Brunel University London

College of Engineering, Design and Physical Science
Electronic and Computer Engineering

Assignment
Java Testing and Measuring

Distributed Computing Systems Engineering Msc

Author: Christoph Gschrey

Lab-Partner: Matthias Gebert

Date: 22. September 2017

Supervisor: Prof. Dr. Peter Väterlein

College of Engineering, Design and Physical Sciences

Department of Engineering and Design

ASSIGNMENT SUBMISSION FORM

Please note: that no course work will be accepted without this cover sheet.

Please ensure: that you keep a copy of work submitted and retain your receipt in case of query.

Student Number:	**SPO ID Number** (Office use only):	
Course:		**Level:**

MODULE	
Module Code:	**Module Title:**
Lab / Assignment:	**Deadline:**
Lab group (if applicable):	**Date Stamp** (Office use only):
Academic Responsible:	
Administrator:	

Please note: that detailed feedback will be provided on a feedback form.

✂.....

RECEIPT SECTION (Office Copy)	
Student Number:	**SPO ID Number** (Office use only):
Student First Name:	**Student Last Name:**
Module Code:	**Module Title:**
Lab / Assignment:	
Lab group (if applicable):	**Deadline:**
Academic Responsible:	**Number of Days late:**

DECLARATION	
I have read and I understand the guidelines on plagiarism and cheating in the Handbook and I certify that my contribution to this report fully complies with these guidelines. I confirm that I have kept a copy of my work and that I have not lent my work to any other students.	
Signed:	**Date Stamp** (Office use only):

✂.....

RECEIPT SECTION (Student Copy)	
Student Number:	**Student Name**:
Lab / Assignment:	
Lab group (if applicable):	**Module Title:**
Academic Responsible:	**Deadline:**
Module Code:	**Date Stamp** (Office use only):

The University penalty system will be applied to any work submitted late.

IMPORTANT: You **MUST** keep this receipt in a safe place as you may be asked to produce it at any time as proof of submission of the assignment. Please submit this form with the assignment attached to theDepartment of Design Education Office in the Michael Sterling Building, room MCST 055.

Contents

1 Introduction **1**

2 IP/ICMP analysis **2**

2.1 Node configuration . . . 4
2.2 Subnet internal IP Destination . . . 4
2.2.1 a) Basic PING command . . . 4
2.2.2 b) PING command with large data package . . . 4
2.2.3 c) PING command with 'don't fragment' flag . . . 4
2.3 Subnet external IP Destination . . . 4
2.3.1 d) Basic PING command with destination in another subnet . . . 4
2.3.2 e) PING command with reduced 'time to live' . . . 4
2.3.3 f) PING command with timestamps . . . 4
2.4 ARP analysis . . . 4
2.4.1 a) Deleting the ARP cache . . . 4
2.4.2 b) Shutting down one PC . . . 4
2.4.3 c) Reconnect after Reboot . . . 4
2.5 IP multicast addressing . . . 4

3 TCP analysis **5**

3.1 Traffic generator handling . . . 5
3.1.1 Connection establishment . . . 5
3.1.2 Data transfer . . . 5
3.1.3 Connection release . . . 5
3.2 Simple TCP Communication . . . 5
3.3 TCP flow control . . . 5
3.4 TCP transmission error recovery/abort . . . 5
3.5 TCP protocol errors (synchronization errors) . . . 5

4 IPv6/ICMPv6 analysis **6**

4.1 Node configuration . . . 6
4.1.1 IPv4 and IPv6 configuration . . . 6
4.1.2 interfaces for IPv6 . . . 6
4.2 PING commands . . . 6
4.2.1 a) Basic ICMPv6 PING command . . . 6
4.2.2 b) ICMPv6 PING command with large data package . . . 6

4.2.3 c) Rebooting PC 6
4.2.4 d) Enforcing Neighbor discovery 6
4.2.5 e) ICMPv6 PING command with destination in another subnet 6
4.2.6 f) PING to a remote tunnel end 6

5 Conclusion **7**

Bibliography **8**

1 Introduction

The following reports refers to the Computer network assignment and is structured into three parts. The first part's topic is an analysis of the network protocols ICMP and IP (both v4), while the seconds part covers the exercises related to TCP. The final chapter describes the exercises for the new versions of ICMP and IP (v6). These exercises were done together with my lab-partner Antonio Parotta.

