

**Department of Computer Science**

**CS2005 Networks & Operating Systems Task 1**

Academic Year 2020-21

Jermain Dunkley

1939797

Table of Contents

[1. Introduction 3](#_Toc55158980)

[2. Test Network Documentation 4](#_Toc55158981)

[3. calcClient and calcServer Documentation 7](#_Toc55158982)

[4. calcClientUpdate and calcServerUpdate Documentation 9](#_Toc55158983)

[5. Report to the NOSSoft Management 11](#_Toc55158984)

[6. Conclusions 14](#_Toc55158985)

# 1. Introduction

This report is in response to a previous claim by NOSSoft and attempts to provide evidence and explain where necessary the problems arising in the school’s new virtual learning application.

For a quick summary of the events to current date, I had discovered numerous bugs—ones that render the application unusable—in the newest update to the application. Previously, the system worked incredibly well, performing all its functions exactly as it ought to, however the newest update makes the application worse than it was before.

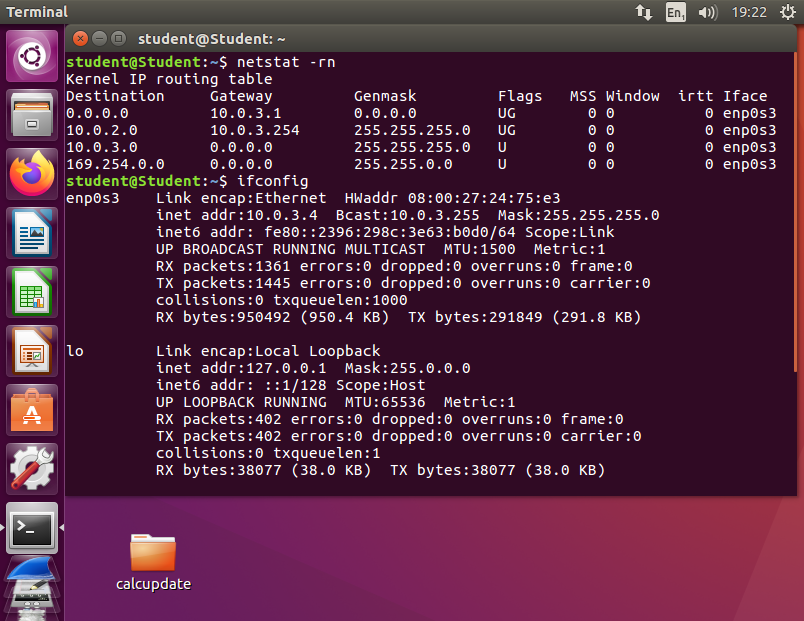
I immediately sought to contact NOSSoft to rollback the update until such a fix for it could be implemented, however I was repeatedly met with reluctance and skepticism, suggesting that there instead might be a problem with our network setup.

As such, this report sets to ease that skepticism and, more importantly, aid in the smooth rollout of a new update that has fixed the various bugs that have arisen in the current update. This report will make clear the various problems that have been encountered so far using Wireshark to systematically capture all transactions between two separate hosts. Both the original and updated client and server will be shown for easy comparison via screenshots. This report will also provide a protocol table detailing the sequence of events between server and client, as well as evidence that our current network settings are correct and running well.

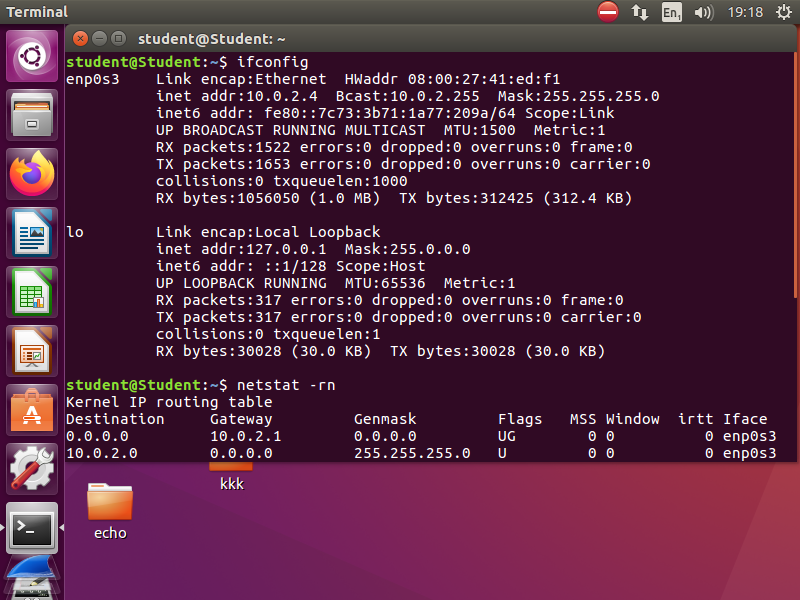
Finally, using the screenshots of both the original and updated calcClient and calcServer, I will layout the problems, how I came across them, the source of those problems being either the client or server, and how those problems should be expected to run.

