BUDDHA SERIES

(Unit Wise Solved Question & Answers)

Course – B.Tech. (CSE Allied-AIML/DS)

College – Buddha Institute of Technology

(AKTU CODE-525)

Department: Computer Science & Allied-
PROGRAM: AIML-DS

Subject: Computer Networks
(BCS 603)

Faculty Name: Mr. Shailesh Kumar Patel

Unit - 4

Q1. Differentiate TCP and UDP in context of the header format. **(AKTU 2022-23)**

Solution:

User Datagram: UDP packets, called user datagrams, have a fixed-size header of 8 bytes.

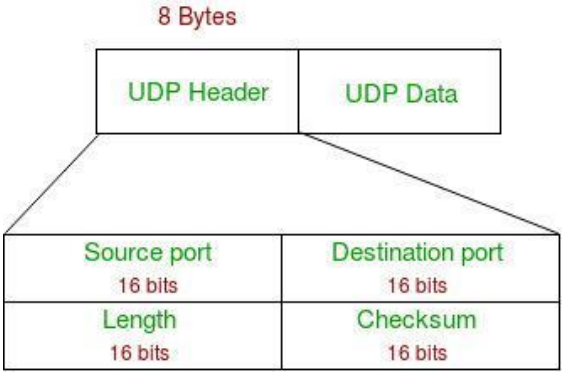


Figure 1: User datagram format

The fields are as follows:

Source port number This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535 (216-1). If the source host is the client (a client sending a request), the port number, in most cases, is an ephemeral port number requested by the process and chosen by the UDP software running on the source host. If the source host is the server (a server sending a response), the port number, in most cases, is a well-known port number.

Destination port number This is the port number used by the process running on the destination host. It is also 16 bits long. If the destination host is the server (a client sending a request), the port number, in most cases, is a well-known port number. If the destination host is the client (a server sending a response), the port number, in most cases, is an ephemeral port number. In this case, the server copies the ephemeral port number it has received in the request packet.

Length This is a 16-bit field that defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes. A user datagram is encapsulated in an IP datagram. There is a field in the IP datagram that defines the total length.

UDP length = IP length - IP header's length

Checksum This field is used to detect errors over the entire user datagram (header plus data).

TCP, like UDP, is a process-to-process (program-to-program) protocol. TCP, therefore, like UDP, uses port numbers. Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

TCP is called a connection-oriented, reliable transport protocol. It adds connection-oriented and reliability features to the services of IP.

Segment:

A packet in TCP is called a segment. Format The format of a segment is shown in Figure 2.

The header of a TCP segment can range from 20-60 bytes. 40 bytes are for options. If there are no options, a header is 20 bytes else it can be of upmost 60 bytes.

Header fields:

Source Port Address –

A 16-bit field that holds the port address of the application that is sending the data segment.

Destination Port Address –

A 16-bit field that holds the port address of the application in the host that is receiving the data segment.

Sequence Number –

A 32-bit field that holds the sequence number, i.e, the byte number of the first byte that is sent in that particular segment. It is used to reassemble the message at the receiving end of the segments that are received out of order.

Acknowledgement Number –

A 32-bit field that holds the acknowledgement number, i.e, the byte number that the receiver expects to receive next. It is an acknowledgement for the previous bytes being received successfully.

