Department of Computer Science

CS2005 Networks & Operating Systems Task 1

Academic Year 2024-25

Rooney Mosiana Nsasa

2315045

Table of Contents

1. Introduction

This is an extensive, formal report consisting of setting up and testing a simple network, trying to troubleshoot the problem that the Netsoft Management team have encountered, reason being due to the client server application update they have released (cafeClientUpdate and cafeServerUpdate) and not our network setup respectively; Below will include a document protocol for our client server application we initially installed and the one updated by Netsoft Management, followed by a brief overview of the report to the management team.

2. Test Network Documentation

The diagram below shows a network topology, which is simply an example of how the network is supposed to function:

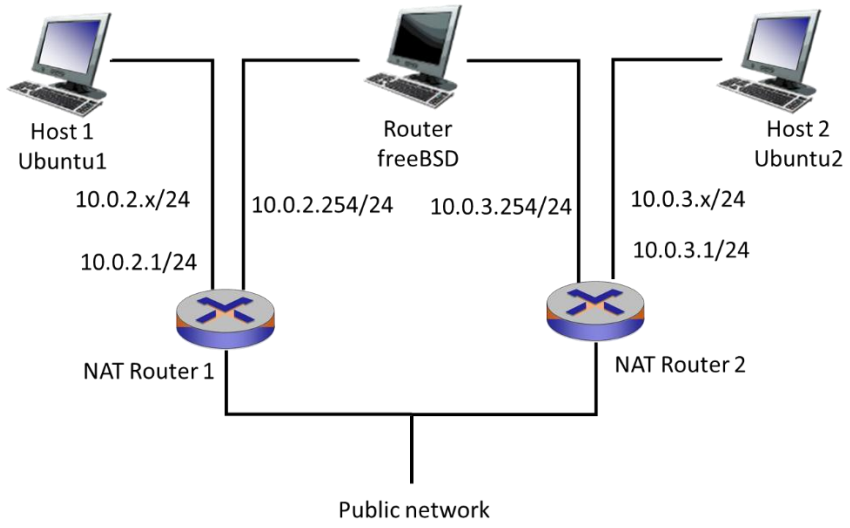


Figure 1: Test Network Configuration

Ifconfig

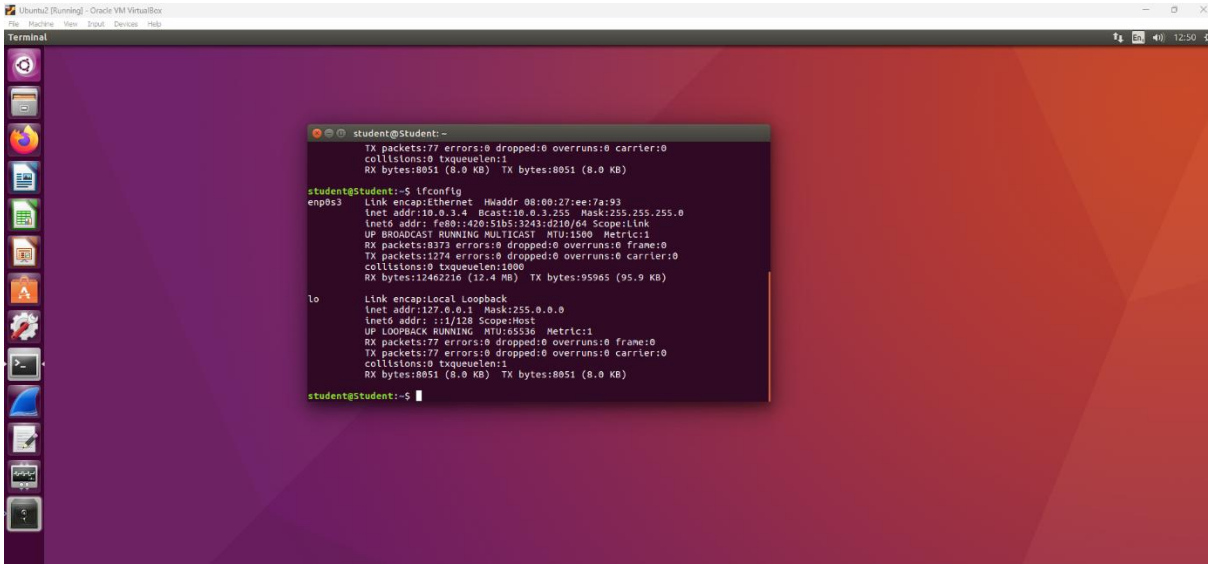
Ubuntu #1

```
student@Student:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:f0:97:fa
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a3d3:a3ac:8068:411b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:254 errors:0 dropped:0 overruns:0 frame:0
          TX packets:170 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:310198 (310.1 KB)  TX bytes:16196 (16.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:121 errors:0 dropped:0 overruns:0 frame:0
          TX packets:121 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11312 (11.3 KB)  TX bytes:11312 (11.3 KB)

student@Student:~$
```

