27/September/2024

Homework-1

CSCE-5580 Computer Network

Uday Bhaskar Valapadasu
11696364

Ans:-

They are 5 layers that build that internet protocol stack.

1.) Application : supporting network applications

Functionality: Application exchanges messages to implement some application service using services of transport layer.

Key protocols:

HTTP : for web browsing

FTP : for file transfers

SMTP: for sending emails

POP3/IMAP: For retrieving emails.

DNS (P0) : resolves domain names to IP address

* It provides interfaces ~~between~~ with user ~~software~~ applications ~~which~~ to provide network services. Allows communication between software applications and the network.

2.) Transport - Layer :

Functionality:

→ This protocol transfers application layer message 'M' from one process to anoth using services of network layer.

→ It encapsulates the application layer message (M) with the transport layer - layer Header (H_t) to create a segment.

* H_t used by transport layer protocol to implement its services.

Key protocoles:

1.) TCP (Transmission Control Protocol):

→ connection oriented

→ Guarantees delivery, order and error recovery.

2.) UDP (User Datagram Protocol):

→ connectionless

→ Faster but don't guarantee delivery or order.

3.) Network layer

Functionalities:

1.) This protocol transfers the transport layer segement from one host to another host using the link layer services.

2.) It encapsulates the the transport layer segement $[H_t|M]$ with the network layer-layer header (H_n) to create a network-layer datagram.

3.) H_n is used by the network layer to provide services.

Key Protocols:-

* IP (Internet protocol) :-
  - → IPV4 : 32-bit address
  - → IPv6 : Uses 128-bit addresses

* ICMP (Internet control Message Protocol): for error messages and diagnosis (eg. ping)

* ARP (Address Resolution Protocol); Resolves IP addresses to MAC address

4.) Link-Layer:

→ This protocol transfers datagram [Hn|Ht+M] from host to neighboring host, using network-layer services.

→ Link-layer protocol encapsulates network datagram [Hn|[Ht|M], with link-layer header H_l to create a link-layer frame. [H_l|[Hn|[Ht|M]]

Common Protocols:

1.) Ethernet: A protocol that defines wired data transmission on local area networks (LANs)

2.) Wi-Fi (IEEE 802.11): Protocol for wireless data transmission

3.) PPP (Point-to-Point Protocol): used for direct communication between two nodes.

5.) Physical Layer: (Lowest layer)

Functionality:

1.) Responsible for the physical connection between devices, including the transmission and reception of raw bitstreams over a physical medium.