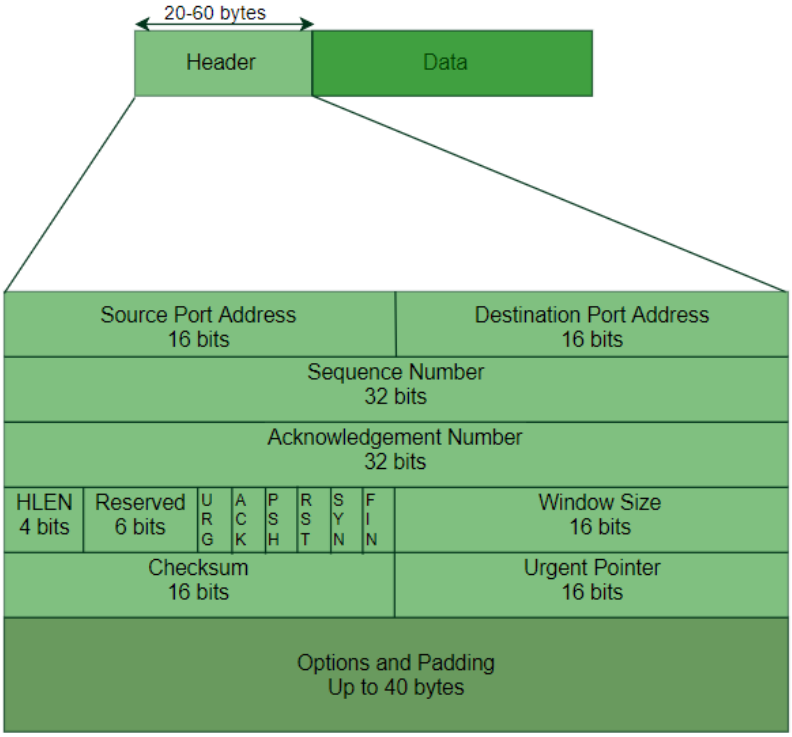


Figure 2: TCP segment format

Header Length (HLEN) –

This is a 4-bit field that indicates the length of the TCP header by a number of 4-byte words in the header, i.e if the header is 20 bytes (min length of TCP header), then this field will hold 5 (because 5 x 4 = 20) and the maximum length: 60 bytes, then it'll hold the value 15(because 15 x 4 = 60). Hence, the value of this field is always between 5 and 15.

Control flags –

These are 6 1-bit control bits that control connection establishment, connection termination, connection abortion, flow control, mode of transfer etc. Their function is:

- URG: Urgent pointer is valid
- ACK: Acknowledgement number is valid(used in case of cumulative acknowledgement)
- PSH: Request for push
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: Terminate the connection

Window size –

This field tells the window size of the sending TCP in bytes.

Checksum –

This field holds the checksum for error control. It is mandatory in TCP as opposed to UDP.

Urgent pointer –

This field (valid only if the URG control flag is set) is used to point to data that is urgently required that needs to reach the receiving process at the earliest. The value of this field is added to the sequence number to get the byte number of the last urgent byte.

Options -

There can be up to 40 bytes of optional information in the TCP header.

Q2. What do you understand by Quality of Service, parameters? List various Quality of Service parameters. **(AKTU 2018-19)**

Solution:

Quality of Service (QoS) is an important concept, particularly when working with multimedia applications. Multimedia applications, such as video conferencing, streaming services, and VoIP (Voice over IP), require certain bandwidth, latency, jitter, and packet loss parameters. QoS methods help ensure that these requirements are satisfied, allowing for seamless and reliable communication.

Quality-of-service (QoS) refers to traffic control mechanisms that seek to differentiate performance based on application or network-operator requirements or provide predictable or guaranteed performance to applications, sessions, or traffic aggregates. The basic phenomenon for QoS is in terms of packet delay and losses of various kinds.

QoS Specification

- Delay
- Delay Variation(Jitter)
- Throughput
- Error Rate

Types of Quality of Service

- **Stateless Solutions** – Routers maintain no fine-grained state about traffic, one positive factor of it is that it is scalable and robust. But it has weak services as there is no guarantee about the kind of delay or performance in a particular application which we have to encounter.
- **Stateful Solutions –** Routers maintain a per-flow state as flow is very important in providing the Quality-of-Service i.e. providing powerful services such as guaranteed services and high resource utilization, providing protection, and is much less scalable and robust.

QoS Parameters

- **Packet loss:** This occurs when network connections get congested, and routers and switches begin losing packets.
- **Jitter:** This is the result of network congestion, time drift, and routing changes. Too much jitter can reduce the quality of voice and video communication.
- **Latency:** This is how long it takes a packet to travel from its source to its destination. The latency should be as near to zero as possible.
- **Bandwidth**: This is a network communications link's ability to transmit the majority of data from one place to another in a specific amount of time.
- **Mean opinion score:** This is a metric for rating voice quality that uses a five-point scale, with five representing the highest quality.