Ubuntu #2



Netstat

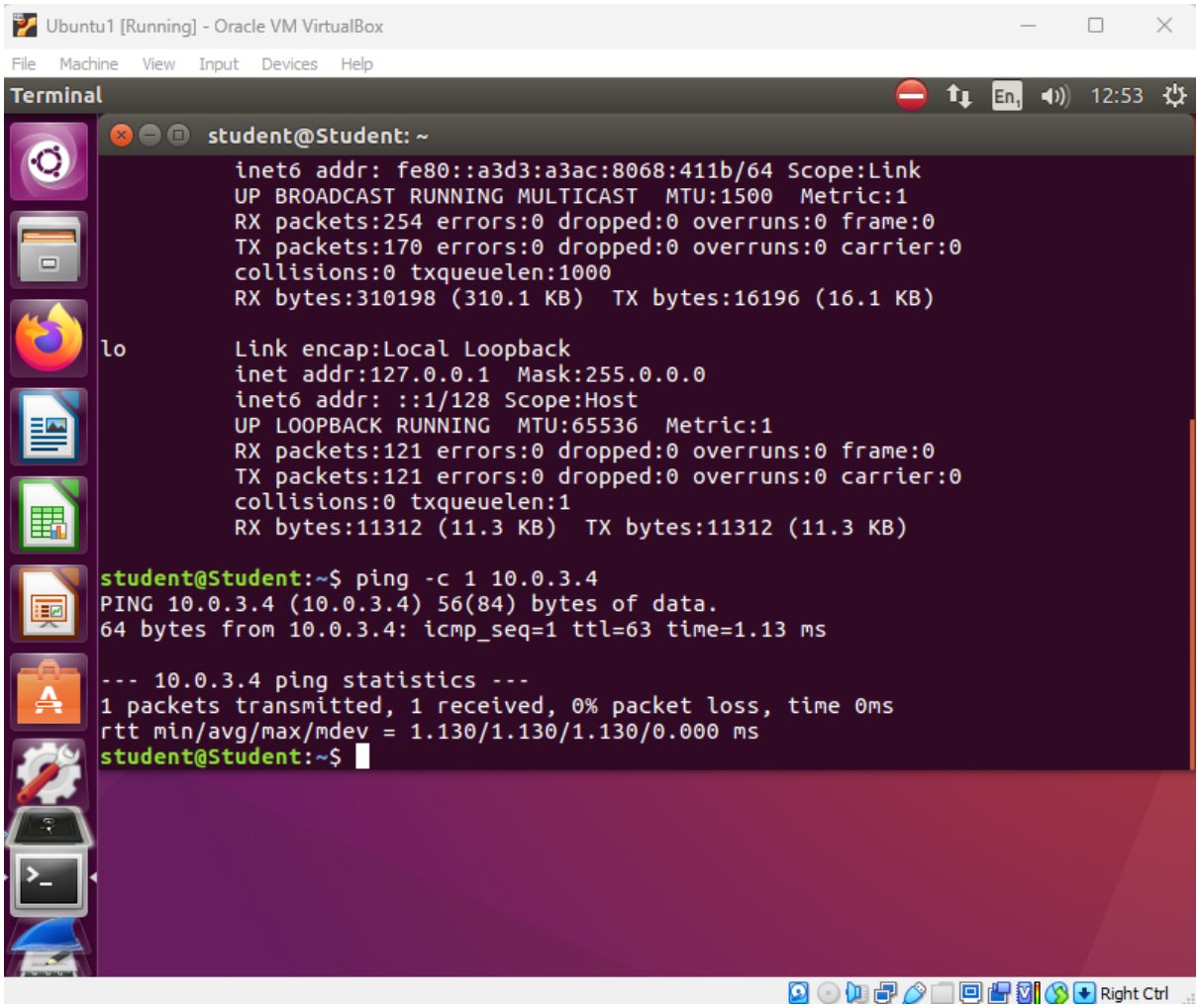
Ubuntu #1

```
student@Student:~$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         10.0.2.1        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
10.0.3.0        10.0.2.254      255.255.255.0   UG        0 0          0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
```