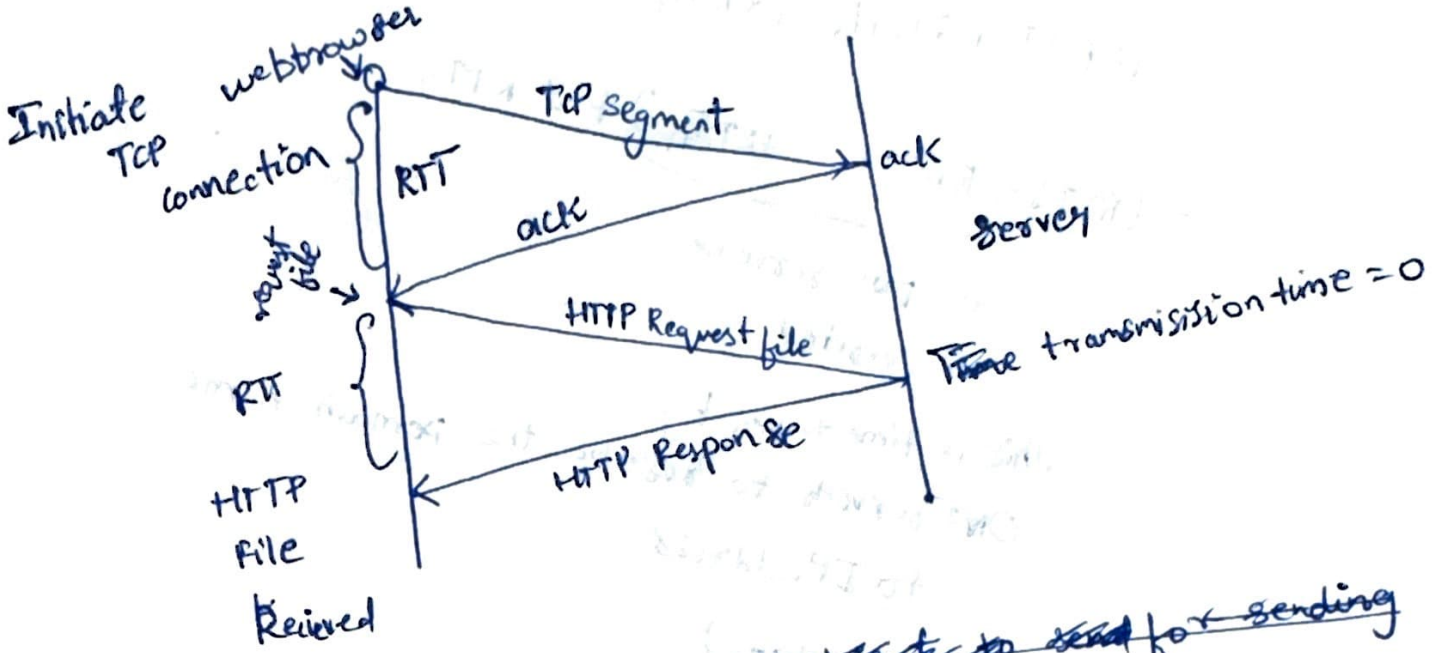
Common Elements:

→ cables: Copper wires, fiber optics
→ Wireless Transmission: Radio frequencies, infrared etc
→ Network Devices: Hubs, repeaters, network interface cards (NICs)

2 Ans:-

2.a) Non-persistent HTTP connections:

- Each object requires a separate TCP connection
- A new connection is established (incurring RTT) and then closed after each object retrieval.
- In, non-persistent HTTP, a new TCP connection is established for each object, including the initial web page



→ ~~The general time taken an objects to send for sending and obj an object and request~~ is:
response

→ The total time taken for an Object to be obtained is

2RTT + Transmission time = 2RTT + 0 = 2RTT

1.) Initial TCP connection for Web page and retrieving the main web page is

$$2RTT_0$$

2.) Retrieving the 8 objects :

$$2RTT_0 * 8 = 16\,RTT_0$$

∴ The total time elapsed with non-persistent HTTP =

$$= 16RTT_0 + 2RTT_0 = 18\,RTT_0$$

$$= (RTT1 + RTT2 + \ldots + RTTn) + 18\,RTT_0$$

n - DNS servers visited

This is time taken by DNS servers to resolve the domain name to IP address (DNS Lookup)

2 Ans:

2. b) Persistent HTTP (Without Pipelining)

1.) DNS Lookup takes $(RTT_1 + RTT_2 \cdots + RTT_n)$ time.

2.) Persistent HTTP creates one connection that is reused for all the objects retrieval.

→ On general time taken for the intiating TCP connection and retreival of Main Web page is:

$$RTT_0 + RTT_0 = 2RTT_0$$

→ Retreiving 8 small objects (Persistent takes only 1RTT for requesting and retreival of object as the connection is alive.

$$1RTT_0 * 8 = 8RTT_0$$

considering without ~~pipelining~~ pipelining, without connections parallel.

∴ The total time elapsed for persistent HTTP connection is:

= DNS Lookup ~~total~~ time + $2RTT_0$ (TCP connection of webpage) + $8RTT_0$ (for 8 small objects)

$= (RTT_1 + RTT_2 + RTT_3 + \cdots RTT_n) + 10RTT_0$ → without pipelining

∴ The persistent connection are more efficient as they reduce the time spent establishing new TCP connections.