Q3. The following is the dump of a TCP header in hexadecimal format: **(AKTU 2018-19)**

05320017 00000001 00000000 500207FF 00000000

(i) What is the source port number?

(ii) What is the destination port number?
(iii) What is the sequence number?
(iv) What is the acknowledgment number?
(v) What is the length of the header?
(vi) What is the type of the segment?
(vii) What is the window size?

Solution: The following is the dump of a TCP header in hexadecimal format:
05320017 00000001 00000000 500207FF 00000000

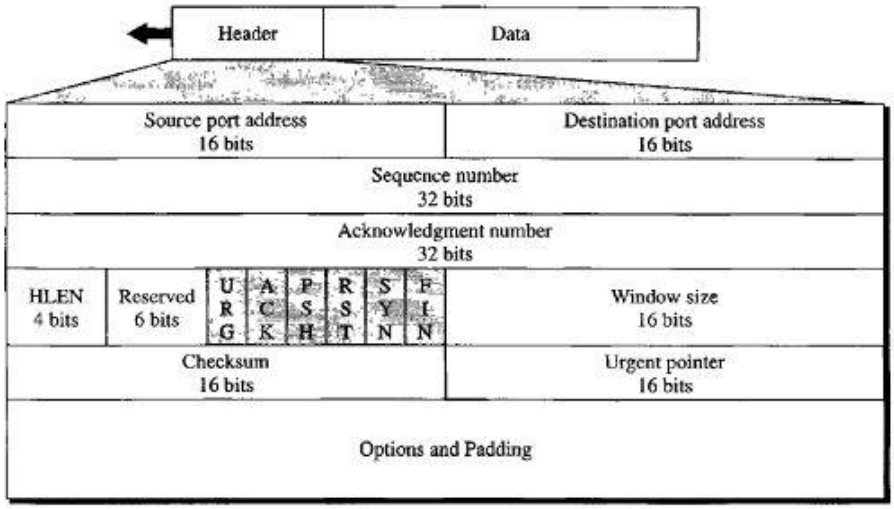


Figure:TCP segment format

(i) The source port number: source port is 2 bytes take =0532
$= (0000010100110010)_2$

0532
$= (1330)_{10}$
(ii) The destination on port number is 2 bytes as destination address = 0017
$= (0000000000010111)_2$
$= (23)_{10}$ (default TCP port)
(iii) The sequence number is 4 bytes as sequence number = 00000001
$= (00000000000000000000000000000001)_2$
$= (1)_{10}$
(iv) The acknowledgment number is next 4 bytes as ask = 00000000
$= (00000000000000000000000000000000)_2$
$= (0)_{10}$
(v) The length of the header is 4 bits as HLEN = 5
$= (0101)_2$
$= (5)_{10}$
This indicates number of sets of 4 bytes which makes the header length = 5*4
20 bytes
(vi) The type of the segment is 6 bits are reserved i.e.0 = 0000 and 2 bits from next 0 (0000) i.e. 00. Remaining two bits i.e. 00 and 4 bits of next 2 i. e. 0010.
Total 6 bits of control field = last two bits of 0 and 4 bits of 2

= 000010

=	U R G	A C K	P S H	R S T	S Y N	F I N

i.e. type of segment is SYN.

(vii) The window size is 2 bytes indicate the window length = 07FF

$= (0000011111111111)_2$

$= (2047)_{10}$

Q4. What is congestion? Briefly describe the techniques that prevent congestion. **(AKTU 2017-18, 2024-25)**

Solution:

Congestion in a network may occur if the load on the network-the number of packets sent to the network-is greater than the capacity of the network-the number of packets a network can handle.

Congestion in a computer network happens when there is too much data being sent at the same time, causing the network to slow down. Just like traffic congestion on a busy road, network congestion leads to delays and sometimes data loss.