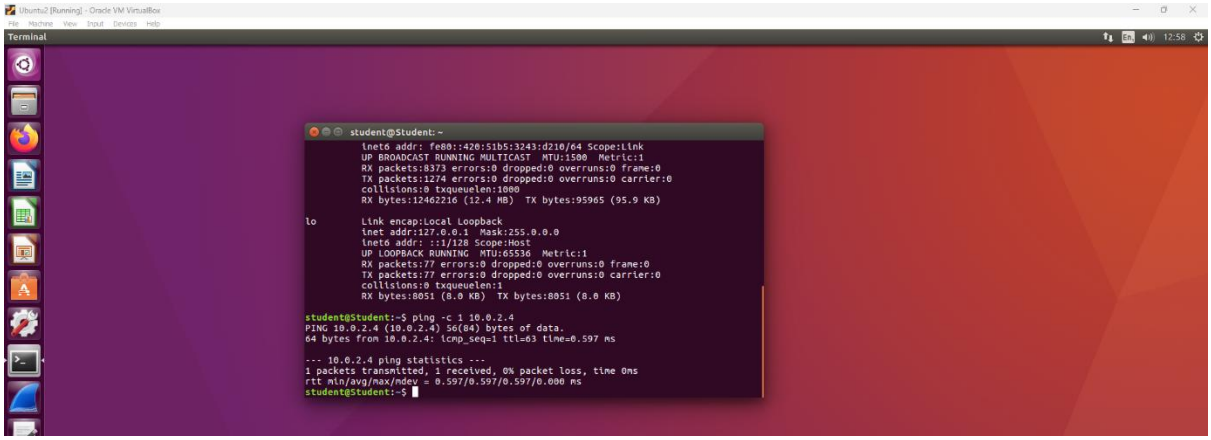
Ubuntu #2

Ping

Ubuntu #1

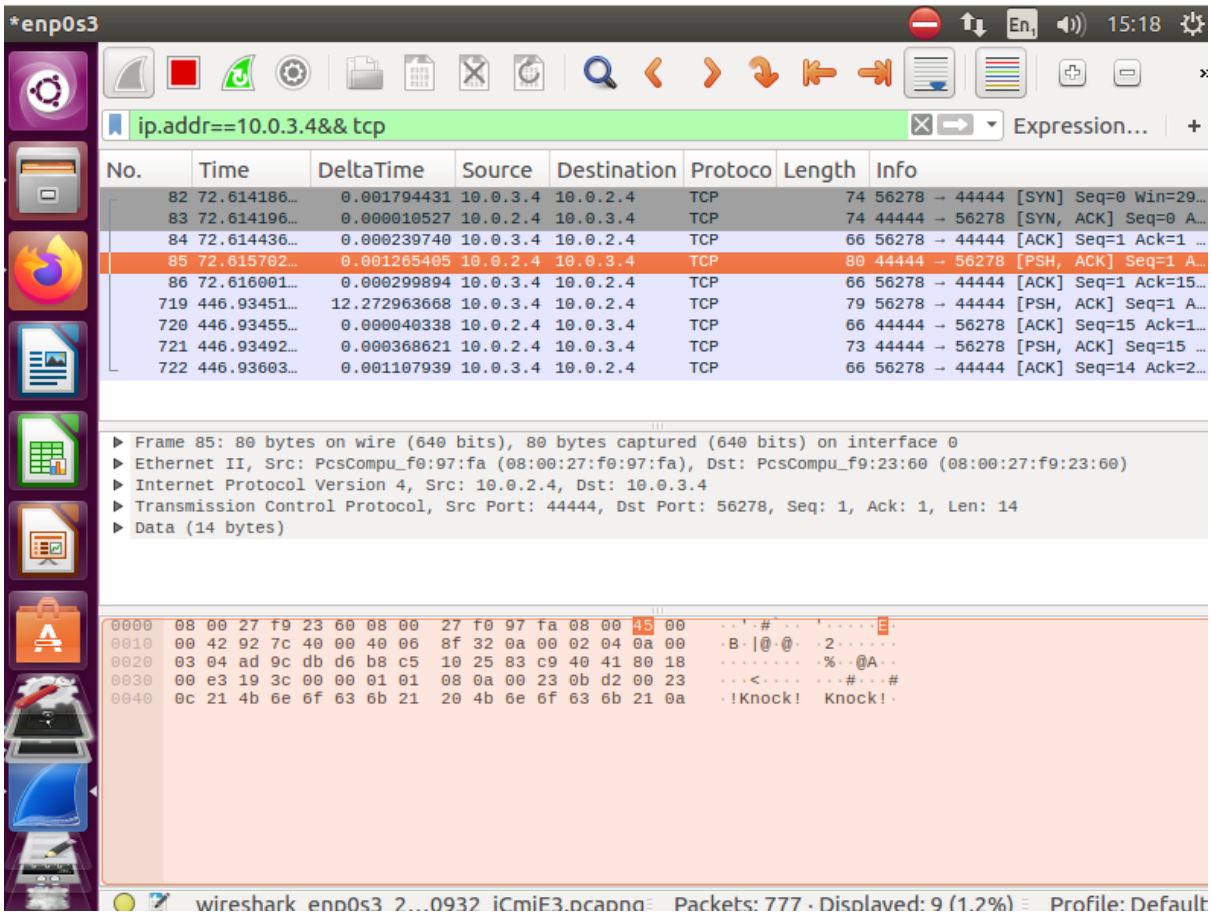


Ubuntu #2

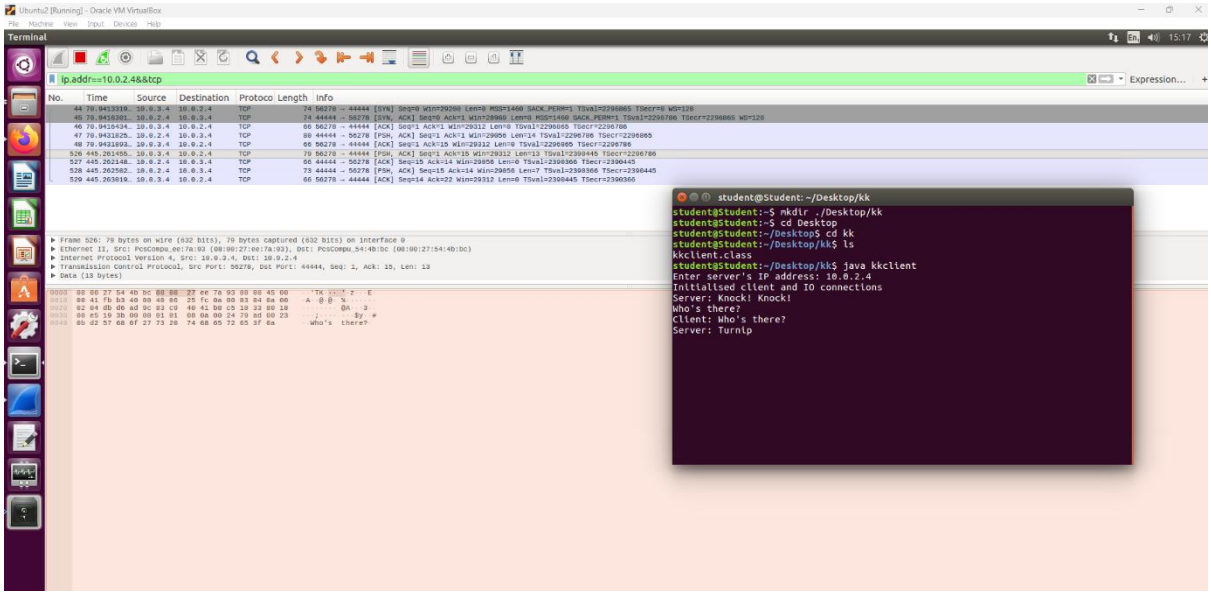


Wireshark

Ubuntu #1



Ubuntu #2


Ubuntu2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
15:17
ip.addr==10.0.2.4&&tcp
Expression...
No. Time Source Destination Protoco Length Info
44 70.9413319… 10.0.3.4 10.0.2.4 TCP 74 56278 → 44444 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2296065 TSecr=0 WS=128
45 70.9416301… 10.0.2.4 10.0.3.4 TCP 74 44444 → 56278 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2296706 TSecr=2296065 WS=128
46 70.9418434… 10.0.3.4 10.0.2.4 TCP 66 56278 → 44444 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2296065 TSecr=2296706
47 70.9431825… 10.0.2.4 10.0.3.4 TCP 80 44444 → 56278 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=14 TSval=2296706 TSecr=2296065
48 70.9431893… 10.0.3.4 10.0.2.4 TCP 66 56278 → 44444 [ACK] Seq=1 Ack=15 Win=29312 Len=0 TSval=2296065 TSecr=2296706