* **Persistent connection with pipelining** :- This is the default mode of HTTP.

Pipelining: Allows multiple HTTP requests to be sent out back-to-back without waiting for the corresponding responses. This feature minimizes the delay between requests and response, improving performance by reducing the total elapsed time.

→ In this type, all 8 object requests are sent out immediately after the initial request for the webpage.

→ Since these requests are pipelined, the objects are fetched almost in parallel, requiring only one RTT_0 for all of them combined because they are sent in a single burst.

$$\text{Total time} = (RTT_1 + RTT2 + \ldots + RTT_n) + 3 \times RTT_0$$

↓

with pipelining

3Ans:

The IP addresses given in the figure

Web client Host A → A
Web client Host C → C
Web server Host B → B
} IP addresses

Flow from server back to the clients:

To host A:	To host C: (Left Process)	To host C: (right process)
Source port: 80 destination port: 26145 Source IP: B destination IP: A	Source port: 80 destination port: 7532 source IP: B destination IP: C	Source port: 80 destination port: [redacted] 26145 Source IP: B destination IP: C

4Ans:-

Lets take an example to understand the scenario of checksum recieved UDP segment

→ Consider two 16-bit words of a packet.

Word 1 : 11001100 11001100

Word 2 : 00110011 00110011

→ Binary addition of two 16-bit words

```
  1100 1100 1100 1100
  0011 0011 0011 0011
  ------------------- → sum (without any overflow)
  1111 1111 1111 1111
```

→ Now, we need to perform 1's complement to get checksum of two words

1111 1111 1111 1111

One's complement = 0000 0000 0000 0000 → this value is checked at recieved UDP segement to verify the integrity of data.

→ Now, lets introduce some transmission errors

Suppose the following bits flip:

- In word 1, the 1st bit (MSB) flips from 1 to 0

Previous word 1: 1100 1100 1100 1100

modified word 1: 0100 1100 1100 1100

- In word 2, the 1st bit (MSB) flips from 0 to 1

Previous word 2: 0011 0011 0011 0011

modified word 2: 1011 0011 0011 0011

→ Now, Adding the corrupted words

```
  0100 1100 1100 1100
+ 1011 0011 0011 0011
-------------------
  1111 1111 1111 1111
```

1's complement = 0000 0000 0000 0000

Thus, even though there were transmission errors (two bit were flipped), the checksum will still verify correctly as 0000 0000 0000 0000 which matches the original checksum.

∴ The receiver cannot be completely sure that no ~~error~~ bit errors have occured due to way the packet's checksum is calculated. if corresponding bits in 16-bit words flips the checksum shows the same result with 1's complement, causing the checksum to verify correctly even if a transmission error occured.

Ans

In this scenario, a deadlock occurs between the sender and the reciever in the rdt2.1 protocol due to the corrupted acknowledgment (ACK).

1.) Sender's initial state:

The sender is in the state "wait for call 1 from above" and sends a data packet with the sequence number 1. The sender then transitions to "wait for ACK or NAK 1", waiting to hear back from the reciever.

2.) Receiver's initial state:

The receiver is in the state "wait for 1 from below", expecting a packet with sequence number 1. It correctly receives packet with sequence number 1, sends an ACK, and transitions to "wait for 0 from below", ready to receive the next packet with sequence number 0.

3. Corrupted ACK:

Unfortunately, the ACK sent by the receiver gets corrupted during transmission, so when the sender recieves it, it cannot tell if it's an ACK or a NAK. Because of this, the sender assumes there was an error and resends the packet with sequence number 1.

4. Receiver Waiting for Packet 0:

The receiver, however, is now in the state 'wait for 0 from below' and is expecting a packet with sequence number 0. But the sender keeps sending the packet with sequence number 1. Since reciver was expecting sequence number 0, it sends a NAK Because it didn't recieve the correct sequence number.

