EL5373

INTERNET ARCHITECTURE AND PROTOCOLS

Zheng Pan

0495069

zp322@students.poly.edu

workstation: APAH

MAC: 00:16:76:a9:82:01

**Lab Report 6**

Due 17 April 2013

[5 Pages]

**Exercise 1**

The TCP connection establishment:

1. An end host initiates a TCP connection by sending a packet with. ISN, n, is the sequence number field and with an empty payload field. This packet also carries the MSS(1460) and TCP receiving window size(5840). The SYN flag bit is set in this packet to indicate a connection request.

**19:20:10.161411 apah.32946 > 128.238.66.102.time: S** **2691390479: 2691390479(0) win 5840 < mss 1460, sackOK, timestamp 90917 0, nop, wscale 0 > (DF) [tos 0x10]**

1. After receiving the request, the other end host replies with a SYN packet acknowledging the byte whose sequence number is the ISN plus 1 (ACK = n + 1=1), and indicates its own ISN m, MSS(1460), and TCP receiving window size(5792).

**19:20:10.161519 128.238.66.102.time > apah.32946: S 2883112826: 2883112826 (0) ack 2691390480 win 5792 < mss 1460, sackOK, timestamp 63100 90917, nop, wscale 0 > (DF)**

1. The initiating host then acknowledges the byte whose sequence number is the ISN increased by 1 (ACK = m + 1).

**19:20:10.161548 apah.32946 > 128.238.66.102.time: . ack 2883112827 win 5840 < nop, nop, timestamp 90917 63100 > (DF) [tos 0x10]**

The TCP connection termination:

1. TCP sends a packet with the FIN flag set

**19:20:10.162101 128.238.66.102.time > apah.32946: F 2883112831: 2883112831 (0) ack 2691390480 win 5792 < nop, nop, timestamp 63100 90917 > (DF)**

1. the other end sends the FIN segment and the ack together.

**19:20:10.163032 apah.32946 > 128.238.66.102.time: F 2691390480: 2691390480 (0) ack 2883112832 win 5840 < nop, nop, timestamp 90917 63100> (DF)** **[tos 0x10]**

1. send ack and end the connection

**19:20:10.163104 128.238.66.102.time > apah.32946: . ack 2691390481 win 5792 < nop, nop, timestamp 63101 90917 > (DF)**

MSS=1460 for both of the hosts.

If there is an intermediate network that has an MTU less than the MSS of each host, 1. If DF is set. The IP datagram will be dropped and an ICMP unreachable error will be sent to the source carrying the MTU of the next link.

2. If DF is not set. The IP datagram will be fragmented in the intermediate network.

The DF flag was set in the tcpdump output.

**Exercise 2**

When using UDP to request a nonexisting server, because of the properties of UDP, the host also send the packet to the remote host, without the reply, the host will receive an ICMP message “port unreachable” .

But for TCP transfer, first of all, we must establish the connection between them. Recording the server is nonexisting, the connection cannot be done, so the server

will send a RST segment packet to the client to close the TCP connection.

**Exercise3**

1. Delayed segment is used as a timer that goes off every K ms. After receiving a data segment, TCP delays sending the ACK until the next tick of the delayed acknowledgement timer, hoping that new data to be sent in the reverse direction will arrive from the application during this period .Otherwise, an ACK segment is sent. Depending on when the data segment is received, when there is new data arriving from the application layer, and when the delayed acknowledgement timer goes off, an ACK may be delayed from 0 ms up to K ms.
2. There is some delayed ACK, because TCP uses delayed ACK to reduce the number of small segments. Using the following example to explain the delayed ACK timer operation. From the datas, the ACK(52) which should be sent after 10, was piggybacked with the data 11 segment.

**8** 0.038825 128.238.66.102 128.238.66.103 TELNET 105 Telnet Data ...

**9** 0.038841 128.238.66.103 128.238.66.102 TCP 66 33191 > telnet [ACK] Seq=28 Ack=52 Win=5840 Len=0 TSval=210678 TSecr=209313

**10** 0.087107 128.238.66.102 128.238.66.103 TELNET 179 Telnet Data ...

(Transmission Control Protocol, Src Port: 33191 (33191), Dst Port: telnet (23), Seq: 28, Ack: 52, Len: 113)

**11** 0.088374 128.238.66.102 128.238.66.103 TELNET 69 Telnet Data ...

(Transmission Control Protocol, Src Port: telnet (23), Dst Port: 33191 (33191), Seq: 52, Ack: 141, Len: 3)

**12** 0.088402 128.238.66.103 128.238.66.102 TCP 66 33191 > telnet [ACK] Seq=141 Ack=55 Win=5840 Len=0 TSval=210683 TSecr=209318

1. It can be used to further limit the number of small segments in the Internet. Nagle Algorithm is not enabled because I cannot see any segment contains more than one character going from my workstation to the remote machine. rom my tcpdump output I can’t see any segment contains more than one character going from my workstation to the remote machine.