# 2. Test Network Documentation

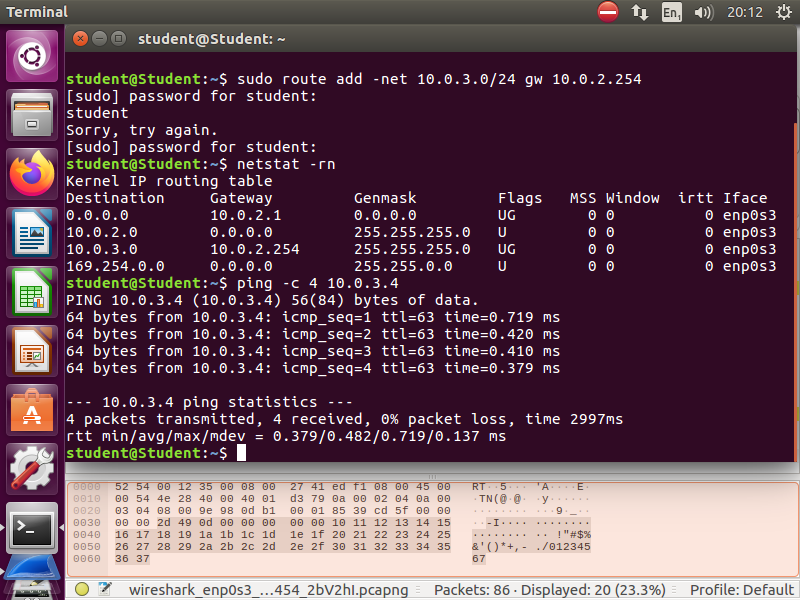
The following will show that the network I am running the application on is working correctly:

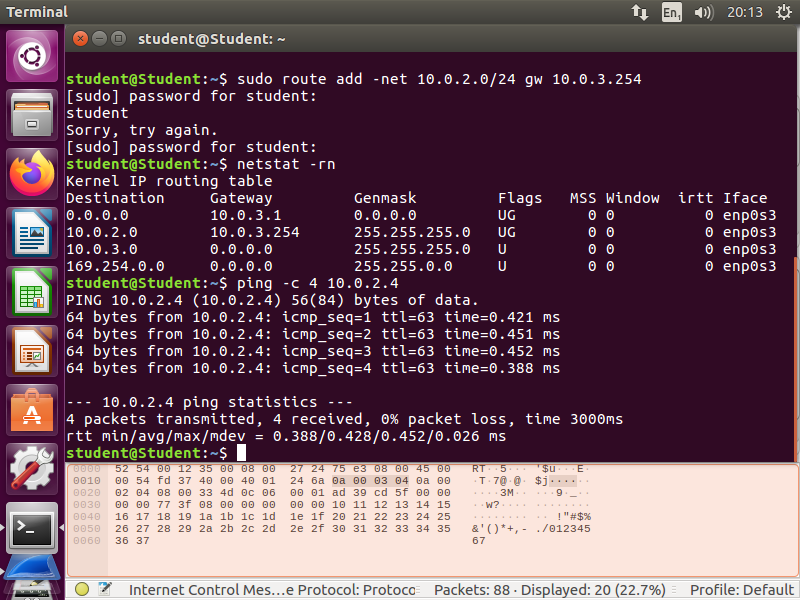


The first thing to note is the IP Address of the client server. After running a test using ifconfig, the figure brought back the inet address, a figure which usually incorporates the host name as well as the IP Address itself. For client machine being run, this appeared as 10.0.3.4.