5.> Endless Loop:

The sender keeps resending packet 1 after receiving NAKs, while the receiver expects packet 0 and keeps rejecting packet 1. This creates a deadlock, trapping both in loop.

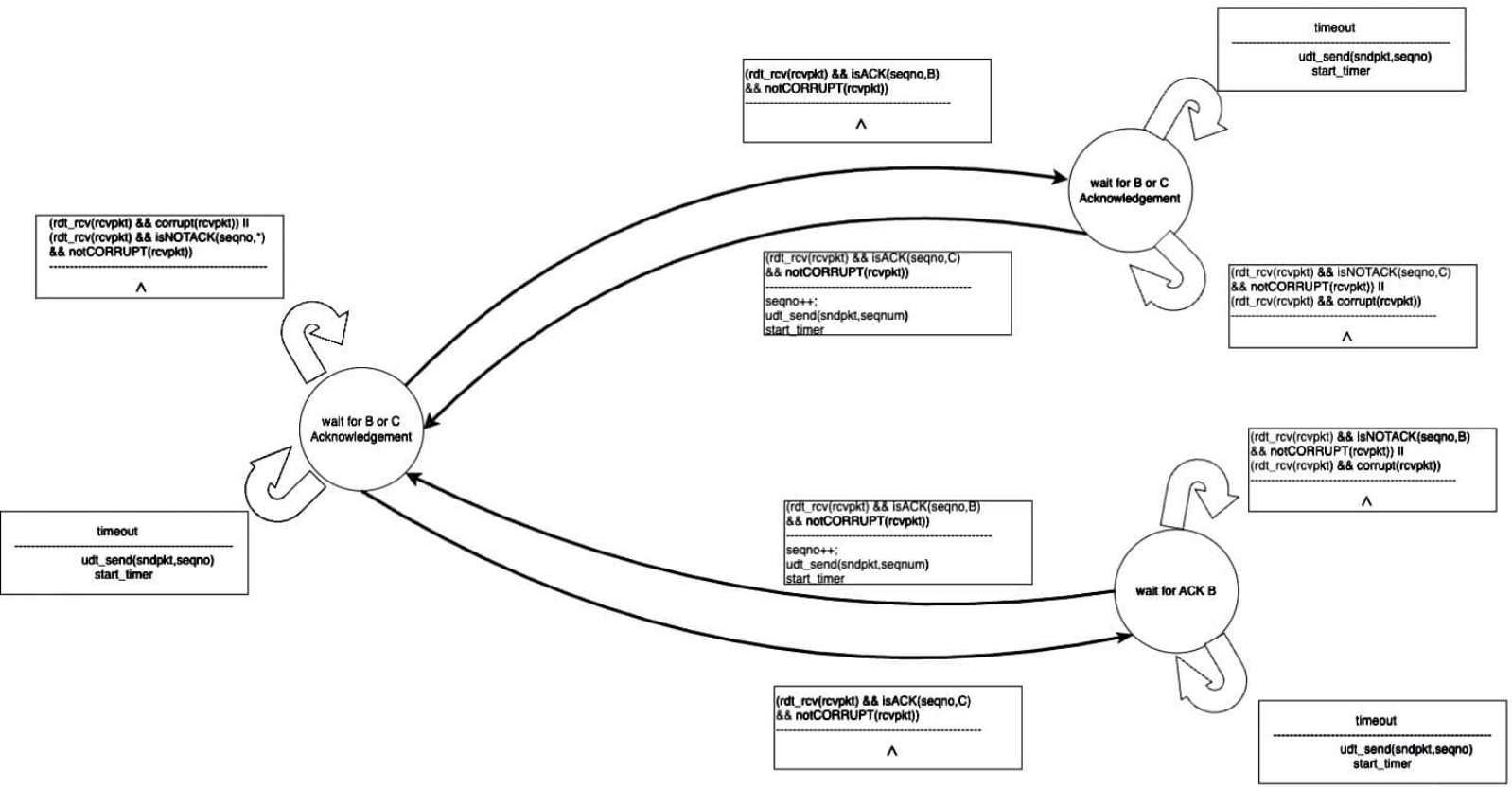
6 Ans:

This problem is a variation of the stop-and-wait protocol (rdt 3.0). Since messages can be lost or retransmitted (due to early timeouts or because one receiver hasn't correctly recieved the data), sequence numbers are necessary. Similar to rdt 3.0, a 1-bit sequence number is sufficient. The Sender's state indicates whether it has received an ACK from B/C, or neither, while the receiver's state tracks which sequence number it is expecting next.

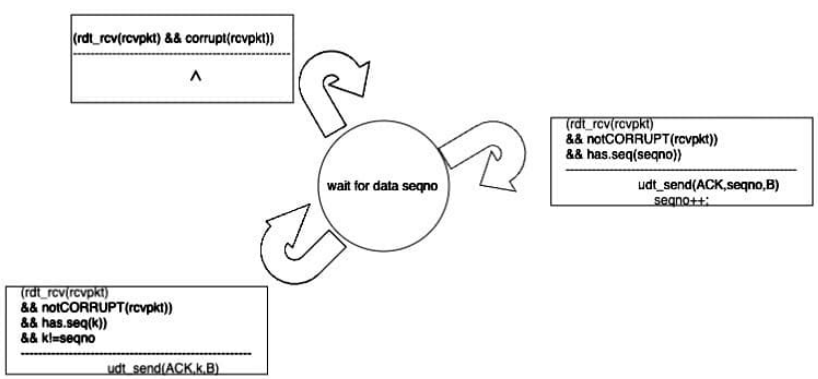
Uday Bhaskar V

11696364

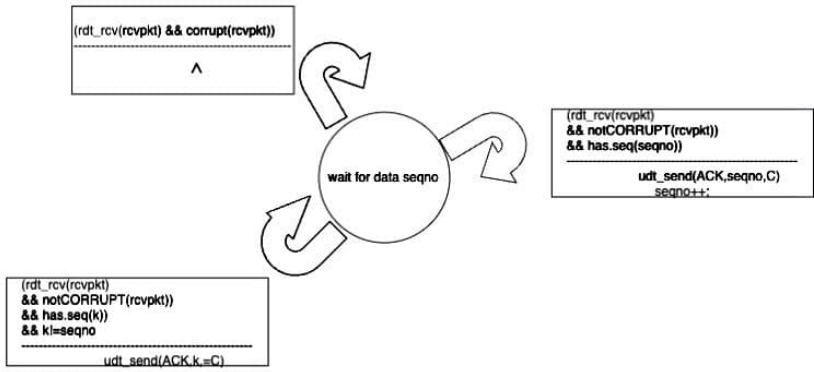
Sender B



Receiver B



Receiver C



7 Ans :-

7.a) Second Segment (Host A to B)

