Computer Networks Homework 2

Fall 2019

Due: 24 February 2020

1. (6 points) List at least three responsibilities of the Transport Layer.

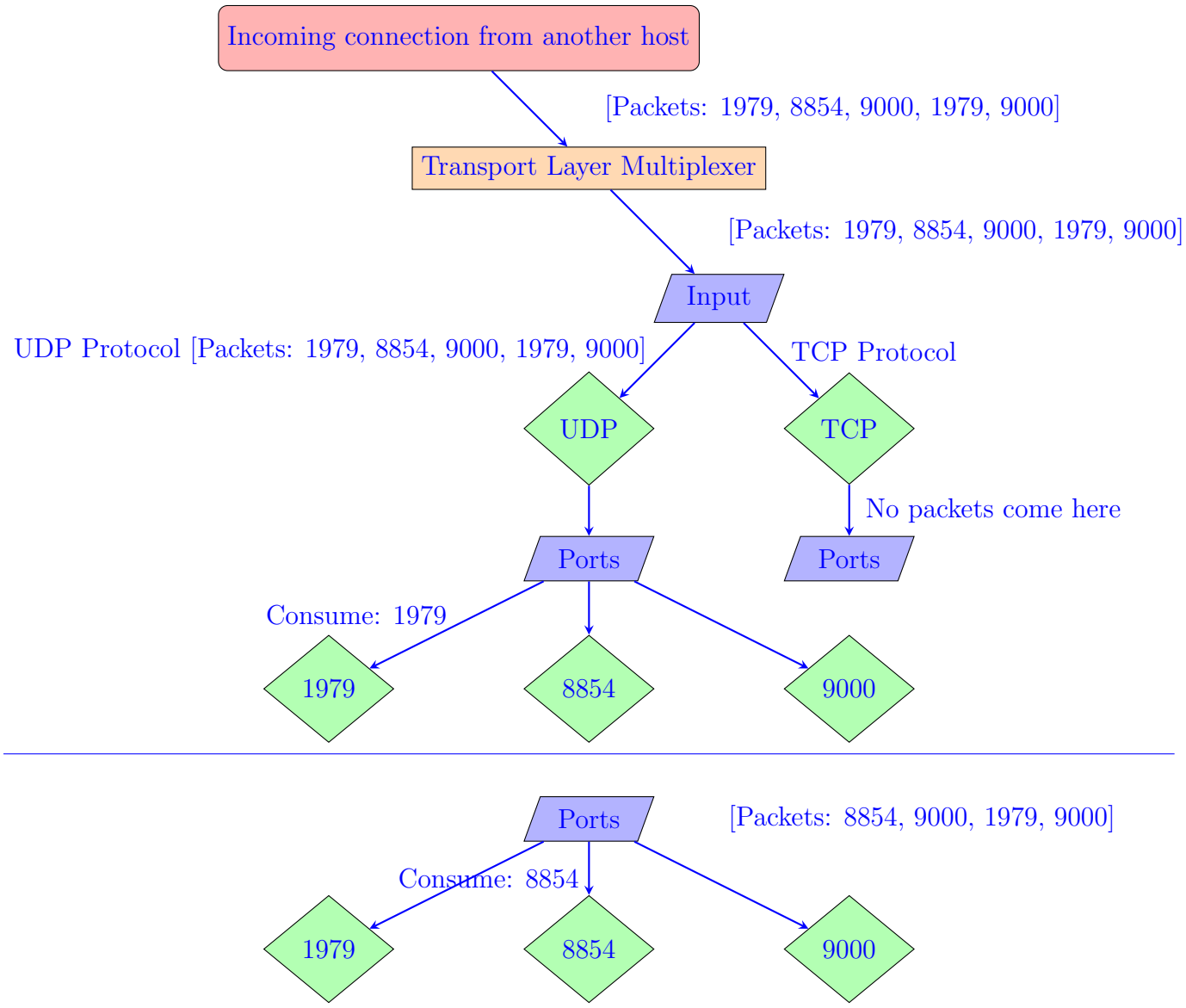
   Solution:

   The Transport Layer is responsible for the following:

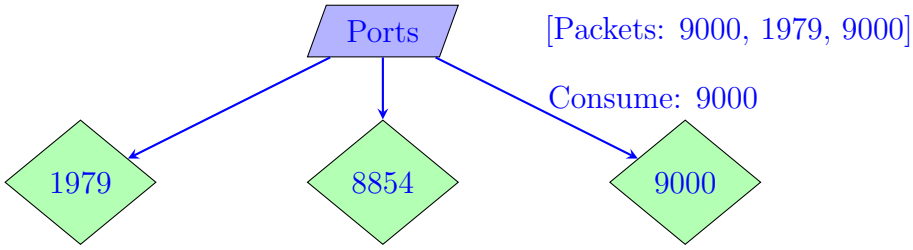
   1. Creating an end to end *connection* between two hosts.
   2. Controlling the flow of packets in a way that does not stress the two connected devices.
   3. Ensuring a complete and verified data transfer in TCP based connections.

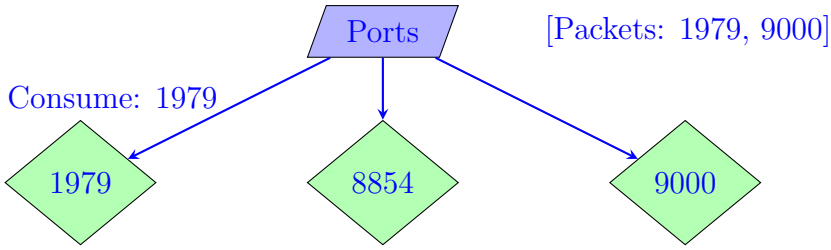
2. (5 points) A sequence of UDP packets addressed to ports 1979, 8854, 9000, 1979, and 9000 arrive at a host (in that order). Draw a simple diagram that shows how these packets enter the demux service and then how they are distributed to the processess.

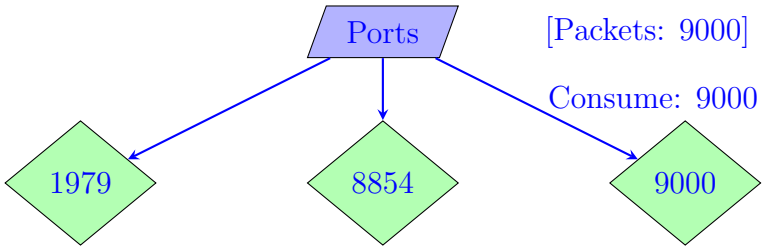
   Solution:


Incoming connection from another host
[Packets: 1979, 8854, 9000, 1979, 9000]
Transport Layer Multiplexer
[Packets: 1979, 8854, 9000, 1979, 9000]
Input
UDP Protocol [Packets: 1979, 8854, 9000, 1979, 9000]
TCP Protocol
UDP
TCP
No packets come here
Ports
Ports
Consume: 1979
1979
8854
9000

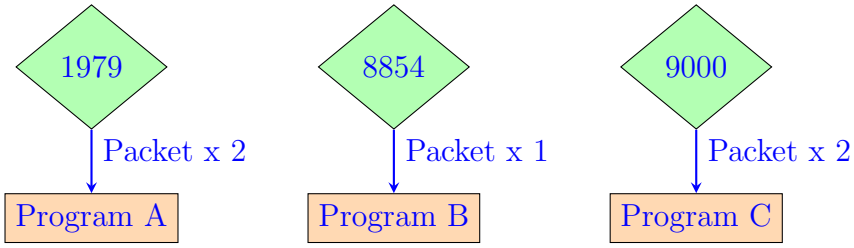
Ports
[Packets: 8854, 9000, 1979, 9000]
Consume: 8854
1979
8854
9000







All port consumptions directly map this way



3. (9 points) Use RFC1700 to find the well-known port numbers for the following services: SMTP, HTTP, Kerberos, POP3, BGP, WHOAMI, LDAP, and RPC (remote process execution). Be careful, RPC has multiple port assignments; list at least two and the associated application or algorithm.

