDL (Interface Description Language) file, and the IDL compiler reads the IDL file to generate stubs of source codes (Fig.6). Because complicated programming codes related to the network communications are produced automatically, the programmer's task is reduced dramatically. Generally, the IDL compiler can generate source codes in multiple languages for multiple operating systems. Thus it is easy to write communication programs for heterogeneous environments.

Applications in NIFS

For the LHD project, a Windows program called plasma.exe uses ONC RPC to configure parameters of the diagnostics devices that are managed by VxWorks. ONC RPC was developed by Sun Microsystems, and used for services of UNIX, such as NFS and NIS. However, because it is open architecture, ONC RPC can be used for Windows, too. The interface is written by RPCL (Remote Procedure Call Language). Fig. 7. shows the part of the IDL used for plasma.exe. In this file, the procedures to control diagnostics devices are defined. RPC compiler, rpcgen, converts this file to produce both server-side and client-side stubs.

Another example of RPC is Analyzed Data Server, which provides the physical data. The interface between the server and the clients is SOAP. SOAP use XML format message to exchange data by various network protocol, such as HTTP or SMTP. The interface is written in WSDL (Web Services Description Language), and the client- and server-side stubs for multiple language are generated by compiling WSDL. Analyzed Data Server uses gSOAP [6], a C++ development toolkit for SOAP, and uses HTTP to serve physical data.

2.2.4 Network Shared Object

Another approach to call functions of remote PC easily is to use network shared objects. This technology can be realized by the middleware or the specific function of the programming language to share the remote objects among the programs distributed over the network. Fig. 8 shows the diagram of remote object call. The client calls the method of a server-side real object over the network through a client-side proxy object. The advantages of this approach are that the programmer does not write the network programming and they can treat the network shared object in the same manner as the programmers treat local objects. On the other hand, the functions depend on the specific applications and the language, but it is difficult to use the function in unsupported environments.

Applications in NIFS

The LABCOM System previously used HARNESS [7] to share the information during the data acquisition. HARNESS is a middleware to enable sharing of the remote object among the object oriented programs over the network, and reduces the complicated task to do the same things without HARNESS. However, the authors decided to stop using HARNESS because of performance issues.

The other examples are language specific functions, Java RMI for Java and dRuby for Ruby. Both Java RMI and dRuby are technologies to treat the network shared object as if it were a local object. Therefore, the programs can handle network communication without the network programming. Java RMI is used to configure the data acquisition parameters from the web browser. The application is a Java Applet. Therefore, it was a natural choice to use Java RMI. dRuby is used for the Automatic Analyzed Server to exchange messages among its component programs.

2.3.5 UDP Based communication

UDP is not a reliable communication as the packets may be lost during the transmission. UDP is not reliable, but it is more efficient than TCP. Therefore, it is suitable to send unimportant data, for example, to monitor data, because it is not a serious problem if a small number of packets is lost.

Applications in NIFS

For the LHD project, the real-time graph application, which is a Java Applet, uses UDP to monitor the data during the long-pulse operation (see Fig.9) Most of the diagnostic data acquired by the batch operation, however, cannot be referred to until the acquisition is completed. This becomes a problem during the long-pulse operation because researchers cannot watch the current long pulse experiment which runs up to 1 hour.

Another example is notification of various information by IP multicast. IP multicast is a technique for sending the same data to a group of clients in a single transmission. Because the server cannot send data multiple times, the number of the clients does not affect the server load, and it is suitable for the one-to-many communications, such as media streaming. For LHD experiment, IP multicast is used to notify the experimental sequence and the new data registration to the Analyzed Data Server. These data are used to run the analysis programs synchronized with the experimental sequence, and to update the summary graph of the last registered data. For these programs, the loss of the packets is not serious. They can be manually executed later. On the other hand, for the data acquisition programs, the loss of the experimental sequence is critical because scientists can never gather the same data again, thus these programs use hardware triggers.

3. Discussion

The current protocol of Analyzed Data Server has been used for more than 10 years, and there are many issues found to be modified. However, it is difficult to change the protocol because all the client programs must be replaced at one time. Because it is difficult to change protocol once that protocol is widely used, one should choose which protocol to use carefully. File sharing is simple, and the scientists often propose this method to provide their data. However, the file sharing causes problems, and it is better to consider alternatives.

UDP is not reliable compared to TCP, and there is the possibility of losing packets. As long as the network switch works properly, it is rare that IP multicast packets are lost in the LAN. However, the authors have experienced that the summary graph failed to update because of the packet losses If network trouble occurs, it is difficult to find a way to fix the problem.

Acknowledgments

The authors wish to thank all the people who support the LHD experiment for their help with this work. This work is supported by the National Institute for Fusion Science (budget No.: NIFS10ULHH014, ULII001).

References

.

[1] H.Yamada, K.Yamazaki, K.Y.Watanabe et. al., “Design of central control and man-machine interface systems for large helical device”, Fusion Technol, (1996) 945

[2] Y. Nagayama, M. Emoto, M. Yoshida et. al., “Web technology to support LHD experiment management”, Fusion Engineering and Design. 87 (2012) 2218

[3] H. Nakanishi, M.Ohsuna, M.Kojima et. al., “Data Acquisition and Management System of LHD”, Fusion Sci. Technol. 58 (2010) 445.

[4] M. Emoto, S. Ohdachi, K. Watanabe, et. al., “Server for experimental data from LHD”, Fusion Engineering and Design 81 (2006) 2019

[5] M. Emoto, M.Yoshoda, C. Suzuki et. al., “Performance Improvement in Real-Time Mapping of Thomson Scattering Data to Flux Coordinates in LHD”, J. Plasma Fusion Res. **7** (2012) 2405058

[6] http://gsoap2.sourceforge.net

[7] Dynamics Research Corp, Parallel Process/Network Communication with HARNESS, Users Guide, (1993)

Tables

Table 1. Technologies used to exchange data between heterogenous computers

	Methods	**Applications for the LHD Project**
HIGH	Shared File	Shot number delivery. To retrieve data acquired by sub-systems
	Database	Experimental configuration database, device information, etc.
LOW	Socket Programming	Raw data archiving, communication between sub-systems, etc.
	Web Technology	Virtual printer (CGI), Shot number delivery (PHP), Experiment scheduler, Diagnostics device control (Ruby on Rails)
	RPC	Diagnostics device control (ONC RPC), Analyzed Data Server (SOAP)
	Network Shared Object	Raw Data Acquisition (HARNESS), Data acquisition parameter setting (JavaRMI), Automatic Analyzing System (dRuby)
	UDP	Real time monitoring (unicast), shot number delivery (multicast), new data registration (multicast).