Sequence number = 207

Source port = 302

destination port number = 80

7.b) → The first segment arrives before the second one

→ The acknowledgment for the first segment = 207

→ The source port = 80

→ The destination = 302

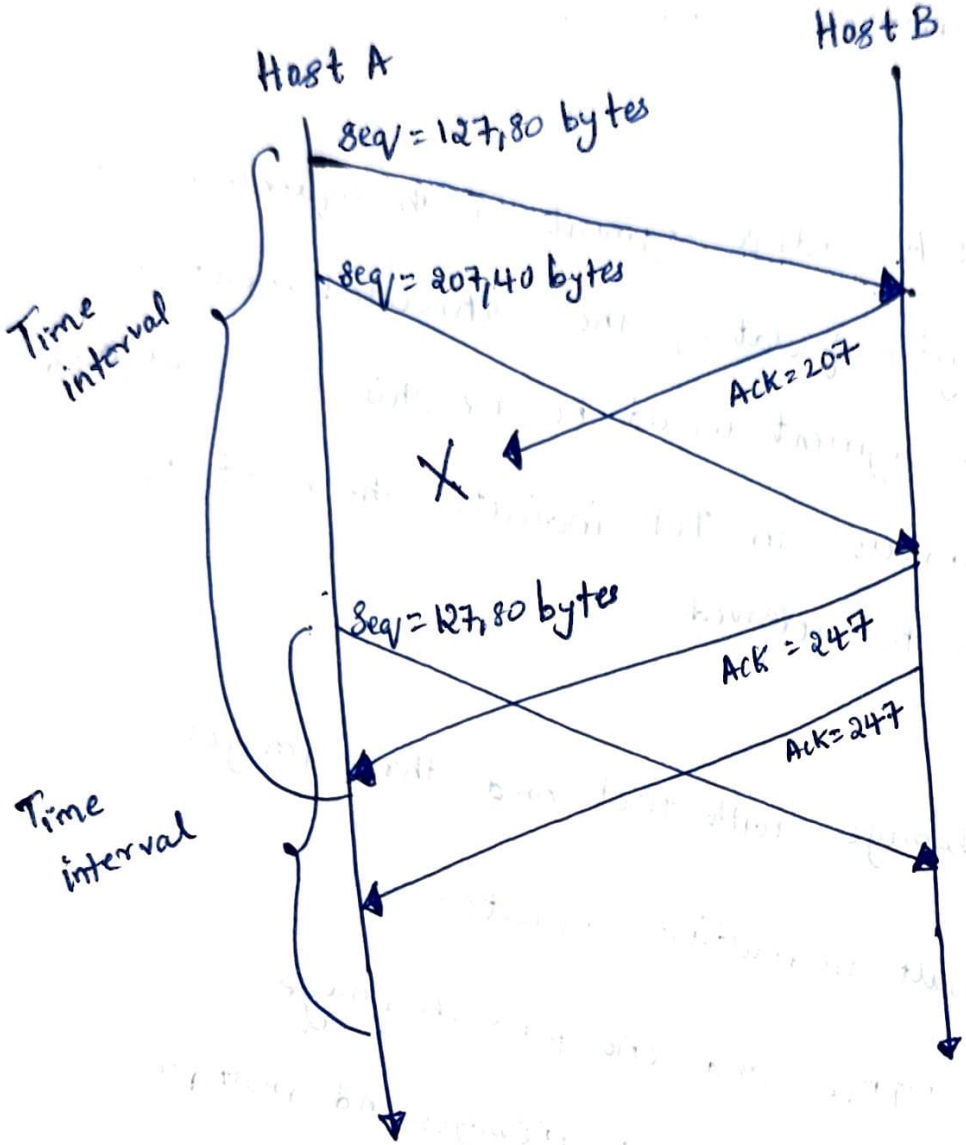
7.c) If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment,

acknowledgement no = 127,

indicating that it is still waiting for bytes 127 and onwards.

7Ans

7.a)



8Ans:

8.1) False. Host B will send acknowledgements to Host A, even without data to piggy back on. TCP requires acknowledgements for reliable data transfers, regardless of whether the receiving host has data to send back.

8.2.) False. Tcp receive window (rwnd) can change during a connection. It's used for flow control and can be adjusted based on the receiver's buffer capacity and network conditions.

8.3) True. if Host A sends a segment with sequence number 38 and 4 bytes of data, the acknowledgment number in the same segment would be 42. This is because of the ACK number in TCP indicates the next byte expected to be received.

8.4.) False. A webpage with text and three images typically results in multiple requests:
One for the HTML and one for each image
This leads to multiple requests messages and multiple response messages, not just one request and four requests.

8.5.) False, HTTP response messages can have empty message bodies. This occurs in response to HEAD requests or with certain status code like 204 No content.