EL5373

INTERNET ARCHITECTURE AND PROTOCOLS

Runze Dong

N10264442

rd1711@nyu.edu

Workstation: APAH Othello_I

MAC: f8:0f:41:c4:7f:aa

IP: 128.238.66.104

Lab Report 6

Due Nov 18, 2014

Exercise 1

Explain TCP connection establishment and termination using the tcpdump output.

TCP connection establishment by three-way handshake mechanism.

1. My host (*.104) sends TCP request segment, SYN=1, sequence number=0
2. Remote host (*.105) receives this request and sends TCP acknowledegment segment, SYN=1, ACK=1, sequence number=0, acknowledgement number=1
3. My host receives remote host's acknowledgement and sends TCP acknowledgement segment, ACK=1, sequence number=1, acknowledge number=1

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	128.238.66.104	128.238.66.105	TCP	74	57738→23 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
2	0.001143	128.238.66.105	128.238.66.104	TCP	74	23→57738 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len
3	0.001206	128.238.66.104	128.238.66.105	TCP	66	57738→23 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TS
4	0.001477	128.238.66.104	128.238.66.105	TELNET	93	Telnet Data ...

TCP termination control

TCP termination should ensure give each end of the connection a chance to shut down its own one-way data flow.

1. Remote host (*.105) sends a TCP segment with FIN flag, FIN=1, ACK=1, sequence number=100, acknowledgement number=113
It means remote host has acknowledged a TCP segment with seq 113 from my host and now it is supposed to close its transmission process.
2. My host (*.104) receives this segment and sends a TCP segment with FIN flag, FIN=1, ACK=1, sequence number=113, acknowledgement number=101.
It means my host has acknowledged remote host's close and now it is also supposed to close its transmission process.

3.Remote hos (*.105) receives this segment and sends acknowledgement segment, ACK=1, sequence number=101, acknowledgement number=114.

25	21.706820	128.238.66.104	128.238.66.105	TCP	66	57738→23 [ACK] Seq=113 Ack=100 Win
26	21.707518	128.238.66.105	128.238.66.104	TCP	66	23→57738 [FIN, ACK] Seq=100 Ack=11
27	21.707612	128.238.66.104	128.238.66.105	TCP	66	57738→23 [FIN, ACK] Seq=113 Ack=10
28	21.708620	128.238.66.105	128.238.66.104	TCP	66	23→57738 [ACK] Seq=101 Ack=114 Win

What were the announced MSS values for the two hosts?

MSS=1460 bytes

What happens if there is an intermediate network that has an MTU less than the MSS of each host? Is the DF flag set in our tcpdump output?

DF flag is set.

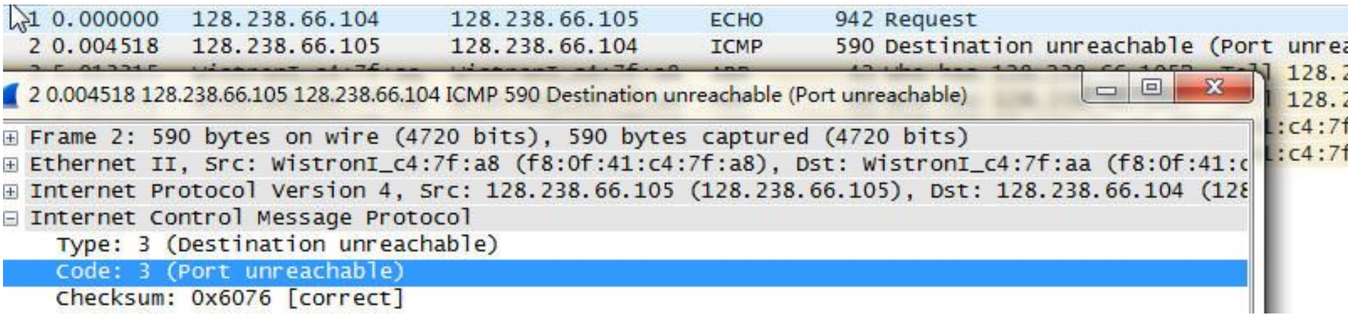
In generally, if DF flag is set, MSS should be less than MTU to avoid IP fragmentation. Otherwise, the segment will be discarded and send ICMP Destination Unreachable-Fragmentation Needed and DF Set. Then the MSS will be re-calculated.

Exercise 2

Explain what happened in both the UDP and TCP cases. When a client requests a nonexisting server, how do UDP and TCP handle this request, respectively?