Figure Captions

Fig.1. Data communications among major components. ***(1/2 pp)***

Fig.2. A part of tables stored in the LHD project database. The database manages the location of the physical data as well as the information of the experiments. ***(1/6 pp)***

Fig.3. The sequence diagram of CGI call. Web server runs the CGI program as the request of the client. ***(1/6 pp)***

Fig.4. Summary graph drawn by Lastshot.py. It is displayed in the front screen and sent to virtual printer to produce PDF file. ***(1/6 pp)***

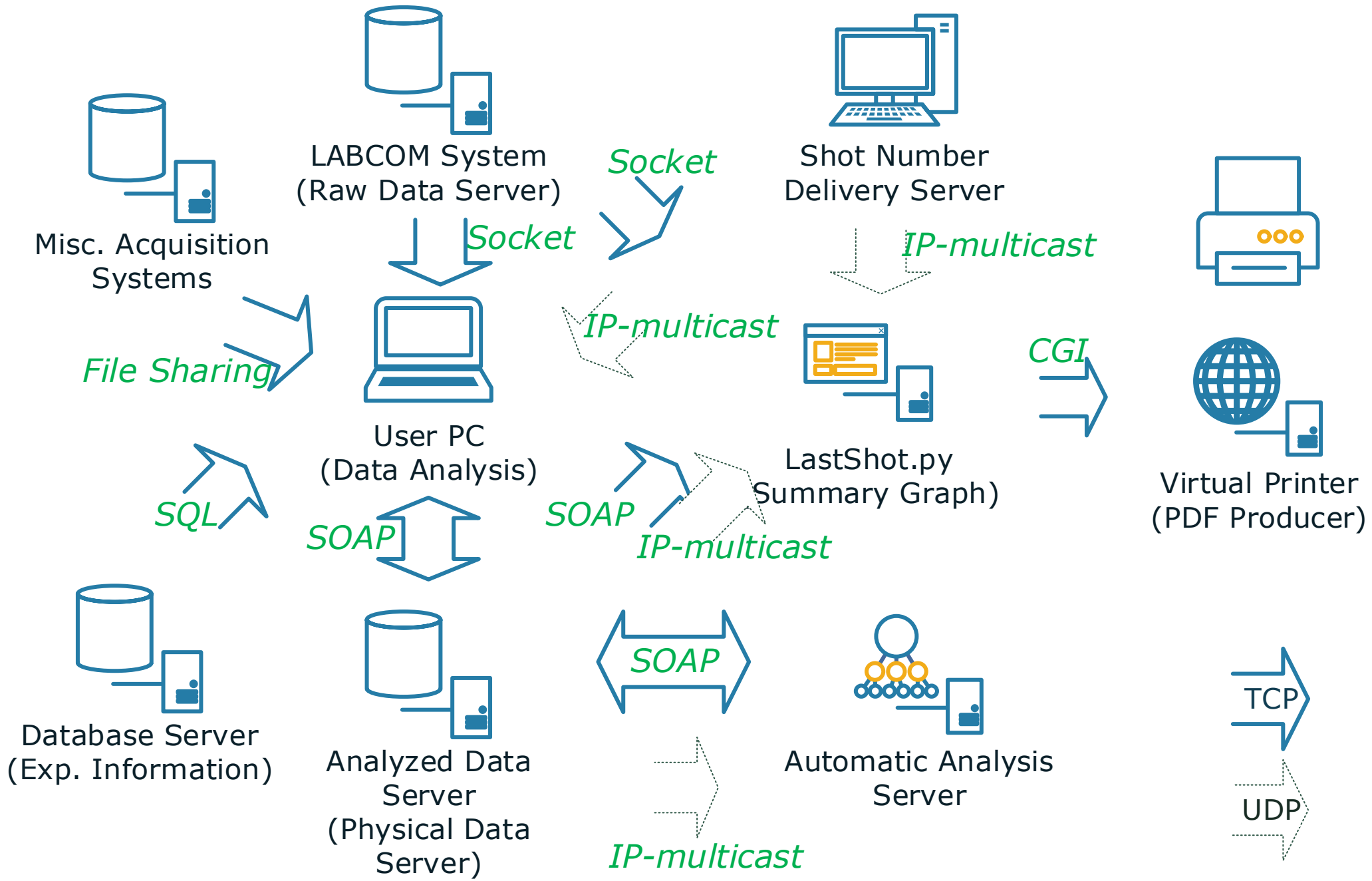
Fig.5. The sequence diagram of Remote Procedure Call (RPC): low level communication is hidden by stubs. ***(1/6 pp)***