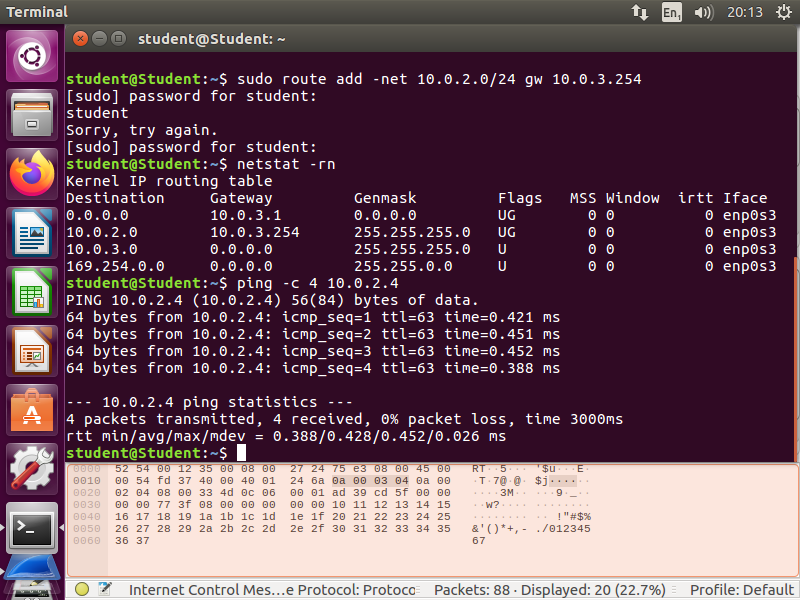
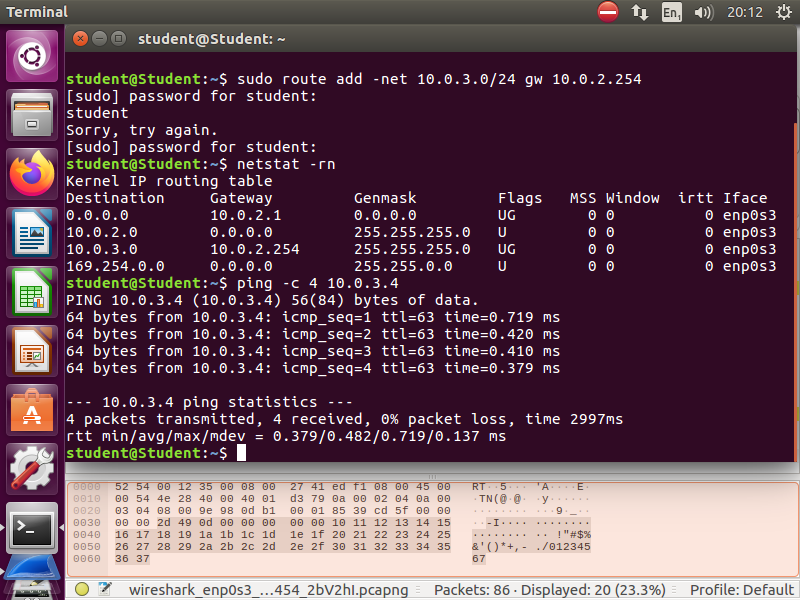


Likewise, for the server machine, when ifconfig is entered, an IP Address of 10.0.2.4 is seen. Furthermore, I can also provide evidence that packets can be exchanged between source and destination from each host.