Congestion Control techniques in Computer Networks:

Congestion control refers to the techniques used to control or prevent congestion. Congestion control techniques can be broadly classified into two categories:

- Open Loop Congestion Control
- Closed Loop Congestion Control

1) Open Loop Congestion Control

Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.

Policies adopted by open loop congestion control –

- **Retransmission Policy:** It is the policy in which retransmission of the packets are taken care of. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.

  To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.

- **Window Policy:** The type of window at the sender's side may also affect the congestion. Several packets in the Go-back-n window are re-sent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and make it worse.

  Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

- **Discarding Policy:** A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discard the corrupted or less sensitive packages and also be able to maintain the quality of a message.

  In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.

- **Acknowledgment Policy:** Since acknowledgements are also the part of the load in the network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.

  The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send an acknowledgment only if it has to send a packet or

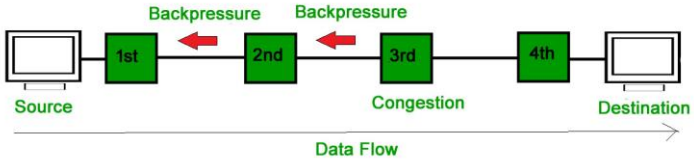
a timer expires.

- **Admission Policy:** In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of congestion or there is congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

2) Closed Loop Congestion Control

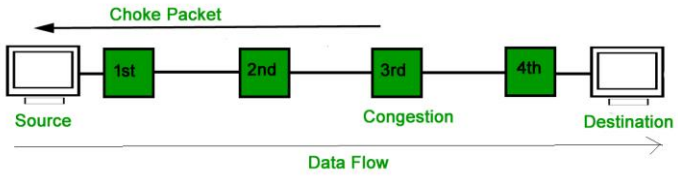
Closed loop congestion control techniques are used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:

- **Backpressure:** It is a technique in which a congested node stops receiving packets from upstream node. This may cause the upstream node or nodes to become congested and reject receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagates in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.



In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and inform the source to slow down.

- **Choke Packet Technique:** Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets have traveled are not warned about congestion.



- **Implicit Signaling:** In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is congestion.
- **Explicit Signaling:** In explicit signaling, if a node experiences congestion it can explicitly send a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating a different packet as in case of choke packet technique. Explicit signaling can occur in either forward or backward direction.
    - **Forward Signaling:** In forward signaling, a signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopts policies to prevent further congestion.
    - **Backward Signaling:** In backward signaling, a signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

Q5. Enumerate on TCP header and working of TCP and differentiate TCP and UDP with frame format.

(AKTU 2017-18)

Solution:

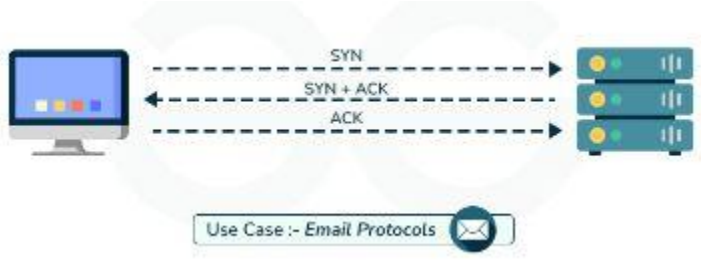
A TCP header consists of the following fields:

- **Source Port:** A 16-bit number identifying the sending application on the source host.
- **Destination Port:** A 16-bit number identifying the receiving application on the destination host.
- **Sequence Number:** A 32-bit number that indicates the order of the first byte of data in the segment, used for reliable data delivery.
- **Acknowledgement Number:** A 32-bit number that acknowledges the next byte the sender expects to receive from the receiver.
- **Data Offset (Header Length):** A 4-bit field specifying the length of the TCP header in 32-bit words.
- **Reserved Flags:** 6 reserved bits that must be set to zero
- **Control Flags:** Several flags like SYN (synchronize), ACK (acknowledge), FIN (finish), RST (reset), URG (urgent) used to control connection establishment and data flow.
- **Window Size:** A 16-bit field indicating how much data the receiver is willing to accept before needing to send another acknowledgement
- **Checksum:** A 16-bit field used for error detection in the TCP header and payload
- **Urgent Pointer (optional):** A 16-bit field used to indicate the position of urgent data within the segment

Working of Transmission Control Protocol (TCP):

Transmission Control Protocol (TCP) model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end to make sure that each message reaches its target location intact. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go.

