**Assignment 1: TCP Client-Server to Print and Manipulate Socket Options**

Table of Contents

[Introduction 2](#_Toc74862158)

[Why C++11? 2](#_Toc74862159)

[Notes on Limitations and Design Constraints 2](#_Toc74862160)

[Results and Program Execution 3](#_Toc74862161)

[References 8](#_Toc74862162)

[Figure 1: Build the project and start the server on Eros. 4](#_Toc74862182)

[Figure 2: Run the client on Zeus. 5](#_Toc74862183)

[Figure 3: Server on Eros processes client payload sent from Zeus. 6](#_Toc74862184)

[Figure 4: New server socket opts after processing. Binary flags are inverted, numeric types are changed, timeval is set to sec = 1 and usec = 50000, Linger is kept intentionally the same. 7](#_Toc74862185)

[Figure 5: Shutting down the server with Ctrl + C. Also Ctrl + Z may be used, or the Linux kill command. 7](#_Toc74862186)

# Introduction

For this assignment, I decided to take some risk and covert the TCP client and server code from C to C++ while removing the dependency to the unp.h header file. The intent was to use readily available, modern networking libraries. Initially I thought to leverage classes and use an object-oriented design approach. I stuck to functional programming as there is no need to instantiate classes or deal with object reuse; the program is rather straightforward and short.

Source code for this program may be found on GitHub here:

<https://github.com/bss8/tcp-socket-options>

## Why C++11?

A small aside but worth mentioning. Why use c++11 compiler? C++11 now supports:

* lambda expressions,
* automatic type deduction of objects,
* uniform initialization syntax,
* delegating constructors,
* deleted and defaulted function declarations,
* nullptr,
* rvalue references

"The C++11 Standard Library was also revamped with new algorithms, new container classes, atomic operations, type traits, regular expressions, new smart pointers, async() facility, and of course a multithreading library."

## Notes on Limitations and Design Constraints

When setting socket options to new values, we must be mindful. SO\_SNDBUF and SO\_RCVBUF have an upper limit, which if exceeded will ignore the value we are trying to use. Otherwise, if we set it to 50,000 for example, the value displayed will be double – 100,000. Likewise, if we set it to 40,000 the value displayed will be 80,000.

Please also note that some options may not be changed – either the protocol is not available, or the server restricts access and modification fails with “Permission denied.” This occurs for SO\_DEBUG, SO\_TYPE, SO\_SNDLOWAT, and TCP\_MAXSEG.

TCP\_MAXSEG: “The maximum segment size for outgoing TCP packets. In Linux 2.2 and earlier, and in Linux 2.6.28 and later, if this option is set before connection establishment, it also changes the MSS value announced to the other end in the initial packet. Values greater than the (eventual) interface MTU have no effect. TCP will also impose its minimum and maximum bounds over the value provided.”

<https://linux.die.net/man/7/tcp>

# Results and Program Execution

The program is correct and performs the specified functionality. The client sends a string payload to the server, containing server options and values. It is in the form: opt\_str,opt\_name,opt\_val

Where opt\_str is the string representation of the socket option, opt\_name is an integer representation of the socket option, and opt\_val is the value. In the case of non-binary types, the value may consist of two items.

Binary values are inverted. If it is off on the client, it is set to on in the server and vice versa. For numeric types, a small value less than 10,000 is simply doubled. Larger values are set to 30,000 (which appear as twice this when getsockopts is invoked, thus we see 60,000).

Certain socket options are not available – getsockopts does not find them. In this case we skip it. Other times, the operating system has restrictions to prevent overutilization of resources. In the case of SO\_DEBUG, we receive a “Permission denied” error when we try to enable it. Lastly, some protocols are not available.

Overall, the client on Zeus TXST Linux server should send 20 socket options to the server on Eros. The server should, in turn, process 20 socket options.

Here is the client payload:

Msg received from client is: SO\_BROADCAST,6,off;SO\_DEBUG,1,off;SO\_DONTROUTE,5,off;SO\_ERROR,4,0;SO\_KEEPALIVE,9,off;SO\_LINGER,13,l\_onoff = 0, l\_linger = 0;SO\_OOBINLINE,10,off;SO\_RCVBUF,8,367360;SO\_SNDBUF,7,87040;SO\_RCVLOWAT,18,1;SO\_SNDLOWAT,19,1;SO\_RCVTIMEO,20,0 sec, 0 usec;SO\_SNDTIMEO,21,0 sec, 0 usec;SO\_REUSEADDR,2,off;SO\_REUSEPORT,15,off;SO\_TYPE,3,1;IP\_TOS,1,0;IP\_TTL,2,64;TCP\_MAXSEG,2,1448;TCP\_NODELAY,1,off;

Below are screenshots demonstrating a full run of the application on the TXST Linux hosts Zeus and Eros. Please refer to the README.md file for instructions on how to build and run this application and the source code for implementation details.

Text

Description automatically generated

Figure : Build the project and start the server on Eros.

Text

Description automatically generated

Figure : Run the client on Zeus.

Text

Description automatically generated

Figure : Server on Eros processes client payload sent from Zeus.

Text

Description automatically generated

Figure : New server socket opts after processing. Binary flags are inverted, numeric types are changed, timeval is set to sec = 1 and usec = 50000, Linger is kept intentionally the same.

A screenshot of a computer

Description automatically generated with low confidence

Figure : Shutting down the server with Ctrl + C. Also Ctrl + Z may be used, or the Linux kill command.

# References

[1] <https://stackoverflow.com/questions/9402254/how-do-you-run-a-function-on-exit-in-c>  
[2] <https://notes.shichao.io/unp/ch7/#so_broadcast-socket-option>  
[3] <https://www.bogotobogo.com/cplusplus/sockets_server_client.php>  
[4] <https://www.beej.us/guide/bgnet/html/>  
[5] <https://smartbear.com/blog/develop/the-biggest-changes-in-c11-and-why-you-should-care/>

[6] <https://stackoverflow.com/questions/4654636/how-to-determine-if-a-string-is-a-number-with-c>

[7] <https://linux.die.net/man/7/tcp>  
[8] W. R. Stevens, Bill Fenner, and Andrew M. Rudoff. UNIX Network Programming – Networking APIs: Sockets and XTI (3nd ed.). Addison-Wesley, 2004. ISBN: 0-13-141155-1.

--- NOTHING FOLLOWS ---