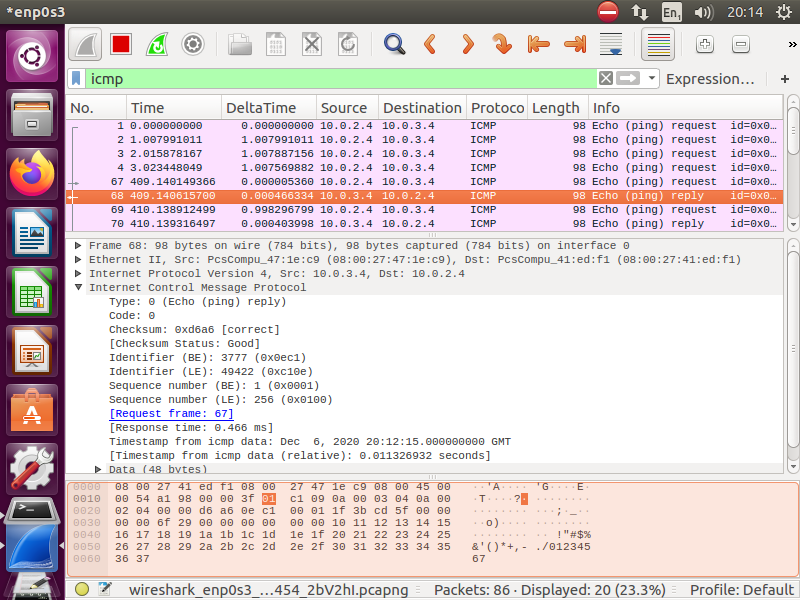


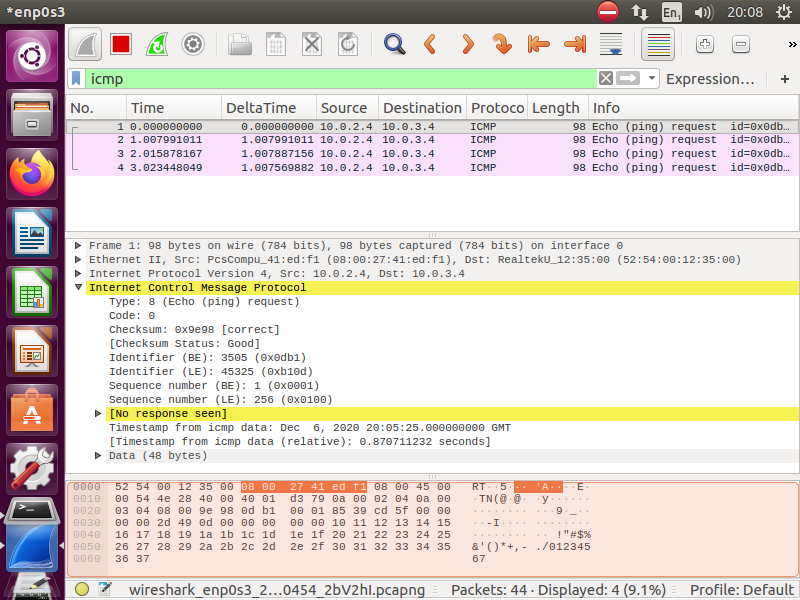


Executing netstat –rn showcases the connection between the two hosts on both sides. This shows each particular host being able to access the other as a destination, indicated by the IP Addresses underneath the Destination column.



Executing a ping request of 4 packets to both IP Addresses, shows that both were successfully received on either end. This can also be seen when captured and analysed with Wireshark.

The message (no.68) is sending a reply back to the IP Address which requested the packet. In this case, the source for the reply is the 10.0.3.4 host, while the destination is the 10.0.2.4 host. The request and reply would be the opposite, however if initiating ping on the 10.0.3.4 host. Without the host machines being connected, if a ping request was to be sent to either host, the following response would occur instead:



# 3. calcClient and calcServer Documentation

|  |  |  |  |
| --- | --- | --- | --- |
|  | **calcClient** |  | **calcServer** |
|  |  |  | [run calcServer] |
|  | [run calcClient] |  |  |
|  |  |  | [accept calcClient connection] |
|  | WHILE NOT TERMINATED |  | WHILE NOT TERMINATED |
|  |  | #S1 | SEND “calculator server ready and waiting” TO calcClient |
|  | RECEIVE “calculator server ready and waiting” FROM calcServer |  |  |
|  | READ “menuOption” FROM user |  |  |
|  | IF USER INPUT “1” |  |  |
|  | SEND “add operands” TO calcServer |  |  |
|  |  |  | RECEIVE “add operands” FROM calcClient |
|  |  |  | SEND “send operands to add” TO calcClient |
|  | RECEIVE “send operands to add” FROM calcServer |  |  |
|  | USER INPUT “3 1” |  |  |
|  |  |  | RECEIVE “3 1” FROM calcClient |
|  |  |  | SEND “4” To calcClient |
|  | RECEIVE “4” FROM calcServer |  |  |
|  | PRINT “4” |  |  |
| #C1 | SEND “next operation please” TO calcServer |  |  |
|  |  |  | RECEIVE “next operation please” FROM calcClient |
|  |  |  | Go back to #S1 |
|  | IF USER INPUT “2” |  |  |
|  | SEND “sub operands” TO calcServer |  |  |
|  |  |  | RECEIVE “sub operands” FROM calcClient |
|  |  |  | SEND “send operands to sub” TO calcClient |
|  | RECEIVE “send operands to sub” FROM calcServer |  |  |
|  | USER INPUT “3 1” |  |  |
|  |  |  | RECEIVE “3 1” FROM calcClient |
|  |  |  | SEND “2” TO calcClient |
|  | RECEIVE “2” FROM calcServer |  |  |
|  | PRINT “2” |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “3” |  |  |
|  | SEND “multi operands” TO calcServer |  |  |
|  |  |  | RECEIVE “multi operands” FROM calcClient |
|  |  |  | SEND “send operands to multiply” TO calcClient |
|  | RECEIVE “send operands to multiply” FROM calcServer |  |  |
|  | USER INPUT “4 2” |  | RECEIVE “4 2” FROM calcClient |
|  |  |  | SEND “8” TO calcClient |
|  | RECEIVE “8” FROM calcServer |  |  |
|  | PRINT “8” |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “4” |  |  |
|  | SEND “div operands” TO calcServer |  |  |
|  |  |  | RECEIVE “div operands” FROM calcClient |
|  |  |  | SEND “send operations to divide” TO calcClient |
|  | RECEIVE “send operands to divide” FROM calcServer |  |  |
|  | USER INPUT “16 2” |  |  |
|  |  |  | RECEIVE “ 16 2” FROM calcClient |
|  |  |  | SEND “8” TO calcClient |
|  | RECEIVE “2” FROM calcServer |  |  |
|  | PRINT “2” |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “0” |  |  |
|  | SEND “endcomms” TO calcServer |  |  |
|  | [Terminate] |  |  |
|  |  |  | RECEIVE “endcomms” FROM calcClient |
|  |  |  | [Terminate] |
|  | ENDWHILE |  | ENDWHILE |