After a particular message is broken down into bundles, these bundles may travel along multiple routes if one route is jammed but the destination remains the same.



For Example: When a user requests a web page on the internet, somewhere in the world, the server process that request and sends back an HTML Page to that user. The server makes use of a protocol called the HTTP Protocol. The HTTP then requests the TCP layer to set the required connection and send the HTML file.

Now, the TCP breaks the data into small packets and forwards it toward the Internet Protocol (IP) layer. The packets are then sent to the destination through different routes.

The TCP layer in the user's system waits for the transmission to get finished and acknowledges once all packets have been received.

Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Type of Service	TCP is a connection-oriented	UDP is the Datagram-oriented

	protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error-checking mechanism using checksums.
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.
Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). This means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
Header Length	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
Weight	TCP is heavy-weight.	UDP is lightweight.
Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It's a connectionless protocol i.e. No handshake
Broadcasting	TCP doesn't support Broadcasting.	UDP supports Broadcasting.
Protocols	TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet .	UDP is used by DNS, DHCP, TFTP, SNMP , RIP , and VoIP .
Stream Type	The TCP connection is a byte stream.	UDP connection is a message stream.
Overhead	Low but higher than UDP.	Very low.
Applications	This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services.	This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc.

Q6. Explain the three-way handshaking protocol to establish the transport level connection.

(AKTU 2017-18, 2024-25)

Solution:

TCP Connection:

In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

Connection Establishment: TCP transmits data in full-duplex mode.

Three-Way Handshaking: The connection establishment in TCP is called three-way handshaking. The process starts with the server. The server program tells its TCP that it is ready to accept a connection. This is called a request for a passive open.

The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process as shown in following figure.

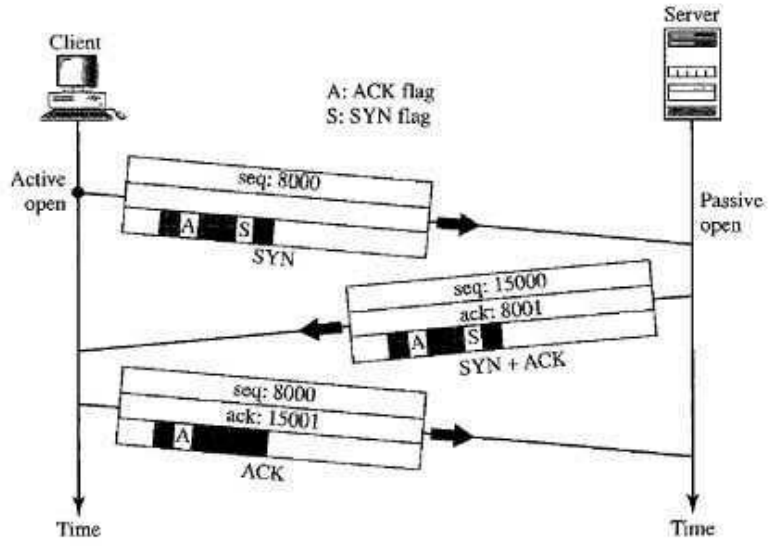


Figure: Connection establishment using three-way handshaking

The three steps in this phase are as follows:

- A SYN segment cannot carry data, but it consumes one sequence number.
- A SYN + ACK segment cannot carry data, but does consume one sequence number.
- An ACK segment, if carrying no data, consumes no sequence number.

Simultaneous Open A rare situation, called a simultaneous open, may occur when both processes issue an active open.

SYN Flooding Attack The connection establishment procedure in TCP is susceptible to a serious security problem called the SYN flooding attack. This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams.