Solution:

- SMTP:
  - 25
  - 465
  - 587
  - 3535 (semi-common but unofficial)
- HTTP:
  - 80
  - 443 (It's not exactly HTTP, but they're almost always paired)
  - 8080 (Mostly used for testing, but very common)

    Many other services use HTTP as their program layer, but will not be listed (as there's hundreds)
- Kerberos:
  - 543 (Login)
  - 544 (Remote Shell)
  - 749 (Administration)
  - 750 (Version IV)
  - 751 (Master Authentication)
  - 752 (Password Server)
  - 753 (User Reg)
  - 754 (Slave Propagator)
  - 760 (Registration)
- POP3:
  - 110
  - 995 (SSL/TLS)
- BGP:
  - 179
- WHOAMI:
  - 565

  - https://www.ietf.org/rfc/rfc1340.txt
- LDAP:
  - 389
  - 636
  - 3268 (Microsoft Active Directory LDAP)
- RPC:
  - 530 (generic)
  - 593 (HTTP RPC) (EX: Microsoft Mail Exchange)
  - 8069 (OpenERP - XML RPC) (EX: Odoo)

4. (16 points) A UDP segment has the payload "MTU!" as a string of four ASCII charaacters. It is being delivered to an application on port 5272 from an application on port 8888. Show the full UDP header for this segment.

   Clearly mark each field with it's purpose and represent the value of each field in binary. Use an 8 b representation for the ASCII characters (a convenient ASCII table is available at `www.asciitable.com`). Show all 1s Complement arthmetic for the checksum calculation, remembering that the checksum is based on 16-bit chunks. As a "real" checksum would require information from the IP packet you should just use the information from UDP and ignore the pseudoheader when calculating your checksum.

   Solution:

   UDP Datagram Header:

   - Source Port: 0001010010011000
   - Destination Port: 0010001010111000
   - Length: 0000000000001100
   - Checksum: 1101100111010001

5. (4 points) Find at least two other important UDP applications besides DHCP, DNS, and QUIC. Provide a citation which proves the application uses UDP.

   Solution:

   - TFTP: https://tools.ietf.org/html/rfc1350
   - NFSv2: https://tools.ietf.org/html/rfc1094

6. (8 points) List the two components of any protocol that adheres to the precepts of Automatic Repeat Requests. How do these two components work together to provide reliable delivery of packets?

Solution:

Primary components:

1. Packet Acknowledgment
2. Timeouts

   These work together to create automatic repeating requests by sending a packet to a host, then waiting for a response back from the host acknowledging that it received the packet. The timeout is there to ensure that the system does not wait forever for an acknowledgment, as it may never come due to a myriad of issues such as packet drops.

   When the acknowledgment occurs the system knows it is safe to move onto the next packet/s. In the other case where a timeout occurs, the system will know to resend the packets as they have not gotten to their destination in a reasonable amount of time.

7. (2 points) What problem is solved by adding a single-bit sequence number to packets using the Stop-and-Wait protocol?

Solution:

This solves the issue of knowing whether or not a packet was double received or if two packets contain the same set of data in a window. This works by numbering each packet with the single bit sequence and the ACK response also contains that number. Allowing a much more accurate idea of what all data has been successfully sent and received.

8. (5 points) You are designing a protocol that uses the sliding window protocol. You have targeted a window size of 16 frames. What is the minimum number of bits needed in the header field for packets of this protocol?

Solution:

Window size $=> 2^N = Size$

$2^N = 16$

$N = \sqrt{16}$

N = 4

9. (3 points) TCP uses a 32 b sequence number. What is the maximum window size that could be used?

Solution:

By raw 32 b $2^(32) = 4294967296$.