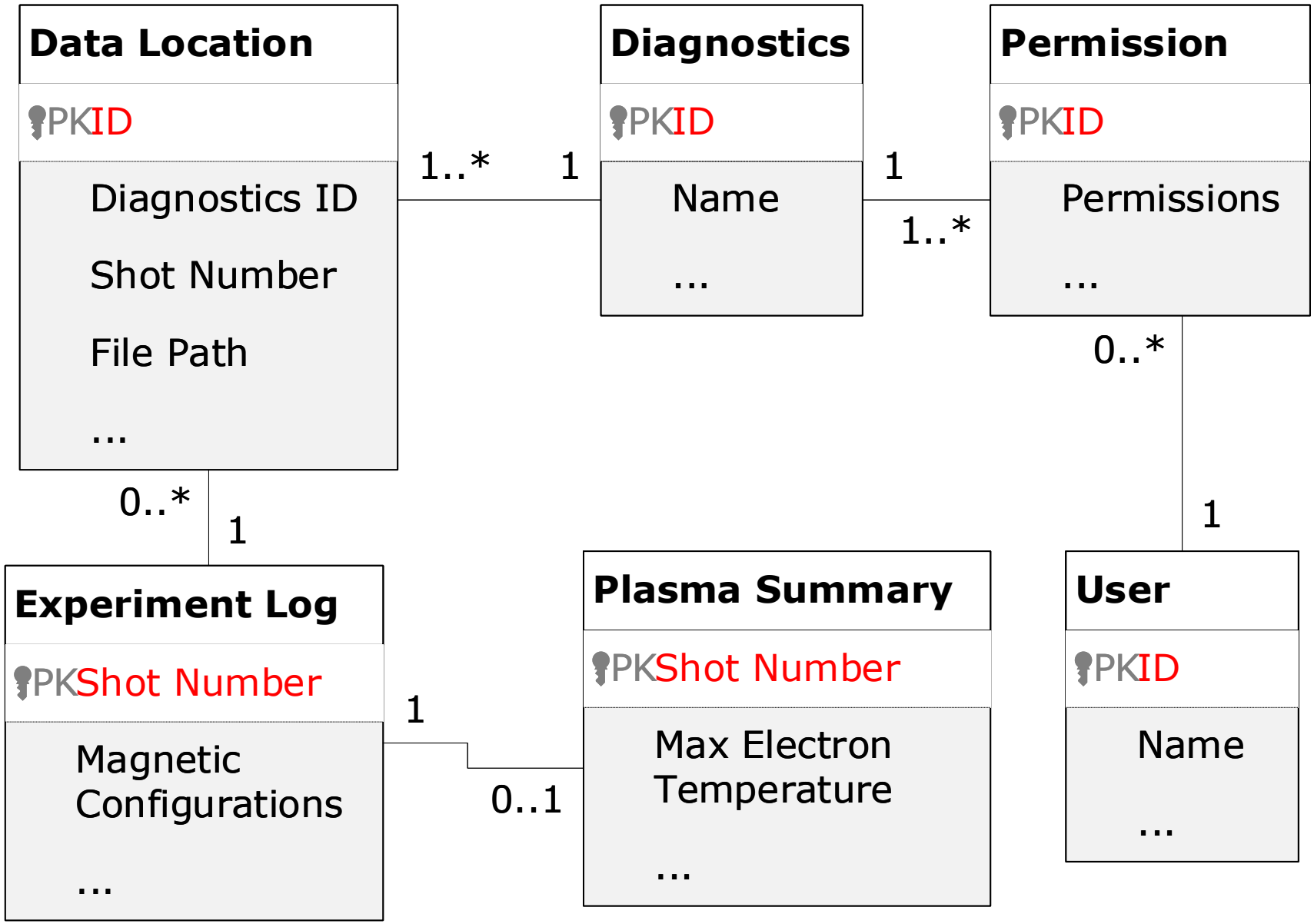
Fig.6. Client and server stubs are produced by compiling IDL file. ***(1/6 pp)***

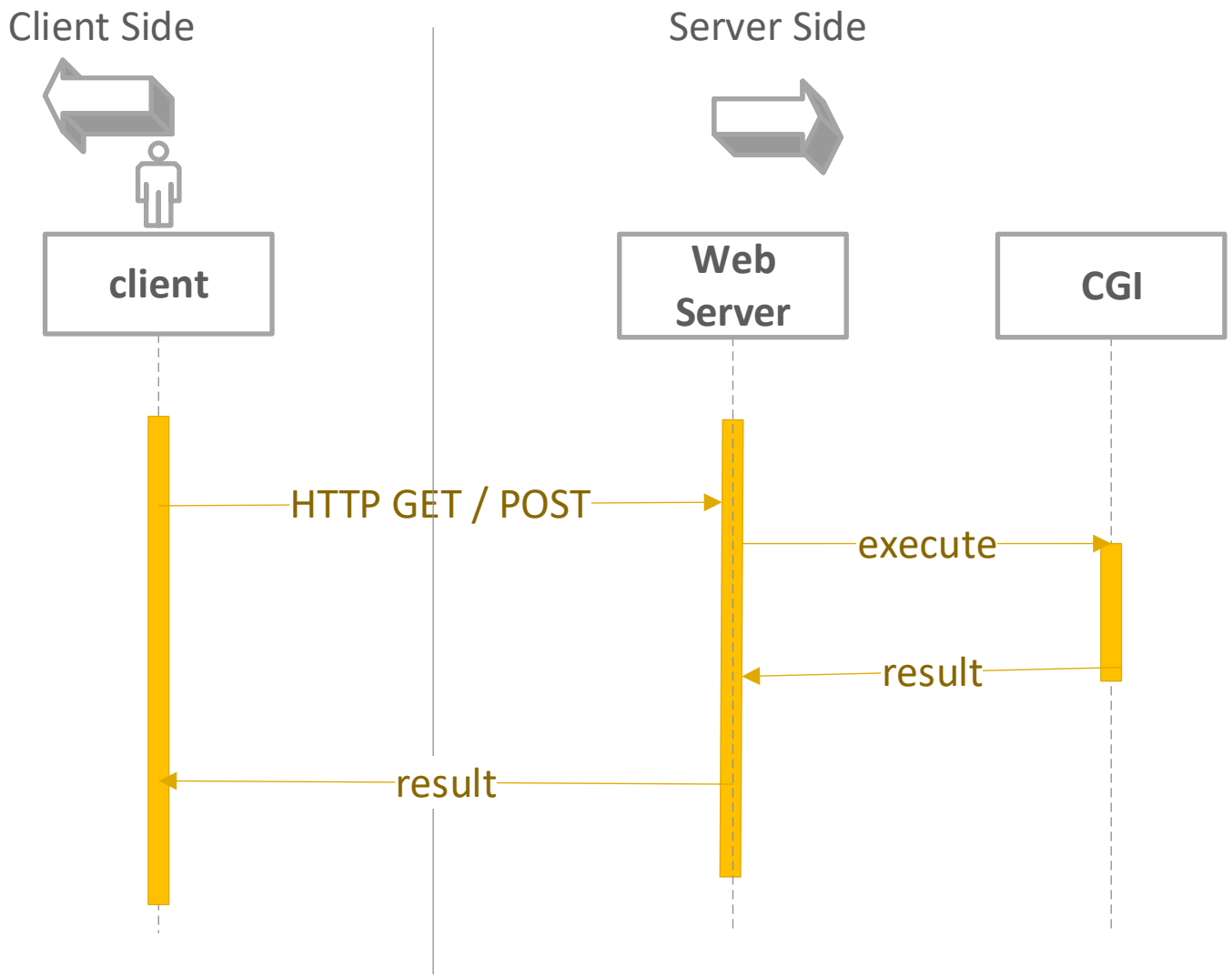
Fig.7. The IDL file for plasma.exe written in RPCL. ***(1/6 pp)***