2 IP/ICMP analysis

In this first part of the laboratory the program Wireshark was used to capture and analyse packages of different network protocols. The traffic was generated by PING-commands to send the observable packages from one lab PC to another. The following network protocols were analyzed:

- Internet Protocol version 4 (IPv4)
- Internet Control Message Protocol version 4 (ICMPv4)
- Address Resolution Protocol (ARP)
- Carrier Sense Multiple Access/Collision Detection (CSMA/CD)

To understand how these protocols work and to be able to explain how they behave in different situations, having a look on the protocol's headers is necessary.

The PING-commands generate packages consisting of different protocol headers and transferable data. Each Ping is transformed into an Ethernet frame containing the IP and ICMP headers. Table **??** is an representation of the basic ICMP Header while Table **??** shows the header for the echo request/reply packages that can be observed via Wireshark when executing the PING-commands.

Table **??** shows the header for the Internet Protocol v4. Noteable here are the entered destination address as well as the source address of the sender. The Time to Live is also an important segment of the header, which will be significant later on for an specific PING-Command. IP provides the possibility to specify options for the transfered packet. This will also be used in one of the PING-Commands.

Table **??** shows the abstract Ethernet II frame. This frame contains the MAC-addresses for source and destination, a type segment as well as the checksum for the frame. Interesting here is the Payload field. This segments contains the headers for ICMP and IP as well as the

bits	0-7	8-15	16-23	24-31
bytes	1	2	3	4
offset 0	Type	Code	Checksum	
offset 32	Data			

Table 2.1: ICMP header

bits	0-7	8-15	16-23	24-31
bytes	1	2	3	4
offset 0	Type	Code	Checksum	
offset 32	Identifier		Sequence Number	
offset 64	data			

Table 2.2: ICMP type 8 echo request/reply packet

bits	0-3	4-7	8-11	12-15	16-18	19-23	24-27	28-31
bytes	1	2	3	4	5	6	7	8
offset 0	Version	IHL	Type of Service		Total Length			
offset 32	Identification				Flags	Fragment Offset		
offset 64	Time to Live		Protocol		Header Checksum			
offset 96	Source Address							
offset 128	Destination Address							
offset 160	Options							

Table 2.3: IPv4 Header

transferable data. The maximal size for this segment is 1500 bytes for one packet. But because is must contain the headers for each networking protocol (ICMP and IP), it can't be fully occupied by transferable data. This is why the Maximum Transmission Unit (MTU) is smaller. It is only 1472 bytes, because the size of the headers must be subtracted from the payload field.
MTU = Payload − IP Header − ICMP Header
1500 byte − 20 byte − 8 byte = 1472 byte

Size in bit	24	24	8	184-6000	16
Size in byte	6	6	2	46 - 1500	4
Frame segments	Destination Address	Source Address	Type	Payload (Data)	FCS

Table 2.4: Ethernet II frame

2.1 Node configuration

2.2 Subnet internal IP Destination

2.2.1 a) Basic PING command

2.2.2 b) PING command with large data package

2.2.3 c) PING command with 'don't fragment' flag

2.3 Subnet external IP Destination

2.3.1 d) Basic PING command with destination in another subnet

2.3.2 e) PING command with reduced 'time to live'

2.3.3 f) PING command with timestamps

2.4 ARP analysis

2.4.1 a) Deleting the ARP cache

2.4.2 b) Shutting down one PC

2.4.3 c) Reconnect after Reboot

2.5 IP multicast addressing

3 TCP analysis

3.1 Traffic generator handling

3.1.1 Connection establishment

3.1.2 Data transfer

3.1.3 Connection release

3.2 Simple TCP Communication

3.3 TCP flow control

3.4 TCP transmission error recovery/abort

3.5 TCP protocol errors (synchronization errors)

4 IPv6/ICMPv6 analysis

4.1 Node configuration

4.1.1 IPv4 and IPv6 configuration

4.1.2 interfaces for IPv6

4.2 PING commands

4.2.1 a) Basic ICMPv6 PING command

4.2.2 b) ICMPv6 PING command with large data package

4.2.3 c) Rebooting PC

4.2.4 d) Enforcing Neighbor discovery

4.2.5 e) ICMPv6 PING command with destination in another subnet

4.2.6 f) PING to a remote tunnel end

5 Conclusion

Bibliography

[Bec04] Kent Beck. *Test-Driven development by Example*. Pearson, 2004. ISBN: 8131715957.

[WMV03] Laurie Williams, E. Michael Maximilien, and Mladen Vouk. "Test-Driven Development as a Defect-Reduction Practice". In: *14th IEEE International Symposium on Software Reliability Engineering ISSRE 2003*. Denver, Colorado: IEEE, 2003. URL: `http://ieeexplore.ieee.org/document/1251029/` (visited on 10/01/2016).