**Exercise4**

There is no difference between the number of data segments and their acknowledgements. The sequence of data segment and their acknowledgements among three tcpdump outputs are differents.

There four kinds of TCP Flags in the tcpdump output :

SYN:

set indicates synchronize sequence numbers. The SYN flag is initially sent when establishing the three way handshake between two hosts.

ACK:

set indicates that the acknowledgement field is significant. The acknowledgement flag is used to acknowledge the successful receipt of packets.

FIN:

set no more data from sender. This flag is used to tear down the virtual connections created using the previous flag(SYN), because of this reason the FIN flag always

appears when the last packets are exchanged between the connections.

PSH: When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process.

There are 5 different options as can be seen from the tcpdump output:

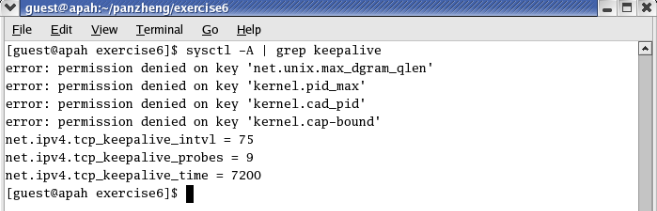
Maximum segment size: Maximum segment size.

SACK: Selective acknowledgement, followed by a list of 1-4 blocks being selectively acknowledged, specified as 32-bit begin/end pointers.

Timestamps: Timestamp and echo of previous timestamp.

NOP: No operation (padding), used to align option fields on 32-bit boundaries for better performance.

Window scale: window scaling factor to expand window size.

**Exercise5**  

The default value of TCP keepalive timer is 7200. The maximum number of TCP keepalive probes a host can send is a 9.

**Exercise6**

guest@apah exercise6]$ sudo tcpdump -nx host 128.238.66.103 and 128.238.66.102

tcpdump: listening on eth0

20:10:23.761301 128.238.66.103.33565 > 128.238.66.102.8888: . 169881:171329(1448) ack 1 win 5840 <nop,nop,timestamp 392277 364474> (DF)

20:10:23.761416 128.238.66.103.33565 > 128.238.66.102.8888: . 171329:172777(1448) ack 1 win 5840 <nop,nop,timestamp 392277 364474> (DF)

20:10:25.851013 128.238.66.103.33565 > 128.238.66.102.8888: P 159641:159745(104) ack 1 win 5840 <nop,nop,timestamp 392294 364474> (DF)

20:10:26.351022 128.238.66.103.33565 > 128.238.66.102.8888: P 159641:159745(104) ack 1 win 5840 <nop,nop,timestamp 392338 364474> (DF)

20:10:27.451025 128.238.66.103.33565 > 128.238.66.102.8888: P 159641:159745(104) ack 1 win 5840 <nop,nop,timestamp 392426 364474> (DF)

20:10:29.311018 128.238.66.103.33565 > 128.238.66.102.8888: P 159641:159745(104) ack 1 win 5840 <nop,nop,timestamp 392602 364474> (DF)

20:10:33.831014 128.238.66.103.33565 > 128.238.66.102.8888: P 159641:159745(104) ack 1 win 5840 <nop,nop,timestamp 392954 364474> (DF)

20:10:33.831149 128.238.66.102.8888 > 128.238.66.103.33565: . ack 159745 win 59368 <nop,nop,timestamp

20:10:30.531170 128.238.66.103.33565 > 128.238.66.102.8888: . 159745:161193(1448) ack 1 win 5840 <nop,nop,timestamp 392954 365156> (DF)

20:10:33.531175 128.238.66.103.33565 > 128.238.66.102.8888: . 161193:162641(1448) ack 1 win 5840 <nop,nop,timestamp 392954 365156> (DF)

There are 5 packets with exactly the same sequence number and ack number. So we could say the cable was disconnected from time 20:10:25.851013 to 20:10:33.831014. When the retransmitted packet is also lost, TCP uses the exponential backoff algorithm to update RTO when the retransmission timer expires for a retransmitted segment. We can see RTO is doubled for each retransmission, but with a maximum value of 64 seconds.

**Exercise7**

No I didn’t see any IP fragmentation

Because the MSS = 1460. The IP datagram = 1460 + 20 (TCP header) + 20 (IP header) = 1500 bytes = MTU of the Ethernet

In Exercise 5 of Chapter 5, we use UDP, UDP doesn’t have MSS, if it is larger than MTU, IP fragmentation occurs.