Fig.8. The sequence diagram of remote object call. The client calls methods of a real object through a proxy object. ***(1/6 pp)***

Fig.9. Real Time Graph. This Java Applet plots the data received from data acquisition systems in real-time by UDP. ***(1/6 pp)***


LABCOM System
(Raw Data Server)
Socket
Shot Number
Delivery Server
Misc. Acquisition
Systems
Socket
IP-multicast
File Sharing
IP-multicast
CGI
User PC
(Data Analysis)
LastShot.py
Summary Graph)
Virtual Printer
(PDF Producer)
SQL
SOAP
SOAP
IP-multicast
Database Server
(Exp. Information)
Analyzed Data
Server
(Physical Data
Server)
SOAP
Automatic Analysis
Server
TCP
UDP
IP-multicast


Data Location
PKID
Diagnostics ID
Shot Number
File Path
...
Diagnostics
PKID
Name
...
Permission
PKID
Permissions
...
1..*
1
1
1..*
0..*
1
0..*
1
Experiment Log
PKShot Number
Magnetic Configurations
...
Plasma Summary
PKShot Number
Max Electron Temperature
...
User
PKID
Name
...
1
0..1


Client Side
Server Side
client
Web Server
CGI
HTTP GET / POST
execute
result
result

137007

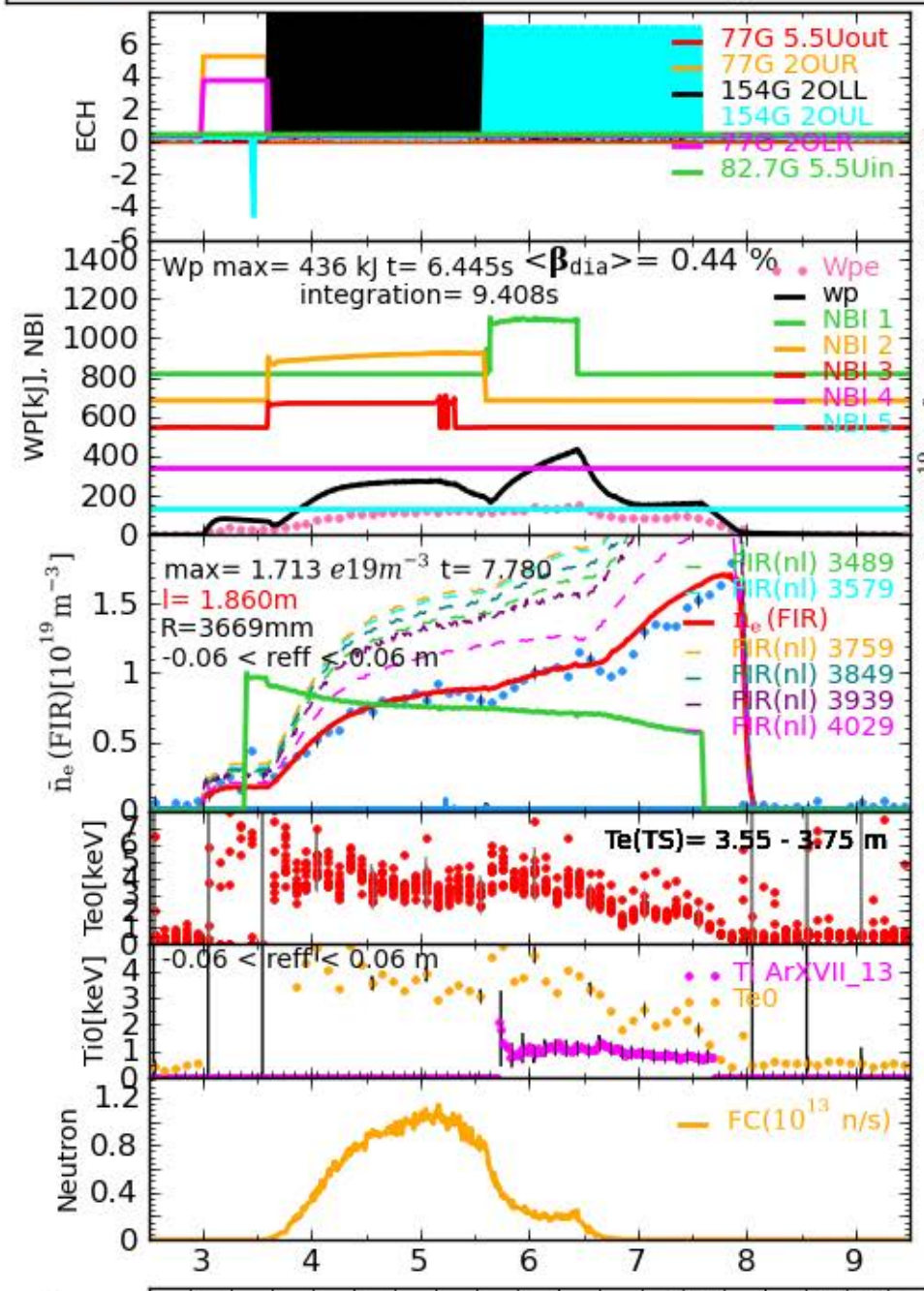
B=2.750T R_{ax}=3.600 m γ=1.254 Bq=100

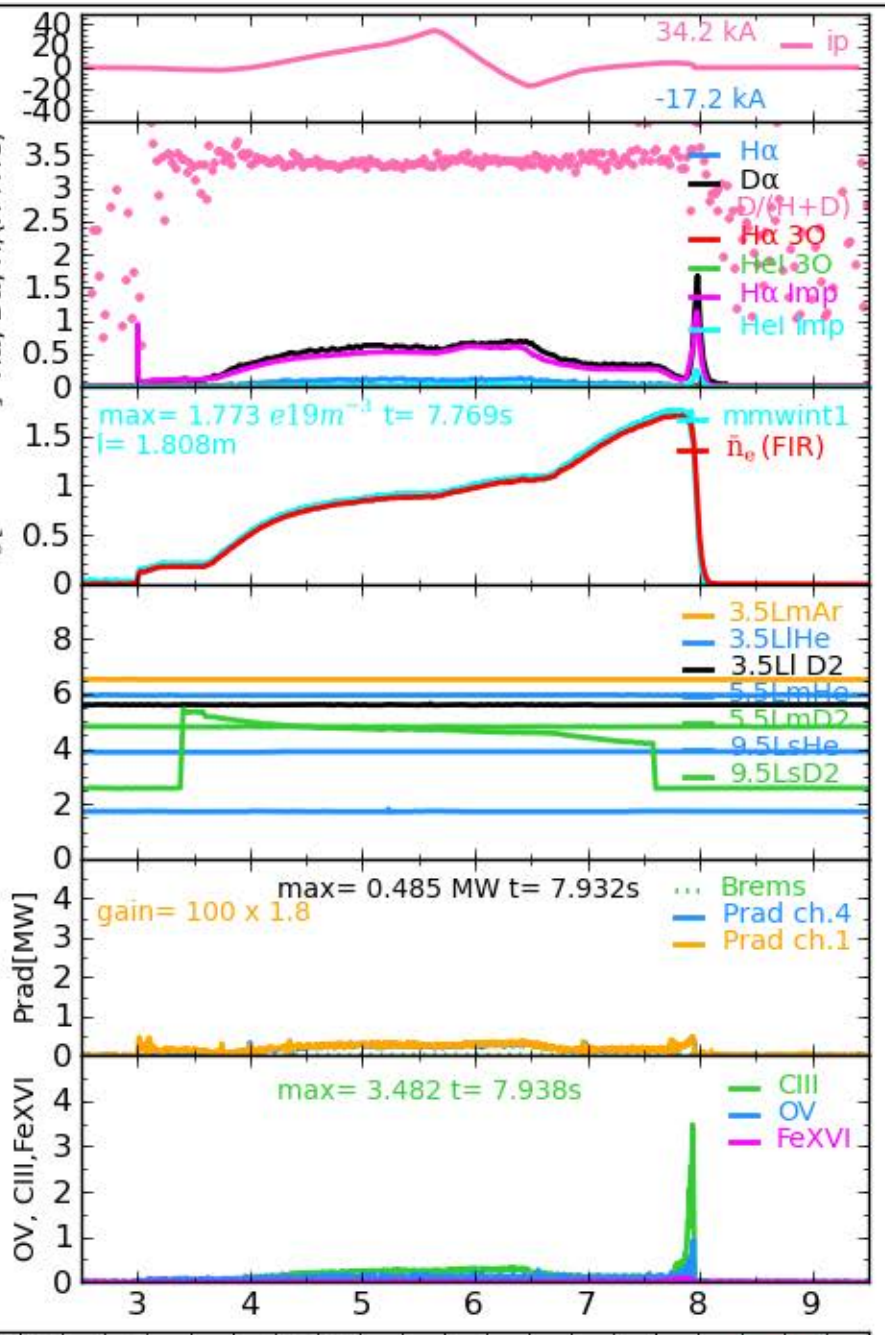
2017-04-27 17:38:28

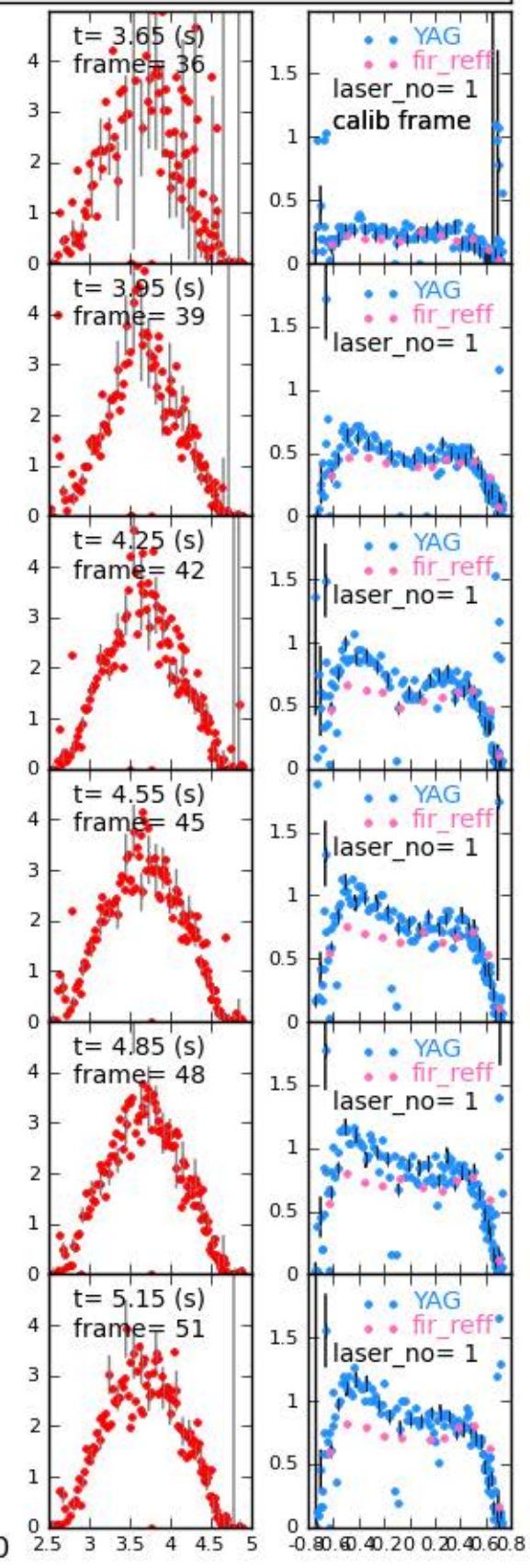
(Boronization 2/13)