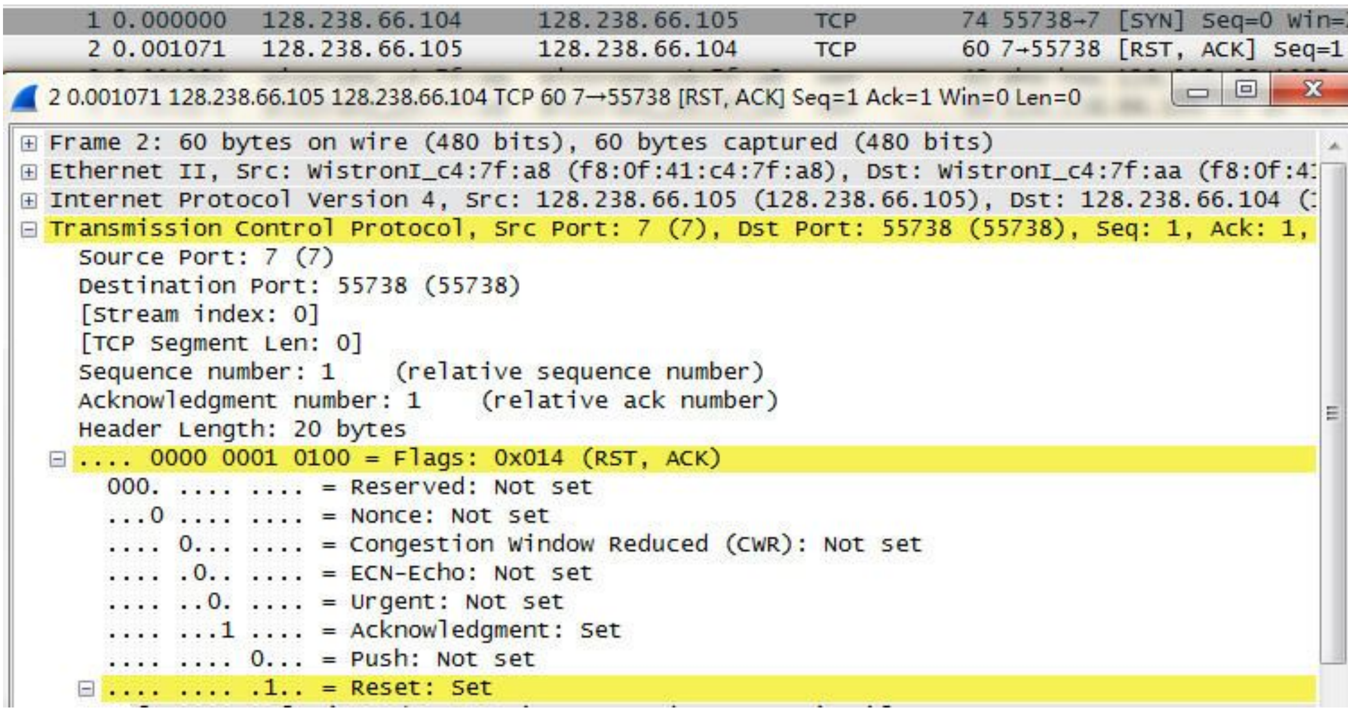
For UDP

Host receives an ICMP error message, type 3 destination unreachable, code 3 port unreachable.



For TCP

Host receives TCP reset segment and fails to establish connection with remote server.



What is your comment on the network resource utilization if we were transmitting a big file?

Even though MSS may bound the size of data segment, TCP has other methods to improve network utilization. Such as delayed acknowledgement, sliding window flow control.

Exercise 3

(1) What is a delayed acknowledgement? What is it used for?
TCP delayed acknowledgement can improve bandwidth efficiency by reducing ACK segment(without data payload) response frequency.
TCP uses delayed ACK timer to manage the delayed ACK. After data is received, the receiver delays sending the ACK until the delayed acknowledgement timer run out. During this period, if a new data will be sent to transmitter from receiver, the ACK can be piggybacked with data segment.

(2) Can you see any delayed acknowledgements in your tcpdump output? If yes, explain the reason. If you don't see any delayed acknowledgements, explain the reason why none was observed.

Yes. Delayed acknowledgement exists. Since we press and hold a button, this host will collect new data continuously within one ACK timer. ACK will be piggybacked with data segment.

It's a data segment sent from my host (*.104) to remote host (*.105). The data field contains 'h' as a message. Also the acknowledgement flag is set. The ACK is piggybacked with new data segment.

