Introduction to Web Science

Assignment 2

Jun Sun
junsun@uni-koblenz.de

Iryna Dubrovska
idubrovska@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: November 16th, 2021, 24:00 CET
Tutorial on: November 18th, 2021, 16:00 CET

This assignment focuses on the concepts of 1) **Internet Protocol** 2) **Transmission Control Protocol**, 3) **Domain Name System** and 4) **Programming in Python**. Some of the tasks may require you to do additional research extending the lecture. Please keep the citation rules in mind.

For all the assignment questions that require you to write a code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Date: 16/11/2021

Team Name: Boehm

Abhinav Ralhan (abhinavr8@uni-koblenz.de)
Fatima Akram (fatimaakram9396@uni-koblenz.de)
Hammad Ahmed (hammadahmed@uni-koblenz.de)
Vishal Vidhani (vvidhani@uni-koblenz.de)

1 Python Programming. Client-Server (15 Points)

Write a simple TCP-based client-server program which does the following:
1. A server is waiting for a client to connect at some defined port (listening),

```
Waiting...
```

2. The client connects to the server's port and gets a message from the server. e.g.

```
b'Thank you for connecting'
```

3. At the same time, a server gets information about the connection (the server accepts and displays information from the client including client's IP address and its (ephemeral) port).

```
Waiting...
Host Air-Iryna.fritz.box established connection with ('192.168.178.28', 61642)
```

You may use **socket** library for this task.

Solution:
Client – Side Code:

```
#!/usr/bin/env python

# coding: utf-8

import socket


client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(('localhost', 12345))

try:

  while True:

    data=client.recv(1024)

    print(str(data))

    break

except KeyboardInterrupt:

  print("Exited by user")

client.close()
```

Server-Side Code:

```
#!/usr/bin/env python

# coding: utf-8

import socket

# Create a TCP/IP socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port where the server is listening

server_address = ('localhost', 12345)

sock.bind(server_address)

sock.listen(1)

while True:

    print("Waiting...")

    client,addr = sock.accept()

    print("HOST " + socket.getfqdn() + " Established connection with ",addr)

    try:

        client.send(bytes('Thank you for connecting','utf-8'))

    except:

        print("Exited by User")

    client.close()

    break

sock.close()
```

2 Questions (20 points)

2.1. True or false. If false, please provide the right answer: (8 points)

1. A passive open on a server side is a result of a 3-step handshake. (**True**)

2. UDP prioritises accuracy of delivery over timely delivery. (**False**)
Reason: UDP prioritises timely delivery over accuracy. TCP prioritise s accuracy over timeliness.

3. TCP divides data to be sent into chunks and adds a TCP header to each of them. These structures are called TCP segments. A TCP segment then encapsulates IP datagram to proceed with exchanging with peers. (**True**)

4. In solicited TCP/IP traffic specifying ports is not required. (**False**)
Reason: Port number is always required to recognize the application in which need to send back packet/data.

5. Checksum is used to assemble packages to files in the end of transaction. (**False**)
Reason: checksum is a simple error detection mechanism for checking data integrity not assembling the data in order.

6. TCP Maximum Segment Size, if different from the default value, is specified during TCP handshake session. (**True**)

7. UDP cannot handle flow control but can handle congestion. (**False**)
Reason: UDP does not do flow control or congestion control. If that Is required with UDP, we can use ToU, a reliable transport protocol on top of UDP.

8. Erroneous packets, if occur in transmission, are retransmitted by TCP and discarded by UDP. (**True**)

2.2. Flow control, error handling, DNS records (12 points)

1. Which role flow control plays in avoiding link layer protocols overload and network slowdown? Name different forms of flow control and briefly explain one of them. (4 points)

Solution:

On the internet there are large files that need to be transferred, but the largest packets cannot be transferred in one go. They are split into many different packets. This creates problems when many people are transferring data across multiple devices.

There are three main problems in flow control:

a. Receiver must reassemble the received frames in correct order. This can be solved by using sequence numbers. It is also called 3-way handshake.

b. Converting packages into respective files from each sender. This can be solved by using port numbers.

c. Network speed is faster at sender's end and slow at receiver's might be slower. This means that there can be an overload of packets at the receiver. This can be solved by the sliding window approach to maintain flow control.

Sliding window approach:
Transmission control protocol (TCP) follows this approach of sliding window protocol in order to maintain flow of packets.
In the sliding window protocol, only a select number of bits can be sent to the receiver at one point of time. Until those bits are processed and then acknowledged (using ACK) as being completely processed, it does not allow the receiver to send more bits.

2. How are errors (missing segments, corrupted segments) detected and handled by TCP? Name and explain main mechanisms. (4 points)

Solution:

TCP handles different types of error detection problems (missing segments, corrupt segments) in the following ways:

Checksum: In a corrupted segmented, checksum field is used to identify data corruption in segments. Usually, the segment is considered as information loss by TCP.

Acknowledgement: Whenever a segment of data is delivered, the receiver sends an ACK signal back to the sender to confirm the data has been received.

