COMP 445 – Theoretical Assignment 2 (TA1)
Winter 2018

Tristan Glatard
tristan.glatard@concordia.ca

Concordia University
Department of Computer Science and Software Engineering

Instructions

- Please submit your assignment as a pdf file on Moodle. The name of the pdf file must contain your name and student id.
- All questions will receive equal points.
- Each question may have zero, one, or more than one correct choices.
- Partial answers will not receive any point.
- Blank answers (no answer) will not be penalized.

Student ID: .....

First Name / Last Name: .....

Signature: .....

Transport Layer

Q1: Among the following services, which ones are provided by UDP?

a) ☐ Congestion control

b) ☐ Flow control

c) ☐ Reliable data transfer

d) ☐ Bandwidth reservation

None of these services are provided by UDP. See slide 16.

Q2: An application may choose to transmit data using UDP rather than just IP (the network-level protocol) when it needs:

a) ☐ High throughput

b) ☑ Multiplexing / de-multiplexing

c) ☐ Security

d) ☐ Connection management (establishment, teardown)

Multiplexing and de-multiplexing of datagrams through sockets is the main service provided by UDP, through port numbers. UDP doesn't provide security or connection management. UDP would provide a high throughput compared to TCP *but not compared to only IP.*

Q3: TCP acknowledgments arrive with RTT values of 29, 31 and 32ms. What is the new estimated RTT value after the third acknowledgement was received, taking an initial estimated RTT of 31ms and $\alpha = 0.125$?

a) ☐ 32ms

b) ☑ 30.9ms

c) ☐ 31.8ms

d) ☐ 31.1ms

See slide 62 in Chapter 3. EstimatedRTT $= (1-\alpha)$EstimatedRTT$+\alpha$SampleRTT.

- *Before the first ACK is received:* EstimatedRTT*=31ms.*
- *After the first ACK is received:* EstimatedRTT*=(1-0.125)*31+0.125*29=30.75ms*
- *After the second ACK is received:* EstimatedRTT*=(1-0.125)*30.75+0.125*31=30.78ms*
- *After the third ACK is received:* EstimatedRTT*=(1-0.125)*30.78+0.125*32=30.93ms*

Q4: Two non-duplicate ACKs are received while a TCP sender is in Slow Start mode with cwnd=1KB, ssthresh=64KB and MSS=1KB. What is the state of the TCP sender after the second ACK is received?

a) ☑ Slow Start

b) ☐ Congestion Avoidance

c) ☐ Fast Recovery

d) ☐ SYN sent

See slide 101 in Chapter 3. In Slow Start mode, 1MSS is added to cwnd every time a non-duplicate ACK is received. In addition, state changes to Congestion Avoidance when cwnd reaches the slow-start threshold (ssthresh). In our situation:

- *After the first ACK is received: cwnd=2KB; cwnd <ssthresh; state remains Slow Start.*
- *After the second ACK is received: cwnd=3KB; cwnd <sstresh; state remains Slow Start.*

Q5: Assuming the same initial state and sequence of events as in the previous question, what will be the value of the cwnd variable (size of the congestion window) after the second ACK is received?

a) ☐ 1KB

b) ☐ 2KB

c) ☑ 3KB

d) ☐ 4KB

See answer to the previous question.

Q6: The content below was captured using Wireshark:

```
▸ Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▸ Ethernet II, Src: IntelCor_50:80:98 (f0:d5:bf:50:80:98), Dst: mynetwork (f0:82:61:f7:5e:80)
▸ Internet Protocol Version 4, Src: 192.168.2.111 (192.168.2.111), Dst: ec2-107-23-96-9.compute-1.amazonaws.com (107.23.96.9)
▾ Transmission Control Protocol, Src Port: 45674 (45674), Dst Port: https (443), Seq: 2630206945, Ack: 3725581375, Len: 0
    Source Port: 45674 (45674)
    Destination Port: https (443)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 2630206945
    Acknowledgment number: 3725581375
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x010 (ACK)
    Window size value: 755
    [Calculated window size: 755]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xd13e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
```

```
0000  f0 82 61 f7 5e 80 f0 d5  bf 50 80 98 08 00 45 00   ..a.^... .P....E.
0010  00 34 8c 2a 40 00 40 06  20 62 c0 a8 02 6f 6b 17   .4.*@.@.  b...ok.
0020  60 09 b2 6a 01 bb 9c c5  c5 e1 de 0f dc 3f 80 10   `..j.... .....?..
0030  02 f3 d1 3e 00 00 01 01  08 0a 40 0f 7e e4 0b 6e   ...>.... ..@.~..n
0040  78 d5                                             x.
```

This trace contains:

a) ☑ A TCP segment that contains an acknowledgment

b) ☐ A UDP segment that contains an acknowledgment

c) ☐ A TCP segment that contains an HTTP message

d) ☐ An HTTP message that contains a TCP segment

The trace shows a TCP segment (see blue highlight). The segment contains an acknowledgment, as shown by the flags. b) The trace doesn't show any UDP segment (and it couldn't, as a packet couldn't use both UDP and TCP). c) Although a TCP segment might contain an HTTP message, it is not the case here: the length of the segment is 0, which means that the segment doesn't contain any data (it's just an ACK). d) An HTTP message may not contain a TCP segment since HTTP is an application-level protocol and TCP is transport-level.

Q7: Among the following mechanisms, which one(s) can be used to provide reliable data transfer?

a) ☑ Checksums

b) ☑ Timeouts

c) ☑ Sequence numbers

d) ☑ Acknowledgments

All these mechanisms may be used to provide reliable data transfer. See slides 28-41.

Q8: In connection-oriented multiplexing, two packets with the same source host, source port, destination host and destination port might be delivered to two different sockets:

a) ☐ Yes

b) ☑ No

In connection-oriented (de-)multiplexing, packets are assigned to sockets based on the source host, source port, destination host and destination port only. See slide 12.

Q9: What is the UDP checksum of D=1010101010101010101010101010101010?

a) ☑ 1010101010101010

b) ☐ 0101010101010101

c) ☐ 1111111111111111

d) ☐ 0000000000000000

See slide 19. The checksum is the complement to 1 of the sum of each 16-bit word in D:

word 1	*1010101010101010*
word 2	*+ 1010101010101010*
wraparound	*1 0101010101010100*
sum	*0101010101010101*
checksum	*1010101010101010*

Q10: In a pipelined protocol, the sender allows N simultaneous non-acknowledged packets. This is meant to:

a) ☐ Increase network utilization, by a factor of $\frac{N}{2}$.

b) ☑ Increase network utilization, by a factor of N.

c) ☐ Reduce packet queuing time, by a factor of $\frac{N}{2}$.

d) ☐ Reduce packet queuing time, by a factor of N.

Pipelining increases network utilization by a factor of N, see slide 45. It won't reduce packet queuing time – it might even increase it when there is congestion.