526 445.261455… 10.0.3.4 10.0.2.4 TCP 79 56278 → 44444 [PSH, ACK] Seq=1 Ack=15 Win=29312 Len=13 TSval=2300445 TSecr=2296706
527 445.262148… 10.0.2.4 10.0.3.4 TCP 66 44444 → 56278 [ACK] Seq=15 Ack=14 Win=29056 Len=0 TSval=2300366 TSecr=2300445
528 445.262582… 10.0.2.4 10.0.3.4 TCP 73 44444 → 56278 [PSH, ACK] Seq=15 Ack=14 Win=29056 Len=7 TSval=2300366 TSecr=2300445
529 445.263019… 10.0.3.4 10.0.2.4 TCP 66 56278 → 44444 [ACK] Seq=14 Ack=22 Win=29312 Len=0 TSval=2300445 TSecr=2300366
Frame 526: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0
Ethernet II, Src: PcsCompu_ee:7a:93 (08:00:27:ee:7a:93), Dst: PcsCompu_54:4b:bc (08:00:27:54:4b:bc)
Internet Protocol Version 4, Src: 10.0.3.4, Dst: 10.0.2.4
Transmission Control Protocol, Src Port: 56278, Dst Port: 44444, Seq: 1, Ack: 15, Len: 13
Data (13 bytes)
0000 08 00 27 54 4b bc 08 00 27 ee 7a 93 08 00 45 00
0010 00 41 fb b3 40 00 40 06 25 fc 0a 00 03 04 0a 00
0020 02 04 db d6 ad 9c 83 c9 40 41 b0 c5 10 22 80 18
0030 00 e5 19 3b 00 00 01 01 08 0a 00 24 79 8d 00 23
0040 6b d2 57 68 6f 27 73 20 74 68 65 72 65 3f 0a
student@Student: ~/Desktop/kk
student@Student:~$ mkdir ./Desktop/kk
student@Student:~$ cd Desktop
student@Student:~/Desktop$ cd kk
student@Student:~/Desktop/kk$ ls
kkclient.class
student@Student:~/Desktop/kk$ java kkclient
Enter server's IP address: 10.0.2.4
Initialised client and IO connections
Server: Knock! Knock!
Who's there?
Client: Who's there?
Server: Turnip

3. cafeClient and cafeServer Documentation

Here you will show the cafeClient and cafeServer protocol. To do this, get a copy of cafeClient and cafeServer from BBL. Run the client and the server as you have done in the Distributed Systems tutorial (on your test network). Using wireshark, systematically capture the protocol for all the transactions. Describe these in your report using a protocol table.