The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers. This SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

Q7. Enumerate how the transport layer unsure that the complete message arrives at the destination and in the proper order. **(AKTU 2017-18)**

Solution:

The transport layer ensures a complete message arrives at the destination in the proper order by: dividing the message into segments, assigning sequence numbers to each segment, and using acknowledgment mechanisms to verify receipt of each segment, allowing the receiver to reassemble the message correctly and re-request any missing segments if needed; this process is primarily implemented through protocols like TCP (Transmission Control Protocol) which utilize a sliding window mechanism for efficient data flow.

The transport layer ensures that all the fragments of a transmission arrive at the destination, not some of them. On the sending end, all the fragments of transmission are given sequence numbers by a transport layer. These sequence numbers allow the receiver's transport layer to identify the missing segment.

The transport layer achieves reliable delivery by using:

- Segmentation
- Sequence Numbers
- Acknowledgment (ACK)
- Retransmission
- Sliding Window

Q8. What is congestion? Explain leaky bucket algorithm. **(AKTU 2003-4)**

Solution:

Congestion in a network may occur if the load on the network-the number of packets sent to the network-is greater than the capacity of the network-the number of packets a network can handle.

Congestion in a computer network happens when there is too much data being sent at the same time, causing the network to slow down. Just like traffic congestion on a busy road, network congestion leads to delays and sometimes data loss.

Leaky Bucket: If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.

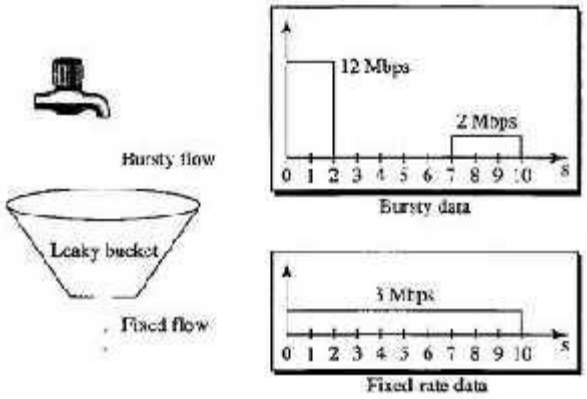


Figure: Leaky bucket

In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In figure the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbps of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbps of data. In all, the host has sent 30 Mbps of data in l0s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10s.

A simple leaky bucket implementation is shown in following figure. A FIFO queue holds the packets. If the traffic consists of fixed-size packets, the process removes a fixed number of packets from the queue at

each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits. The following is an algorithm for variable-length packets:

- Initialize a counter to n at the tick of the clock.
- If n is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until n is smaller than the packet size.
- Reset the counter and go to step 1.

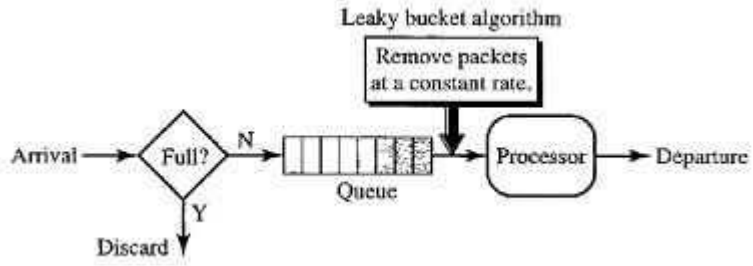


Figure: Leaky bucket implementation

Q9. Discuss TCP window management in detail. Also explain silly window syndrome and their solution.

(AKTU 2018-19)

Solution:

TCP window management, also known as the "sliding window" mechanism, is a crucial feature of the Transmission Control Protocol (TCP) that allows the sender to control the amount of data it can send to the receiver before needing an acknowledgment, effectively regulating data flow and preventing the receiver from being overwhelmed by a rapid influx of data, especially on networks with varying bandwidths or latency; it dynamically adjusts the sending window size based on the receiver's available buffer space and network conditions, ensuring efficient data transmission.

