Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium, Volume 20, No. 1, ISSN 1726-9679
ISBN 978-3-901509-70-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2009
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
DAAAM World Symposium

SIMULINK TCP/IP SERVER

SYSEL, M[artin]

Abstract: *This paper describes TCP/IP Server block in the program MATLAB/Simulink. The new developed Simulink block and instructions for building this one are described here. This server block enables Simulink models to accept network communication from remote applications, devices and from TCP/IP Client Simulink blocks. A very similar functionality (more complex) is provided by the TCP/IP block in the Instrument Control Toolbox offered by MathWorks.*
Key words: *simulink, communications, TCP/IP, Server*

1. INTRODUCTION

Simulink can communicate with remote applications using developed Simulink block. This block enables sending live data from Simulink model to an application, devices using TCP/IP. It is possible to send data to the TCP/IP Server block. The base element of the block is S-function block, which use C MEX file (***, 2008 c). The first, Simulink model stages and callback methods are described, then Windows Socket API implementation is briefly presented and finaly the procedure how to create communication block is described.

2. S-FUNCTION

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink environment. An S-function is a computer language description of a Simulink block written in MATLAB, C, C++, Ada, or Fortran. S-functions are dynamically linked subroutines that the MATLAB interpreter can automatically load and execute. S-functions use a special calling syntax called the S-function API that enables to interact with the Simulink engine. By following a set of simple rules, it can be implemented an algorithm in an S-function and used the S-Function block to add it to a Simulink model.

2.1 S-function Simulation Stages

Execution of a Simulink model proceeds in stages. First comes the initialization phase. In this phase, the Simulink engine incorporates library blocks into the model, propagates signal widths, data types, and sample times, evaluates block parameters, determines block execution order, and allocates memory. The engine then enters a simulation loop, where each pass through the loop is referred to as a simulation step. During each simulation step, the engine executes each block in the model in the order determined during initialization. For each block, the engine invokes functions that compute the block states, derivatives, and outputs for the current sample time. The entire simulation loop then continues until the simulation is complete.

A MEX S-function consists of a set of callback methods that the Simulink engine invokes to perform various block related tasks during a simulation. Because the engine invokes the functions directly, MEX S-functions must follow standard naming conventions specified by the S-function API.

MEX S-functions provide many sample time options, which allow for a high degree of flexibility in specifying when an S-function executes. If the behavior of S-function is a function of discrete time intervals, it can be defined a sample time to control when the Simulink engine calls the S-function mdlOutput and mdlUpdate.

3. WINDOWS SOCKETS

Traditional network programming implemented in Windows environment uses Windows Sockets API (Winsock API - WSA). WSA is similar to Linux Sockets programming with a few exception such as header files, that provided to suit Windows environment and enhances the functionalities. Windows Sockets 2 (Winsock) enables programmers to create advanced Internet, intranet, and other network capable applications to transmit application data across the wire, independent of the network protocol being used. With Winsock, programmers are provided access to advanced Microsoft Windows networking capabilities. Winsock programming previously centered around TCP/IP. (***, 2009). There are two distinct types of socket network appli-cations: Server and Client. Servers and Clients have different behaviors; therefore, the process of creating them is different. The developed Simulink block is a client. Next follows the general model for creating a streaming TCP/IP Server.

3.1 Server

- Initialize Winsock.
- Create a socket.
- Bind the socket.
- Listen on the socket for a client.
- Accept a connection from a client.
- Receive and send data.
- Disconnect.

4. SERVER BLOCK DESCRIPTION

This chapter contains simplified description of the source code of the developed Simulink server block. The base element of the block is S-function block, which use C MEX file. Finally, it is necessary compile source code. Compiling the MEX-Files is similar to compiling with gcc or any other command line compiler. Following MATLAB command links the object code together with the library WS2_32.lib (or it is possible to use older library wsock32.lib). Win32 architecture is supposed. (***, 2008 a, b).

>> *mex -O server.cpp* WS2_32.lib *-DWIN32*

4.1 Defines and Includes

The S-function code starts with the define and include statements. It is necessary to define statement which specifies the name of the S-function.

After defining these two items, the code includes simstruc.h, which is a header file that gives access to the SimStruct data structure and the MATLAB Application Program Interface (API) functions. The simstruc.h file defines a data structure, called the SimStruct, which the Simulink engine uses to maintain information about the S-function. The

simstruc.h file also defines macros that enable MEX-file to set values in and get values (such as the input and output signal to the block) from the SimStruct. The winsock2.h and ws2tcpip.h should be added to access sockets under Microsoft Windows. Next parts describe callback method implementations.

4.2 mdlInitializeSizes

The Simulink engine calls mdlInitializeSizes to inquire about the number of input and output ports, sizes of the ports, and any other information (such as the number of states) needed by the S-function.