131	17.110366	128.238.66.104	128.238.66.105	TELNET	67	Telnet Data ...
132	17.111802	128.238.66.105	128.238.66.104	TELNET	67	Telnet Data ...
133	17.140853	128.238.66.104	128.238.66.105	TELNET	67	Telnet Data ...
134	17.142237	128.238.66.105	128.238.66.104	TELNET	67	Telnet Data ...
135	17.171131	128.238.66.104	128.238.66.105	TELNET	67	Telnet Data ...
136	17.172517	128.238.66.105	128.238.66.104	TELNET	67	Telnet Data ...
137	17.199550	128.238.66.104	128.238.66.105	TELNET	67	Telnet Data ...
138	17.200900	128.238.66.105	128.238.66.104	TELNET	67	Telnet Data ...
139	17.239977	128.238.66.104	128.238.66.105	TCP	66	57888→23 [ACK] Seq=165 Ack=452 Win=30336
140	23.643399	128.238.66.104	128.238.66.105	TELNET	67	Telnet Data ...
141	23.644620	128.238.66.105	128.238.66.104	TELNET	67	Telnet Data ... [Malformed Packet]
142	23.644672	128.238.66.104	128.238.66.105	TCP	66	57888→23 [ACK] Seq=166 Ack=453 Win=30336

```
⊟ Transmission Control Protocol, Src Port: 57888 (57888), Dst Port: 23 (23),
    Source Port: 57888 (57888)
    Destination Port: 23 (23)
    [Stream index: 0]
    [TCP Segment Len: 1]
    Sequence number: 160    (relative sequence number)
    [Next sequence number: 161    (relative sequence number)]
    Acknowledgment number: 447    (relative ack number)
    Header Length: 32 bytes
  ⊟ .... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 237
    [Calculated window size: 30336]
    [Window size scaling factor: 128]
  ⊞ Checksum: 0x86d5 [validation disabled]
    Urgent pointer: 0
  ⊞ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ⊞ [SEQ/ACK analysis]
⊟ Telnet
    Data: h
```

(3) What is the Nagle algorithm used for? From your tcpdump output, can you tell whether the Nagle algorithm is enabled or not? Give the reason for your answer. From your tcpdump output for when you typed very rapidly, can you see any segment that contains more than one character going from your workstation to the remote machine?

Nagle's algorithm works by combining a number of small outgoing segments, and sending them all at once. Specifically for interactive data flows, TCP sends one byte and buffers all subsequent bytes until an acknowledgement for the first byte is received. Then all buffered bytes are sent in a single segment. It will improve bandwidth efficiency but lead to more delay problem.

Nagle algorithm isn’t enabled because none of segments contains more than one character sending from my host to remote machine. This means TCP doesn’t pack all buffered bytes into one single segment.

Exercise 4

(1) Using one of three tcpdump outputs, explain the operation of TCP in terms of data segments and their acknowledgements.
After the server(*.105) sending data segments, client will give acknowledgement for this packet. The acknowledgement number will be set as sequence number received from server plus the number of byte of data part.
In this shoot-screen, sequence number in data segment is 52982. So the next acknowledgement number should be 54430 (52982+14448) in ACK segment from client to server.

```
106 3.525853   128.238.66.105   128.238.66.104   SSHv2   1514 Server: Encrypted packet (len=1448)
107 3.525892   128.238.66.104   128.238.66.105   TCP       66 37136→22 [ACK] Seq=2686 Ack=54430 Win=14
108 3.528849   128.238.66.105   128.238.66.104   SSHv2   1514 Server: Encrypted packet (len=1448)
109 3.528888   128.238.66.104   128.238.66.105   TCP       66 37136→22 [ACK] Seq=2686 Ack=55878 Win=14
```

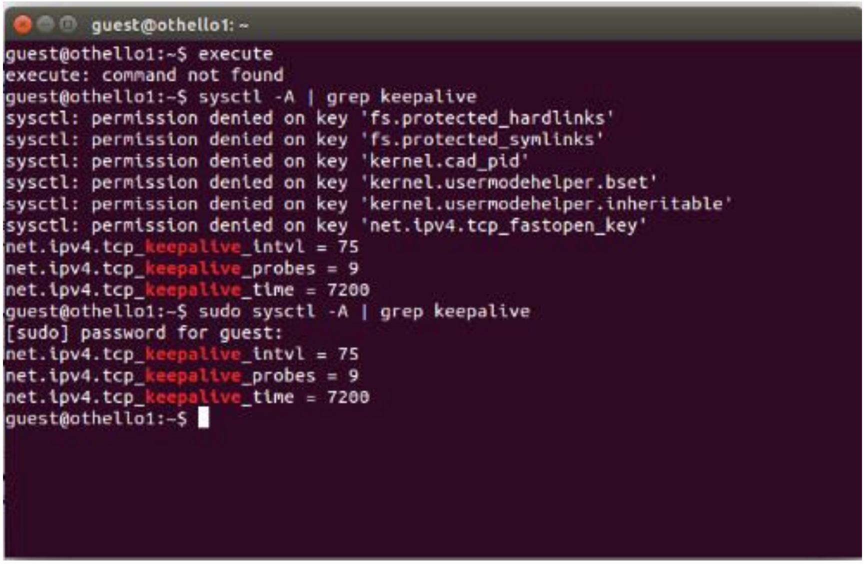
```
Transmission Control Protocol, Src Port: 22 (22), Dst Port: 37136 (37136), Seq: 52982,
  Source Port: 22 (22)
  Destination Port: 37136 (37136)
  [Stream index: 0]
  [TCP Segment Len: 1448]
  Sequence number: 52982    (relative sequence number)
  [Next sequence number: 54430    (relative sequence number)]
  Acknowledgment number: 2686    (relative ack number)
  Header Length: 32 bytes
```