However, the RFC explicitly says the maximum is $2^16 = 65536$.

https://www.ietf.org/rfc/rfc1323.txt

10. (12 points) Show the states the TCP state machine moves through in the following situations:
    (a) The client initiates a connection, then changes its mind with a `close()` function.
    (b) A standard closing of the connection from the `Established` state by the client.
    (c) The client initiates a new connection, but the server never responds.

Solution:

A Start − > Allocate/Bind Socket − > Initiate connection − > Close connection − > clear Socket − > End Process

B Start − > Allocate/Bind Socket − > Initiate connection − > Send Initiation Packet SYN − > Receive ACK − > connection established − > close command − > Send FIN Packet − > Wait for ACK − > Close connection − > clear Socket − > End Process

C Start − > Allocate/Bind Socket − > Initiate connection − > Send Initiation Packet SYN − > Repeat previous until timeout − > clear Socket − > End Process

11. (4 points) They may sound similar, but explain the differences between the `PUSH` and the `URGENT` flags in a TCP segment.

Solution:

PUSH forces the buffered bytes to be sent along the line immediately, where they will be received and queued on the receiving host. Where URGENT is different, is it will be received into a special URGENT queue in which it will be available immediately and without touching the main buffer.

12. (6 points) Several segments are sent by a client containing 852 b, 777 b, and 212 b. Prior to these transmissions, the last ACK received by the client was for sequence number `0x001109AB`. What are the sequence numbers and ACKs for these segments?

Solution:

- 852 b => 0x110cff
- 777 b => 0x111008
- 212 b => 0x1110dc

13. (10 points) A TCP host is using the original timeout equations and 10 packets arrive with the following RTT values: 145 ms, 145 ms, 155 ms, 144 ms, 100 ms, 255 ms, 200 ms, 185 ms, 166 ms, 145 ms. Show how the value of the value of EstRTT changes as each packet arrives. Start with a 100 ms value for EstRTT.

Solution:

Original Timeout Equation: $RTO = Rb$

Where: $R = RTT$ and $b =$ delay variance, recommended to be static 2

1. 145ms: Fail, with EstRTT 100 $->$ EstRTT = 200
2. 145ms: Success, with EstRTT 200 $->$ EstRTT = 100
3. 155ms: Fail, with EstRTT 100 $->$ EstRTT = 200
4. 144ms: Success, with EstRTT 200 $->$ EstRTT = 100
5. 155ms: Fail, with EstRTT 100 $->$ EstRTT = 200
6. 100ms: Success, with EstRTT 200 $->$ EstRTT = 100
7. 255ms: Fail, with EstRTT 100 $->$ EstRTT = 200
8. 200ms: Success, with EstRTT 200 $->$ EstRTT = 100
9. 185ms: Fail, with EstRTT 100 $->$ EstRTT = 200
10. 166ms: Success, with EstRTT 200 $->$ EstRTT = 100
11. 145ms: Fail, with EstRTT 100 $->$ EstRTT = 200

14. (10 points) Repeat the previous question using the Jacobson/Karels equations. Let the important values be: EstimatedRTT = 100 ms, Deviation = 0, $\mu = 1$, $\gamma = 4$, $\delta = 0.875$.

Solution:

$$\text{RTT} = \alpha \cdot \text{old\_RTT} + (1 - \alpha) \cdot \text{new\_round\_trip\_sample}$$

1. 145ms: $->$ estimateRTT = 139ms
2. 145ms: $->$ estimateRTT = 144ms
3. 155ms: $->$ estimateRTT = 153ms
4. 144ms: $->$ estimateRTT = 145ms
5. 155ms: $->$ estimateRTT = 153ms
6. 100ms: $->$ estimateRTT = 106ms
7. 255ms: $->$ estimateRTT = 235ms
8. 200ms: $->$ estimateRTT = 206ms
9. 185ms: $->$ estimateRTT = 186ms
10. 166ms: $->$ estimateRTT = 168ms
11. 145ms: $->$ estimateRTT = 147ms