Basically you just need to add your protocol table. Nothing else (apart from a small amount of descriptive text).

	cafeClient		**cafeServer**
			[run cafeServer]
	[run cafeClient]		
			[accept cafeClient connection]
	WHILE NOT TERMINATED		WHILE NOT TERMINATED
			SEND "cafe server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "1"		
	SEND "Cola" TO cafeServer		
			RECEIVE "Cola" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	RECEIVE "Send quantity" FROM cafeServer		
	USER INPUT "1"		
	SEND "1" TO cafeServer		
			RECEIVE "1" FROM cafeClient
			SEND "Your new credit is 8.80" TO cafeClient
	READ "Your new credit is 8.80" FROM user		
	SEND "Next order please" TO cafeServer		
			RECEIVE "Next order please" FROM cafeClient
			SEND "Café server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "2"		
	SEND "Sandwich" TO cafeServer		RECEIVE "Sandwich" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	READ "Send quantity" FROM user		
	USER INPUT "1"		
	SEND "1" TO cafeServer		
			RECEIVE "1" FROM cafeClient
			SEND "Your new credit is 5.60" TO cafeClient

	READ “Your new credit is 8.80” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “3”		
	SEND “Crisps” TO cafeServer		RECEIVE “Crisps” FROM cafeClient
			SEND “Send quantity” TO cafeClient
	RECEIVE “Send quantity” FROM cafeServer		
	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “1” FROM cafeClient
			SEND “Your new credit is 5.10” TO cafeClient
	READ “Your new credit is 5.10” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “4”		
	SEND “Chocolate” TO cafeServer		RECEIVE “Chocolate” FROM cafeClient
			SEND “Send quantity” TO cafeClient
	RECEIVE “Send quantity” FROM cafeServer		
	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “1” FROM cafeClient
			SEND “Your new credit is 3.75” TO cafeClient
	READ “Your new credit is 3.75” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “1”		
	SEND “Cola” TO cafeServer		
			RECEIVE “Cola” FROM cafeClient
			SEND “Send quantity” TO cafeClient
	RECEIVE “Send quantity” FROM cafeServer		
	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “1” FROM cafeClient

			SEND “Your new credit is 2.55” TO cafeClient
	READ “Your new credit is 2.55” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “2”		
	SEND “Sandwich” TO cafeServer		RECEIVE “Sandwich” FROM cafeClient
			SEND “Send quantity” TO cafeClient
	READ “Send quantity” FROM user		
	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “1” FROM cafeClient
			SEND “You don’t have enough credit. Add credit and try again.” TO cafeClient
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “5”		
	SEND “Add credit” TO cafeServer		
			RECEIVE “Add credit” FROM cafeClient
			SEND “Send credit value” TO cafeClient
	READ “Send credit value” FROM user		
	USER INPUT “10.0”		
	SEND “10.0” TO cafeServer		
			RECEIVE “10.0” FROM cafeClient
			SEND “Your new credit is 12.55” TO cafeClient
	READ “Your new credit is 12.55” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “6”		
	SEND “See credit” TO cafeServer		
			RECEIVE “See credit” FROM cafeClient
			SEND “12.55” TO cafeClient
	READ “12.55” FROM user		
	SEND “Next order please” TO cafeServer		

			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	SEND “endcomms” TO cafeServerUpdate		
	[Terminate]		RECEIVE “endcomms” FROM cafeClientUpdate
			[Terminate]
	ENDWHILE		ENDWHILE

4. cafeClientUpdate and cafeServerUpdate Documentation

This section captures your cafeClientUpdate and cafeServerUpdate protocol. To do this, get a copy of the updated software from BBL. Run the client and the server as you have done in the Distributed Systems tutorial (on your test network). Using wireshark, systematically capture the protocol for all the transactions. Describe these in your report using a protocol table as well as any problem that you have identified.