# 4. calcClientUpdate and calcServerUpdate Documentation

|  |  |  |  |
| --- | --- | --- | --- |
|  | **updatedCalcClient** |  | **updatedCalcServer** |
|  |  |  | [run calcServer] |
|  | [run calcClient] |  |  |
|  |  |  | [accept calcClient connection] |
|  | WHILE NOT TERMINATED |  | WHILE NOT TERMINATED |
|  |  | #S1 | SEND “calculator server ready and waiting” TO calcClient |
|  | RECEIVE “calculator server ready and waiting” FROM calcServer |  |  |
|  | READ “menuOption” FROM user |  |  |
|  | IF USER INPUT “1” |  |  |
|  | SEND “add operands” TO calcServer |  |  |
|  |  |  | RECEIVE “add operands” FROM calcClient |
|  |  |  | **SEND “send operands to sub” TO calcClient** |
|  | **RECEIVE “send operands to sub” FROM calcServer** |  |  |
|  | USER INPUT “3 1” |  |  |
|  |  |  | RECEIVE “3 1” FROM calcClient |
|  |  |  | **SEND “2” To calcClient** |
|  | **RECEIVE “2” FROM calcServer** |  |  |
|  | **PRINT “2”** |  |  |
| #C1 | SEND “next operation please” TO calcServer |  |  |
|  |  |  | RECEIVE “next operation please” FROM calcClient |
|  |  |  | Go back to #S1 |
|  | IF USER INPUT “2” |  |  |
|  | SEND “sub operands” TO calcServer |  |  |
|  |  |  | RECEIVE “sub operands” FROM calcClient |
|  |  |  | **SEND “send operands to add” TO calcClient** |
|  | **RECEIVE “send operands to add” FROM calcServer** |  |  |
|  | USER INPUT “3 1” |  |  |
|  |  |  | RECEIVE “3 1” FROM calcClient |
|  |  |  | **SEND “4” TO calcClient** |
|  | **RECEIVE “4” FROM calcServer** |  |  |
|  | **PRINT “4”** |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “3” |  |  |
|  | SEND “multi operands” TO calcServer |  |  |
|  |  |  | RECEIVE “multi operands” FROM calcClient |
|  |  |  | SEND “send operands to multiply” TO calcClient |
|  | RECEIVE “send operands to multiply” FROM calcServer |  |  |
|  | USER INPUT “4 2” |  | RECEIVE “4 2” FROM calcClient |
|  |  |  | **SEND “16” TO calcClient** |
|  | **RECEIVE “16” FROM calcServer** |  |  |
|  | **PRINT “16”** |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “0” |  |  |
|  | **SEND “div operands” TO calcServer** |  |  |
|  |  |  | RECEIVE “div operands” FROM calcClient |
|  |  |  | SEND “send operations to divide” TO calcClient |
|  | RECEIVE “send operands to divide” FROM calcServer |  |  |
|  | USER INPUT “16 2” |  |  |
|  |  |  | RECEIVE “ 16 2” FROM calcClient |
|  |  |  | SEND “8” TO calcClient |
|  | RECEIVE “8” FROM calcServer |  |  |
|  | PRINT “8” |  |  |
|  | Go back to #C1 |  |  |
|  | IF USER INPUT “4” |  |  |
|  | **SEND “endcomms” TO calcServer** |  |  |
|  | [Terminate] |  |  |
|  |  |  | RECEIVE “endcomms” FROM calcClient |
|  |  |  | [Terminate] |
|  | ENDWHILE |  | ENDWHILE |

The updated calcClient and calcServer protocols were captured using Wireshark and analysing the transactions of packets between two machines. This information was displayed in Wireshark’s packet-contents window from which I was able to glean the problems that have arisen from the update. These problems have been identified in red in the above protocol table.