Retransmission: Sometimes when these problems are present with segments, the same segments are retransmitted again.

3. Imagine that you need to introduce three additional subdomains to your domain example.com (e.g., www.example.com, mail.example.com, maps.example.com) but DNS server only allows for one record. As a solution, the record in a DNS zone is constructed as *.example.com. Please explain the concept of such records and what disadvantages it can have. (4 points)

Solution:

In Domain Name System (DNS), there are multiple ways to create different components in a website server. A subdomain is a new website extending from the base domain. Subdomain is a way to create multiple websites within the same domain while at the same time maintaining hierarchy amongst it. A subdomain cannot exist without a base domain.

For example, this is a very useful approach while building websites for one university but in different locations. *.university.edu can be the base domain while * can be replaced with the campus location.

Disadvantages of subdomains include maintaining all different subdomain websites while making updates to these websites is a larger task in comparison to making changes to just one domain.

3 IP Header Checksum (10 points)

Below is an IP header (IPv4) from an IP packet received at destination.

```
4500 0044 1c46 4000 4011 b2e6 a27d 1382 ac1E 0406
```

1. Find out, what protocol is specified in the header. (1 point)

Ans: The protocol bits will be in 4011 Hex, so we will convert this hexadecimal to binary.

```
    4          0          1            1
 8 4 2 1    8 4 2 1    8 4 2 1      8 4 2 1
 0 1 0 0    0 0 0 0    0 0 0 1      0 0 0 1
```

Binary number: 0100 0000 0001 0001

so 0001 0001 is the protocol bits, so we convert it to the decimal

10001 = $(1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$ = > 16 + 0 + 0 + 0 + 1 = 17
So, **17 is UPD protocol**.

2. Find out, what checksum is specified in the header. (1 point)

Ans: The checksum is the 16 bits of the IP header. So, IP header the check sum is **b2e6**

```
    b          2          e            6             } b => 11, e=>14
 8 4 2 1    8 4 2 1    8 4 2 1      8 4 2 1
 1 0 1 1    0 0 1 0    1 1 1 0      0 1 1 0
```

Checksum binary bits are: **1011 0010 1110 0110**

3. Find out, whether the data that was transmitted is corrupted. In other words, find out whether the checksum is correct. (8 points)
You may do it by translating the hexadecimal numbers to binary to perform the calculation or you can keep it in hexadecimal space. Please, indicate each step of your calculations and its respected outcome.

Ans: To calculate the checksum of IP header, we will add each 16 bits with each other except the checksum bits because it will be zero at the destination end.

```
      4 5 0 0
    + 0 0 4 4
_________________
      4 5 4 4          -> 1st result
```

```
   4 5 4 4
 + 1 C 4 6
___________
   6 1 8 A        -> 2nd result

   6 1 8 A
 + 4 0 0 0
___________
   A 1 8 A        -> 3rd result


   A 1 8 A
 + 4 0 1 1
___________
   E 1 9 B        -> 4th result


   B 2 E 6        -> we will not add this 16 bit hexadecimal, as it is the checksum send
                  by the sender and we are calculating the checksum at the
                  destination end.


   E 1 9 B
 + A 2 7 D
___________
 1 8 4 1 8        -> 5th result, as there's a carry of 1, so we will add this carry to the
                     result to keep it 16 bits in next step.


   8 4 1 8
 +       1
___________
   8 4 1 9      -> 5Th result, after additional carry removed.


   8 4 1 9
 + 1 3 8 2
___________
   9 7 9 B        -> 6th result


   9 7 9 B
 + A C 1 E
___________
 1 4 3 B 9        -> 7th, again carry so we add this result with carry to keep it in 16 bits
```

```
     4 3 B 9
  +        1
________________
     4 3 B A          -> 7th result, after additional carry removed.


     4 3 B A
  +  0 4 0 6
________________
     4 7 C 0          -> 8th result.
```

So, the final sum is **4 7 C 0**. Now we will take the 1's complement of this Hexadecimal. Let's convert it to the binary first.

```
        4         7         C          0          } C => 12
     8 4 2 1   8 4 2 1   8 4 2 1    8 4 2 1
     0 1 0 0   0 1 1 1   1 1 0 0    0 0 0 0
```

Binary of 47c0 is 0100 0111 1100 0000

1's complement of the above binary is: 1011 1000 0011 1111 (B83F) is the final check sum at the destination end.

Let's compare the check sum with the given check sum in the header packet.

B 2 E 6 -> 1011 0010 1110 0110
B 8 3 F -> 1011 1000 0011 1111

So, the check sums are not equal. Therefore, the data is corrupted.

Important Notes

SUBMISSION

- Solutions have to be submitted to the SVN repository. Use the directory name groupname/assignment2/ in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the SVN repository. Upload *all* .py files of your program! Use UTF-8 as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistentindentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This pdfLaTeX template was adapted by Jun Sun based on the LuaLaTeX version by Lukas Schmelzeisen.

LaTeX

Use pdflatex assignment_X.tex to build your PDF.