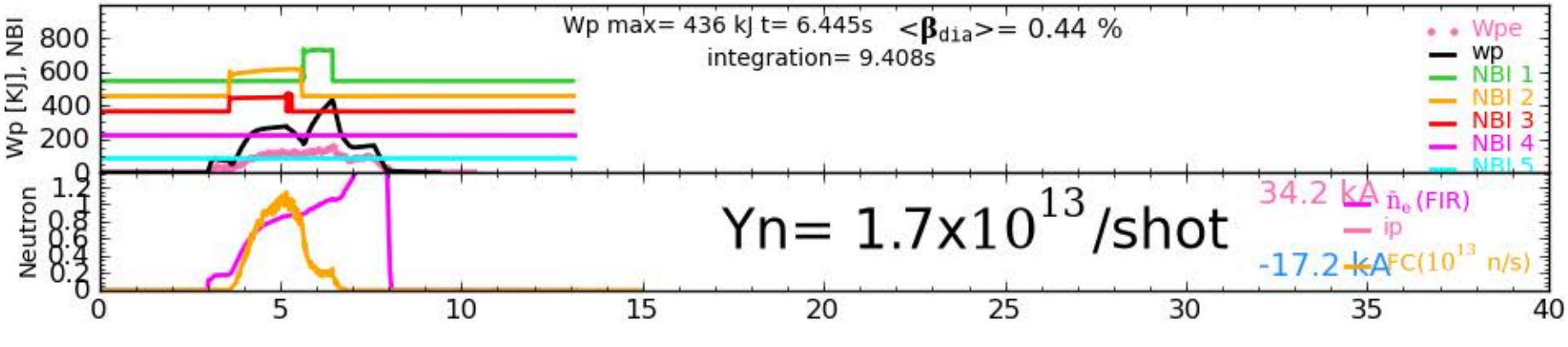
[(3) Core Phys.] turbulent dynamics

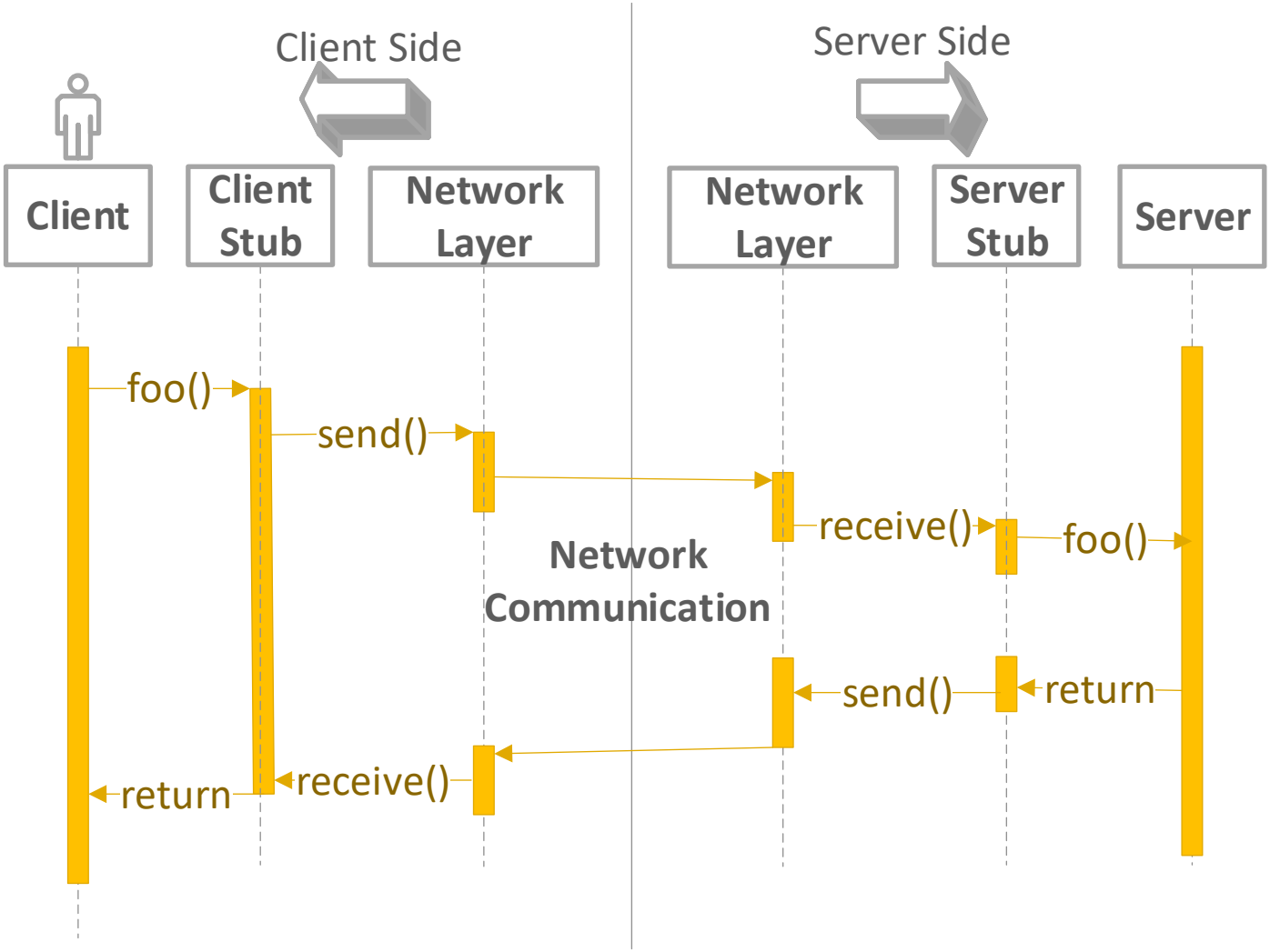
Total neutron per shot = 1.7×10^{13}

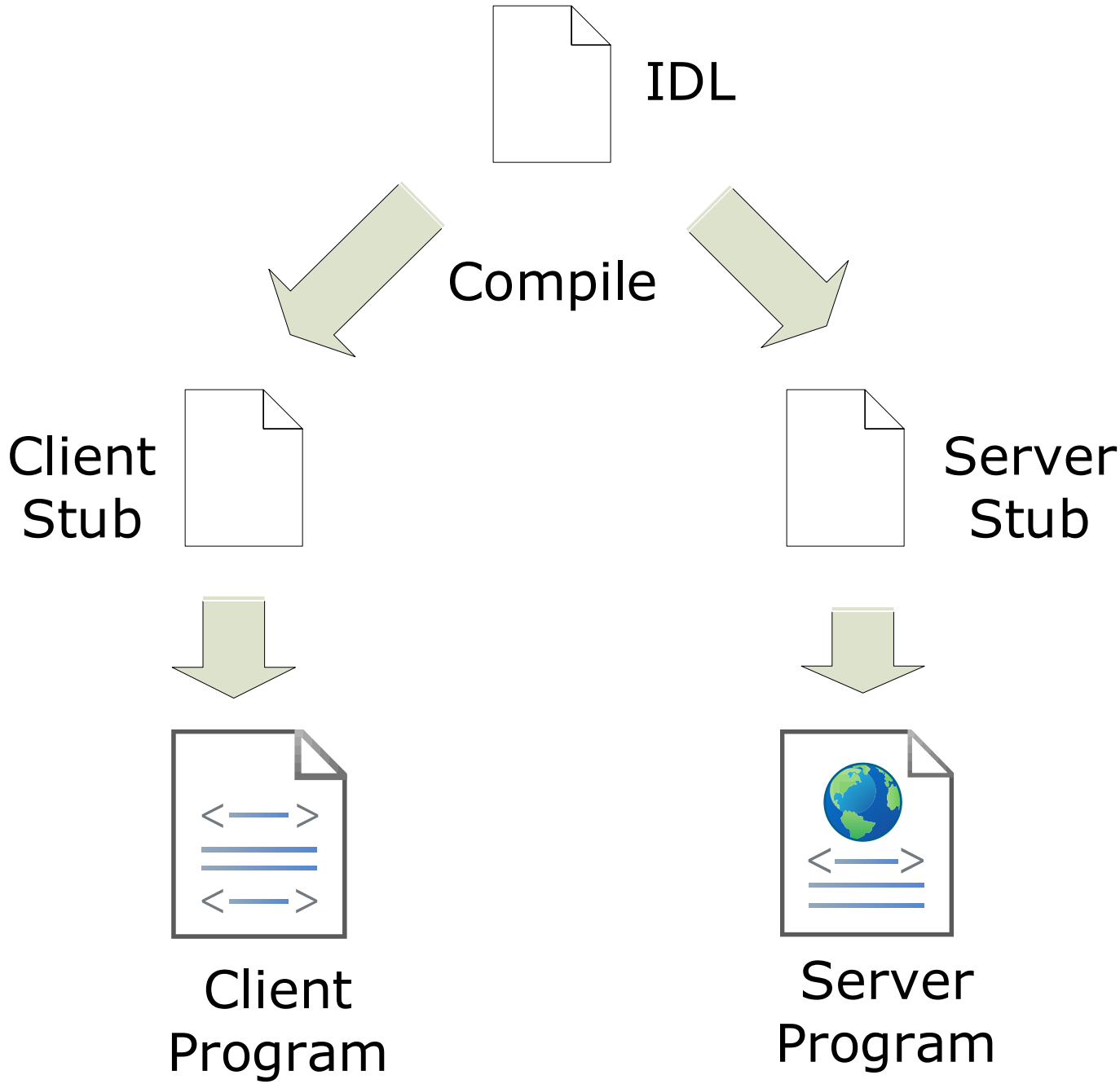






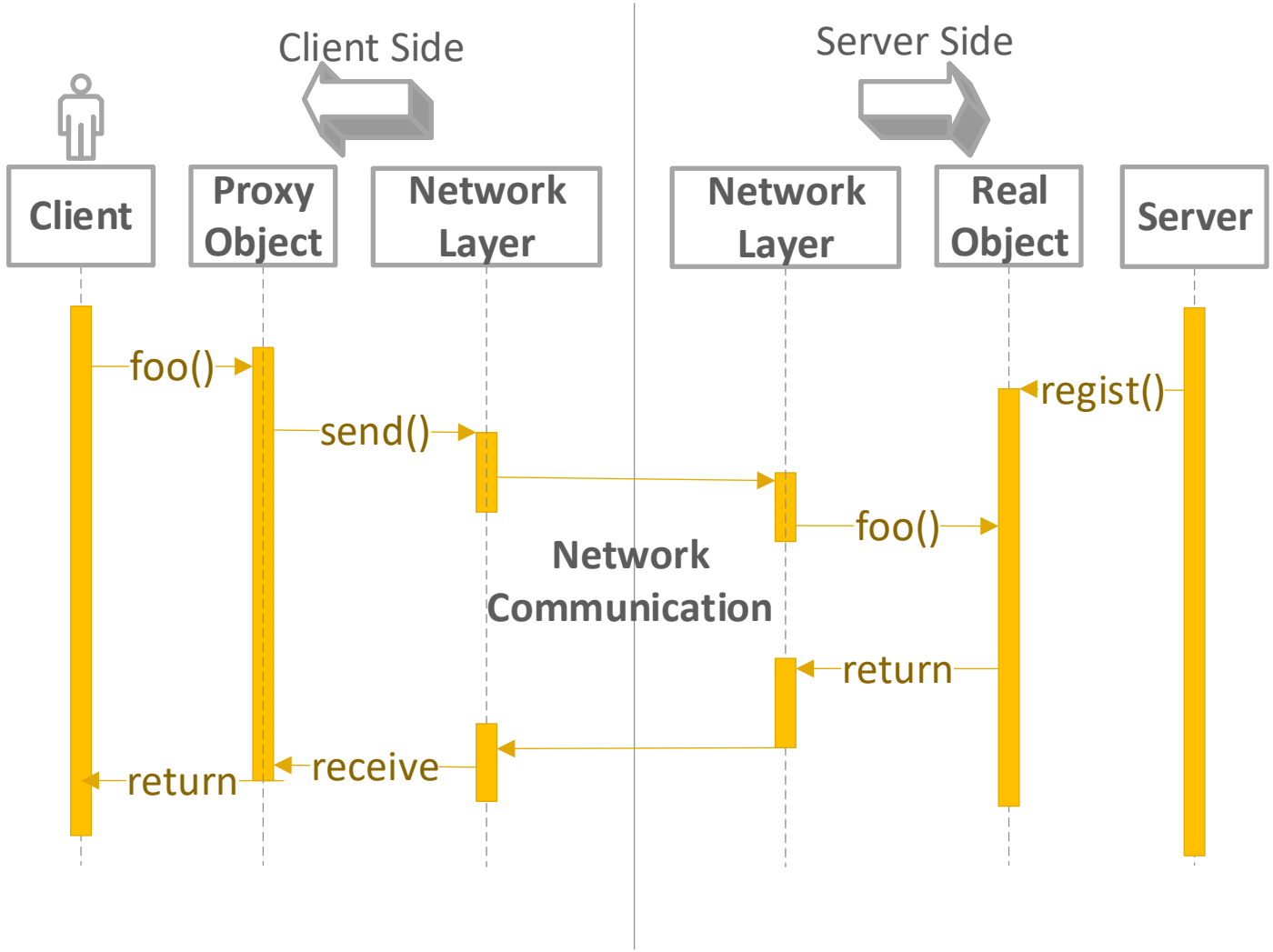


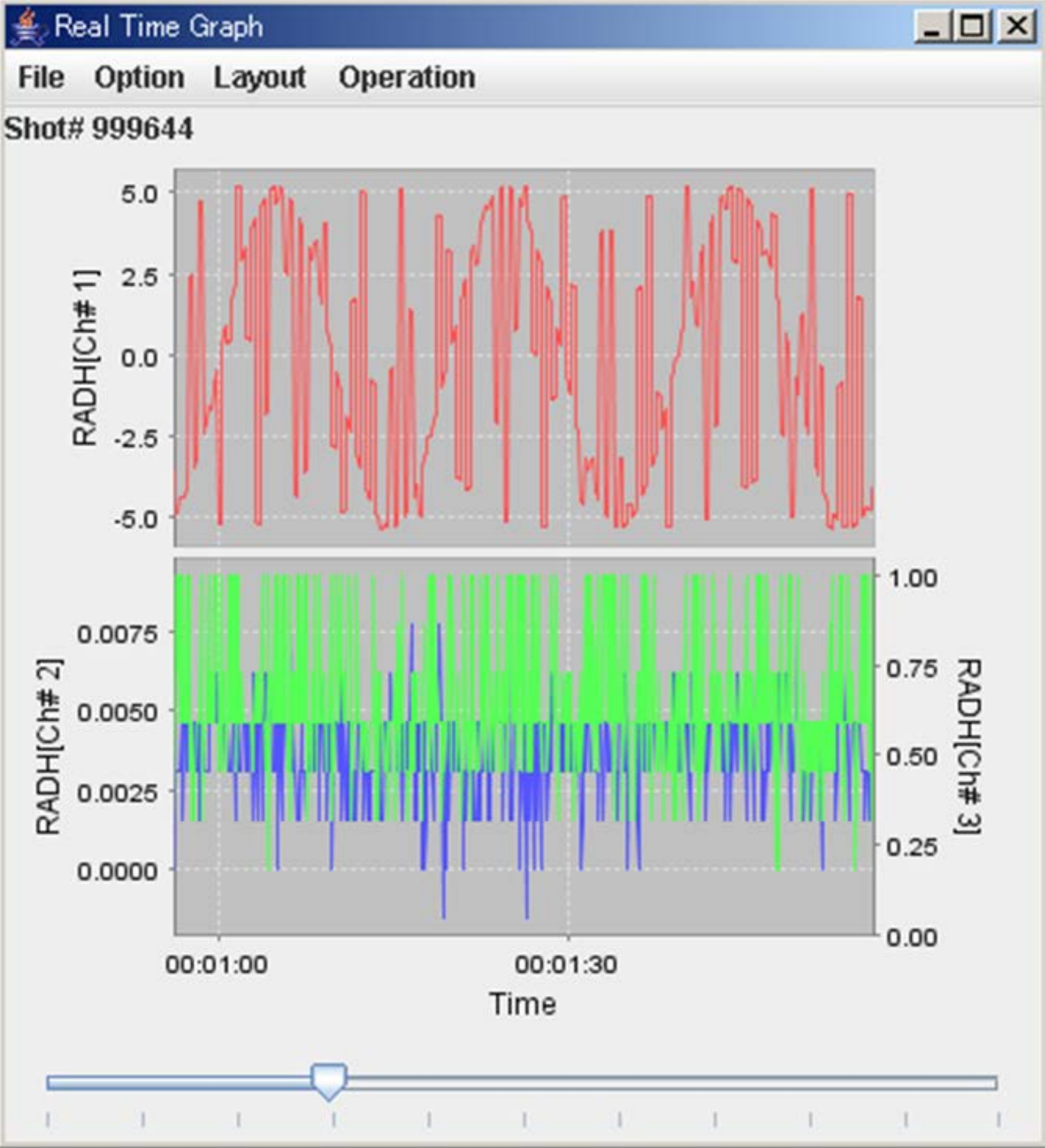

Client Side
Server Side
Client
Client Stub
Network Layer
Network Layer
Server Stub
Server
foo()
send()
receive()
foo()
Network Communication
send()
return
return
receive()


IDL
Compile
Client
Stub
Server
Stub
Client
Program
Server
Program

```
…
…
struct Timing_RPC_ret {
       int    status;
       unsigned int val;
};

program PLASMA_PROC
       {
       version FIRST_VERS
              {
              McInfo
              NETMCGET( McStatus ) = 1;
              McStatus
              NETMCSET( McInfo ) = 2;
              int
              LOCK_CARD( UidInfo ) = 3;
              int
              UNLOCK_CARD( UidInfo ) = 4;
              int
              BEFORESETMCINFOCALL( McStatus ) = 5;
…
```


Client Side
Server Side
Client
Proxy Object
Network Layer
Network Layer
Real Object
Server
foo()
send()
regist()
foo()
Network Communication
return
receive
return


Real Time Graph
File
Option
Layout
Operation
Shot# 999644
5.0
2.5
0.0
-2.5
-5.0
RADH[Ch# 1]
0.0075
0.0050
0.0025
0.0000
RADH[Ch# 2]
1.00
0.75
0.50
0.25
0.00
RADH[Ch# 3]
00:01:00
00:01:30
Time