The netout implementation of mdlInitializeSizes specifies the following size information:

ssSetNumSFcnParams(S, 2);

It defines two input parameters:

- Port – (Integer input parameter) – Defines listening server port. In computer networking, a port is an application-specific or process-specific software construct serving as a communications endpoint used by Transport Layer protocols of the Internet Protocol Suite such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). A specific port is identified by its number associated with the IP and the protocol used for communication.
- Sample time period – Sampling period of the signal output.

ssSetOutputPortWidth (S,0,DYNAMICALLY_SIZED);

It defines one dynamically sized output port, that's why TCP input is in the special format which is easy modifiable.

4.3 mdlInitializeSizes

The Simulink engine calls mdlInitializeSampleTimes to set the sample times of the S-function. A server block executes in specified period (the third input parameter).

ssSetSampleTime(S,0, mxGetScalar(ssGetSFcnParam(S, 1)));

4.4 mdlStart

Simulink invokes this optional method at the beginning of a simulation. It should initialize and connect the windows socket. Input parameter Port is used, TCP communication is used here.

```
WSAStartup(MAKEWORD(2,2), &wsaData);
getaddrinfo(NULL, DEFAULT_PORT, &hints, &result);
ListenSocket = socket(result->ai_family, result->ai_socktype,
result->ai_protocol);
bind( ListenSocket, result->ai_addr, (int)result->ai_addrlen);
```

4.5 mdlOutputs

The engine calls mdlOutputs at each time step to calculate the block outputs. The server implementation of mdlOutputs takes the socket input signal and writes the data to the Simulink output signal. This example considers just only one listening socket and size of the signal is one. Server works in blocking mode, it means that Server block wait for the data and program execution is blocked. Next part of the program should be executed just only once in the beginning.

```
listen(ListenSocket, SOMAXCONN);
ClientSocket = accept(ListenSocket, NULL, NULL);
// No longer need listening server socket
closesocket(ListenSocket);
```

Following code wait for the data from the socket.

```
recv(ClientSocket, recvbuf, recvbuflen, 0);
output = strtod(recvbuf,NULL);
real_T *y = ssGetOutputPortRealSignal(S,0);
y[0] = output;
```

4.6 mdlTerminate

The engine calls mdlTerminate to provide the S function with an opportunity to perform tasks at the end of the simulation. This is a mandatory S function routine. The server S-function terminate created socket.

```
shutdown(ClientSocket, SD_SEND);
closesocket(ClientSocket);
WSACleanup();
```

4.7 Server Block

The TCP/IP server block accepts data from the network socket, it uses TCP/IP protocol. The data are received at fixed intervals during a simulation. The TCP/IP server block has one input port. The size of the output port is one, but next version of the developed server block will have dynamic size (inherited from the incoming data). This block has no input ports.

The Sink Block Parameters dialog box can be used for selecting communication parameters (Figure 1).

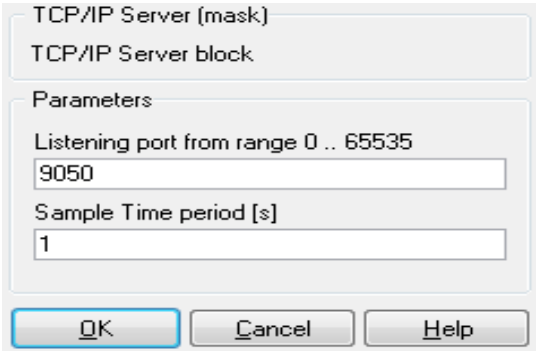

Fig. 1. The TCP/IP Server Block Parameters dialog box

It is possible to specify a remote server address, port and sample time period. The sample time period is the rate at which the block send the data to specified port on the server during the simulation.

5. CONCLUSION

The TCP/IP server block has been developed. This server block enables Simulink models to accept network communication from remote applications and devices over TCP/IP network. This paper describes this block and simplified instructions for building this block. Final code is open source. The code and behaviour of the server are more complicated and offer many possibilities to the future (e.g. implement protocol).

6. ACKNOWLEDGMENTS

This work was supported by the Ministry of Education of the Czech Republic under grant No. MSM 7088352101.

7. REFERENCES

*** (2008). *Writing S-Functions*. The Mathworks Inc., Natick, USA

*** (2008). *MATLAB C and Fortran API reference*. The Mathworks Inc., Natick, USA

*** (2008). *Instrument Control Toolbox 2.7*. The Mathworks Inc., Natick, USA

*** (2009). Windows Sockets 2. *Available from*: http://msdn.microsoft.com/en-us/library/ms740673(VS.85).aspx. *Accessed:* 2009-03-15