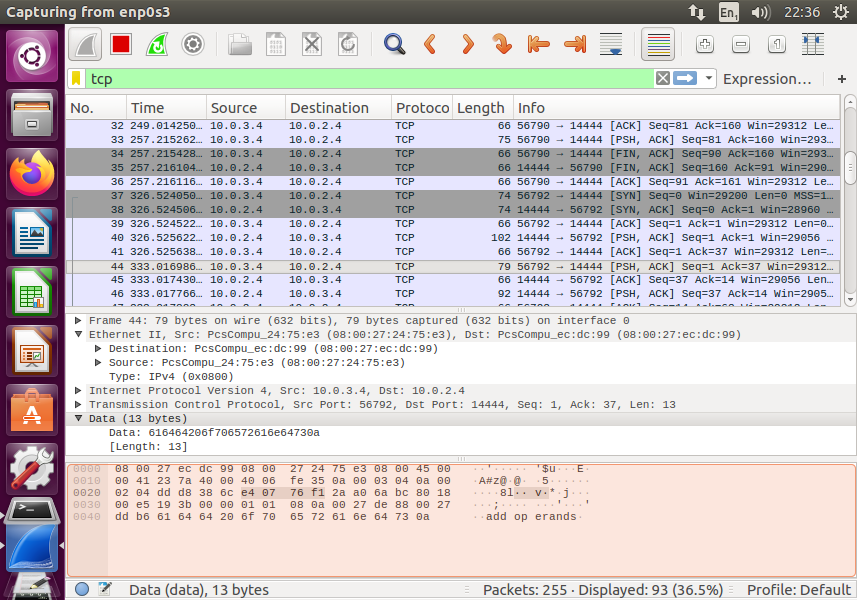
What I have documented would lead to me believe that certain parts of the code for the application are linking to methods they shouldn’t be, and that these may be due to extra data added to the code that is causing bugs to appear in the application.

# 5. Report to the NOSSoft Management

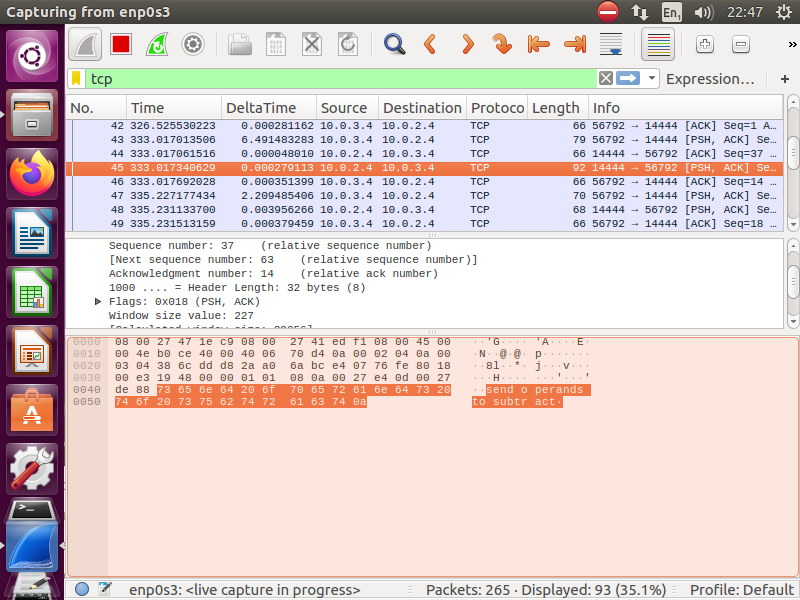
This section aims to clarify and expand upon the problems that I have been facing with the newer update. To quickly identify, the first problem lies in the swapping of certain functions, although there are two different groups of swapped functions and the origin of those problems differ with each group. The final problem lies in the multiplication function.

I’ll start by expanding on the first point, namely, that the addition and subtraction functions have been swapped with one another. The test performed, as seen with the protocol table above, involved inputting the numbers “3 1” as operands into both the addition and subtraction operations. There is strong evidence to suggest that this occurred on the server side, as beginning with a message exchanged from the server (10.0.2.4) to the client (10.0.3.4), information within the third row of the packet-content window began to differ from previous messages. This is reflected in the protocol table in which the same packet sends back the ‘operands to sub’ message to the client protocol. The proper function of both addition and subtraction operations should be reversed, with addition adding and subtraction subtracting numbers.