Basically you just need to add your protocol table and a small amount of descriptive text that shows where you think things have gone wrong with the application and explain how you have tested the protocol.

Protocol table example:

	cafeClient		**cafeServer**
			[run cafeServer]
	[run cafeClient]		
			[accept cafeClient connection]
	WHILE NOT TERMINATED		WHILE NOT TERMINATED
			SEND "cafe server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "1"		
	SEND "Cola" TO cafeServer		
			RECEIVE "Cola" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	RECEIVE "Send quantity" FROM cafeServer		
	USER INPUT "1"		
	SEND "1" TO cafeServer		
			RECEIVE "2" FROM cafeClient
			SEND "Your new credit is 7.60" TO cafeClient
	READ "Your new credit is 7.60" FROM user		
	SEND "Next order please" TO cafeServer		Reads double the quantity, despite the user inputting 1
			RECEIVE "Next order please" FROM cafeClient
			SEND "Cafe server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "2"		
	SEND "Sandwich" TO cafeServer		RECEIVE "Crisps" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	READ "Send quantity" FROM user		Menu options 2 and 3 (Sandwich, Crisps) have been swapped around
	USER INPUT "1"		
	SEND "1" TO cafeServer		
			RECEIVE "1" FROM cafeClient

			SEND "Your new credit is 7.10" TO cafeClient
	READ "Your new credit is 7.10" FROM user		
	SEND "Next order please" TO cafeServer		
			RECEIVE "Next order please" FROM cafeClient
			SEND "Café server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "3"		
	SEND "Crisps" TO cafeServer		RECEIVE "Sandwich" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	RECEIVE "Send quantity" FROM cafeServer		
	USER INPUT "1"		Menu options 2 and 3 (Sandwich, Crisps) have been swapped around
	SEND "1" TO cafeServer		
			RECEIVE "1" FROM cafeClient
			SEND "Your new credit is 0.70" TO cafeClient
			Reads double the quantity, despite the user inputting 1
	READ "Your new credit is 0.70" FROM user		
	SEND "Next order please" TO cafeServer		
			RECEIVE "Next order please" FROM cafeClient
			SEND "Café server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "4"		
	SEND "Chocolate" TO cafeServer		RECEIVE "Chocolate" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	RECEIVE "Send quantity" FROM cafeServer		
	USER INPUT "1"		
	SEND "1" TO cafeServer		
			RECEIVE "1" FROM cafeClient
			SEND "Your new credit is 1.70" TO cafeClient
	READ "Your new credit is 1.70" FROM user		Selecting chocolate adds to the quantity, instead of deducting
	SEND "Next order please" TO cafeServer		
			RECEIVE "Next order please" FROM cafeClient
			SEND "Café server ready and waiting" TO cafeClient
	RECEIVE "cafe server ready and waiting" from cafeServer		
	READ "menuOption" FROM user		
	USER INPUT "1"		
	SEND "Cola" TO cafeServer		
			RECEIVE "Cola" FROM cafeClient
			SEND "Send quantity" TO cafeClient
	RECEIVE "Send quantity" FROM cafeServer		

	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “5” FROM cafeClient
			SEND “You don’t have enough credit. Add credit and try again.” TO cafeClient
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “5”		
	SEND “Add credit” TO cafeServer		
			RECEIVE “Add credit” FROM cafeClient
			SEND “Send credit value” TO cafeClient
	READ “Send credit value” FROM user		
	USER INPUT “10.0”		
	SEND “10.0” TO cafeServer		
			RECEIVE “10.0” FROM cafeClient
			SEND “Your new credit is 11.70” TO cafeClient
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “5”		
	SEND “Add credit” TO cafeServer		
			RECEIVE “Add credit” FROM cafeClient
			SEND “Send credit value” TO cafeClient
	READ “Add credit” FROM user		
	USER INPUT “10.0”		
	SEND “10.0” TO cafeServer		
			RECEIVE “10.0” FROM cafeClient
			SEND “Your new credit is 11.70” TO cafeClient
	READ “Your new credit is 11.70” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “3”		
	SEND “Crisps” TO cafeServer		
			RECEIVE “Sandwich” FROM cafeClient
			SEND “Send quantity” TO cafeClient
	RECEIVE “Send quantity” FROM cafeServer		