Does the number of data segments differ from that of their acknowledgements?
No. They are same.

Compare all the tcpdump outputs you saved. Is the number of acknowledgments and/or segments different between the three different executions?
No. The three tcpdump outputs are same.

Exercise 5

What is the default value of the TCP keepalive timer?
7200s
What is the maximum number of TCP keepalive probes a host can send?
9



Exercise 6

From the tcpdump output, there are some exactly same ACK packets at 6.32s from server to client
the cable was unplugged at **6.32s** approximately, and reconnected at **19s.**

5042	6.315141	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3770630 Ack=1 Win=29312 L
5043	6.315148	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [PSH, ACK] Seq=3772078 Ack=1 Win=29
5044	6.319679	128.238.66.105	128.238.66.104	TCP	78	[TCP Dup ACK 5041#1] 8888→59773 [ACK] Seq=1 Ac
5045	6.319730	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3773526 Ack=1 Win=29312 L
5046	6.320746	128.238.66.105	128.238.66.104	TCP	78	[TCP Dup ACK 5041#2] 8888→59773 [ACK] Seq=1 Ac
5047	6.320792	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3774974 Ack=1 Win=29312 L
5048	6.324316	128.238.66.105	128.238.66.104	TCP	78	[TCP Dup ACK 5041#3] 8888→59773 [ACK] Seq=1 Ac
5049	6.324358	128.238.66.104	128.238.66.105	TCP	1514	[TCP Fast Retransmission] 59773→8888 [ACK] Seq
5050	6.325627	128.238.66.105	128.238.66.104	TCP	78	[TCP Dup ACK 5041#4] 8888→59773 [ACK] Seq=1 Ac
5051	6.325677	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3776422 Ack=1 Win=29312 L
5052	6.329765	128.238.66.105	128.238.66.104	TCP	66	8888→59773 [ACK] Seq=1 Ack=3776422 Win=182144
5053	6.329812	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3777870 Ack=1 Win=29312 L
5054	6.331333	128.238.66.105	128.238.66.104	TCP	66	8888→59773 [ACK] Seq=1 Ack=3777870 Win=185856
5055	6.331384	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3779318 Ack=1 Win=29312 L
5056	6.331391	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3780766 Ack=1 Win=29312 L
5057	6.339683	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3782214 Ack=1 Win=29312 L
5058	19.215699	128.238.66.104	128.238.66.105	TCP	1514	[TCP Retransmission] 59773→8888 [ACK] Seq=3777
5059	19.220173	128.238.66.105	128.238.66.104	TCP	66	8888→59773 [ACK] Seq=1 Ack=3779318 Win=185856
5060	19.220231	128.238.66.104	128.238.66.105	TCP	1514	59773→8888 [ACK] Seq=3783662 Ack=1 Win=29312 L

Describe how the retransmission timer changes after sending each retransmitted packet, during the period when the cable was disconnected.
The retransmission timer is **doubled**. TCP control use exponential backoff algorithm to update RTO.

Explain how the number of data segments that the sender transmits at once (before getting an ACK) changes after the connection is reestablished.
There are **one data segment** transmitted at once after the connection is reestablished.

Exercise 7

Did you observe any IP fragmentation?
I didn't see IP fragmentation.

We have find MSS= 1460byte. MSS(1460)+TCP header(20)+IP header(20)=1500=MTU
So the TCP segment size is bounded by MSS and it generally should be less than MTU to avoid IP fragmentation.
In chapter 5, UDP doesn't have this mechanism to restrict its size. If it is over MTU, it will be fragmented.