**Client #1 (Addition Operation)**



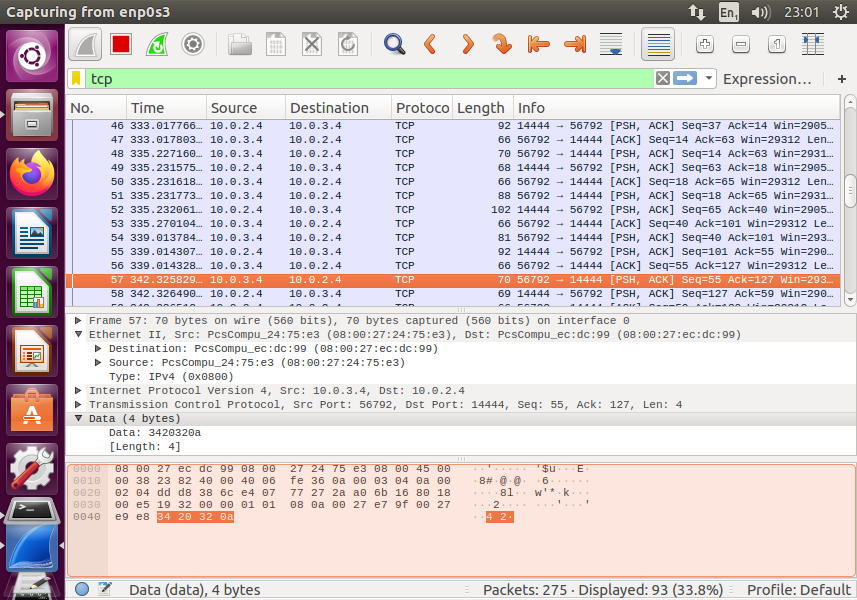
**Server #1 (Addition Operation)**



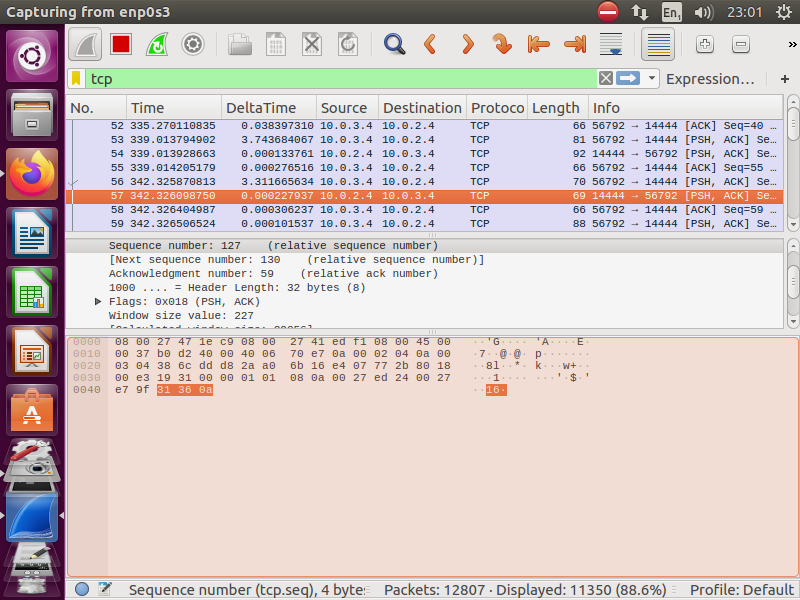
The second problem lies in two other operations being switched. These would be the quit and division operations, or options “0” and “4”. A test was performed in which the contents of each packet were analysed, much like the previous problem. This however, appears to be on the client side. As seen in the protocol table, when “4” is entered, a request for ‘endcomms’ is sent to the server and when “0” is entered, a request for ‘div operands’ is sent. This can pose a significant problem if not quickly fixed as the quit operation under a differently named function suggests an unusable application. As with the previous problem, the behaviour of each function should be switched but from the client code rather than the server.

The third problem is the multiplication function. When used, it does not swap functions like the others, but produces a completely different function entirely. It acts as an exponentiation where the first number is the base and the second is a power raised by. For this test, I entered “4 2”, the result being “16”. A second test of “10 2” with the result being “100” confirmed my suspicions. This problem occurs on the server’s side and appears to be interpreting the operands as an exponent rather than actual multiplication. Instead of this, when entering operands such as “4 2”, this should multiply and display “8”. Likewise “10 2” should display “20”.