	USER INPUT “1”		
	SEND “1” TO cafeServer		
			RECEIVE “1” FROM cafeClient
			SEND “Your new credit is 5.30” TO cafeClient
	RECEIVE “cafe server ready and waiting” from cafeServer		
	READ “menuOption” FROM user		
	USER INPUT “6”		
	SEND “See credit” TO cafeServer		
			RECEIVE “See credit” FROM cafeClient
			SEND “5.299999999” TO cafeClient
	READ “5.299999” FROM user		
	SEND “Next order please” TO cafeServer		
			RECEIVE “Next order please” FROM cafeClient
			SEND “Café server ready and waiting” TO cafeClient
	SEND “endcomms” TO cafeServerUpdate		
	[Terminate]		RECEIVE “endcomms” FROM cafeClientUpdate
			[Terminate]
	ENDWHILE		ENDWHILE

5. Report to the NetSoft Management

This is a shortly concluded report analyzing the two protocols between cafeServer and cafeServerUpdate and after testing, several issues have been found, particularly within the cafeUpdate protocol. Below is a detailed list of the problems, how they were identified and suggestions to fix the current situation:

Problem 1: Quantity Error

- The first issue I encountered when running the cafeUpdate protocol was that the server read double the quantity entered by the client; For example, if the user inputs a quantity of 1 the server will read it as 2. This error was observed through a careful comparison of both tables, where the user sending "1" (SEND “1” TO cafeServer), the server responds with doubled quantities (e.g., RECEIVE “2” FROM cafeClient).

Problem 2: Menu Items Swapped

- The second issue I encountered was that menu items 2 and 3 (Sandwich, Crisps) have seemingly been swapped around, this is a client application issue where the menu-to-item mapping is incorrect. This is evident within the cafeUpdate protocol, where when the user selects menu option "2" (Sandwich), the server processes it as "Crisps" (RECEIVE “Crisps” FROM cafeClient) and vice versa. This issue appears consistently, and this could be an app issue where the menu item listing is incorrect so a way to resolve this issue is by updating it to make sure that the protocol is running correctly with the right items listed.

Problem 3: Incorrect Credit Deduction

- The third problem encountered is that certain items have the credit incorrectly updated after the server has read the user input. For example, within the cafeUpdate protocol selecting “Chocolate” adds to the credit instead of deducting it in addition to even though the menu items were swapped, selecting “Sandwich” causes the server to correctly read the quantity depending on the user input however it will deduce double the quantity instead of the quantity entered by the user. This could likely be because of an issue within the server application causing the item to be added instead of subtracted, so possibly correcting the server logic so that it deducts the items accordingly could solve the problem.

Problem 4: Incorrect Credit Display

- Another issue encountered was when the user selected option 6 (See credit), the displayed value was showing differently from the actual credit value. This is evident within the protocol table as the user would see a value of 5.299999999 instead of it being rounded to 5.30 probably due to an issue in the server application, where the credit value is improperly formatted or retrieved.

6. Conclusions

The analysis of the *cafeUpdate* protocol has revealed several critical issues affecting quantity processing, menu item mapping, credit deduction, and credit display accuracy. These errors, whether originating from the client or server application, impact the overall reliability of the system. Addressing these problems will require thorough debugging and updates to both data transmission and processing logic. Implementing precise data handling, correcting mapping inconsistencies, refining credit transaction calculations, and formatting numerical outputs properly will ensure a more stable and user-friendly experience. With these adjustments, the *cafeUpdate* protocol can function as intended, providing accurate results and seamless interactions for users.