**Client #1 (Multiplication Operation)**



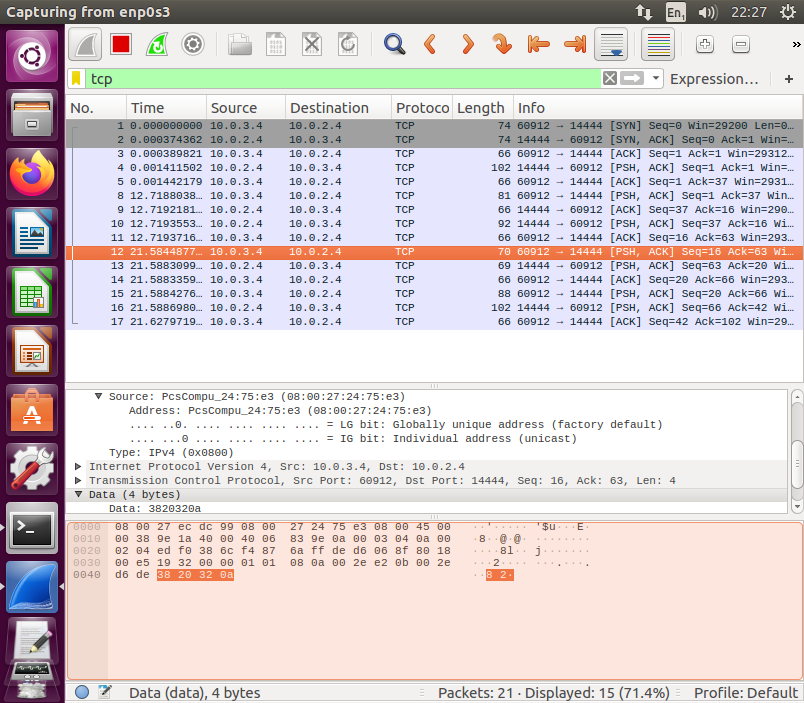
**Server #1 (Multiplication Operation)**



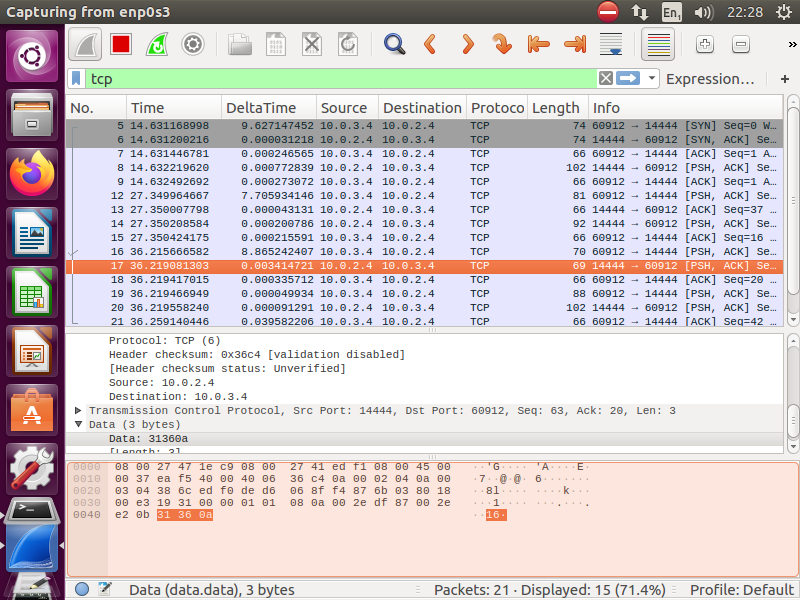
Extra hexadecimal characters and position changes on the 3rd line may have affected the output of the operations.

For comparison, this is how the same multiplication operation would have worked in the previous version of the application:

**Client#1 (Old Calc Multiplication Operation)**



**Server#1(Old Calc Multiplication Operation)**



**6. Conclusions**

Throughout this report, I have documented proof that my network was functioning correctly when using the app. This was done through the use of several tests, each which have helped to explain my knowledge and awareness of how data is exchanged through the network. I have also identified a list of problems with how I came across them as well as providing how they should work. These problems seem to stem from broken code and appear on both the server and client side depending on the problem. I hope that this may be enough to allay any doubts you may have had and persuade you to fix the current update, as it is imperative that a fix is implemented soon. As the system is now, I could not recommend it to the teaching faculty and students, but should the application return to a similar state as it was in its previous iteration, this opinion would change to reflect that. I hope you take this report into careful consideration as you move forward with the application.