In TCP window management:

- **Sliding Window Concept:** Imagine a window that slides along the data stream, representing the range of data that the sender can currently transmit without waiting for an acknowledgment. As the receiver acknowledges received data, the window slides forward, allowing the sender to send more data.
- **Window Size:** The size of the window determines the maximum amount of data the sender can send before needing an acknowledgment. This size is communicated by the receiver through the TCP header, allowing the sender to adjust its sending rate based on the network conditions and receiver's capacity.
- **Acknowledgement (ACK):** When the receiver successfully receives a segment of data, it sends an acknowledgment packet back to the sender, indicating that it is ready to receive more data and updating the window size accordingly.
- **Flow Control:** TCP window management acts as a flow control mechanism, preventing the sender from sending data faster than the receiver can process it.

How TCP window management works:

- **Initial Window Size:** When a TCP connection is established, the receiver advertises its initial window size to the sender.
- **Data Transmission:** The sender transmits data within the specified window size.
- **Acknowledgment:** Upon receiving data, the receiver sends an acknowledgment packet with a new window size, indicating how much more data it is ready to receive.
- **Sliding Window:** As the sender receives acknowledgments, the window slides forward, allowing it to send new data segments.

- **Congestion Control:** In case of network congestion, the sender may need to decrease its window size to avoid overwhelming the network, and gradually increase it again when congestion subsides.

Silly Window Syndrome:

Silly Window Syndrome is a problem that arises due to poor implementation of TCP. It degrades the TCP performance and makes the data transmission extremely inefficient. The problem is called so because:

- It causes the sender window size to shrink to a silly value.
- The window size shrinks to such an extent that the data being transmitted is smaller than TCP Header.

The two major causes of this syndrome are as follows:

- Sender window transmitting one byte of data repeatedly.
- Receiver window accepting one byte of data repeatedly.

Cause-1: Sender window transmitting one byte of data repeatedly –

Suppose only one byte of data is generated by an application. The poor implementation of TCP leads to transmit this small segment of data. Every time the application generates a byte of data, the window transmits it. This makes the transmission process slow and inefficient. The problem is solved by Nagle's algorithm.

Nagle's algorithm suggests:

- Sender should send only the first byte on receiving one byte data from the application.
- Sender should buffer all the rest bytes until the outstanding byte gets acknowledged.
- In other words, sender should wait for 1 RTT(Round Trip Time).

After receiving the acknowledgement, sender should send the buffered data in one TCP segment. Then, sender should buffer the data again until the previously sent data gets acknowledged.

Cause-2: Receiver window accepting one byte of data repeatedly –

Suppose consider the case when the receiver is unable to process all the incoming data. In such a case, the receiver will advertise a small window size. The process continues and the window size becomes smaller and smaller. A stage arrives when it repeatedly advertises window size of 1 byte. This makes receiving process slow and inefficient. The solution to this problem is Clark's Solution.

Clark's solution suggests:

- Receiver should not send a window update for 1 byte.
- Receiver should wait until it has a decent amount of space available.
- Receiver should then advertise that window size to the sender.

Nagle's algorithm is turned off for those applications that require data to be immediately send. Nagle's algorithm can introduce delay as it sends only one data segment per round trip.

Both Nagle's as well as Clark's algorithm can work together. Both are complementary.

Q10. What is TCP? Connection termination in TCP is symmetric, whereas connection establishment is not. Why? **(AKTU 2012-13)**

Solution:

TCP (Transmission Control Protocol) is one of the main protocols of the TCP/IP suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. Transmission Control Protocol (TCP) ensures reliable and efficient data transmission over the internet.

Transmission Control Protocol (TCP) is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP.

The position of TCP is at the transport layer of the OSI model. TCP also helps in ensuring that information

is transmitted accurately by establishing a virtual connection between the sender and receiver.

Yes, connection termination in TCP is considered "symmetric" because both sides of the connection must actively participate in the closing process by sending FIN (Finish) segments and acknowledging them, ensuring that neither side can fully close the connection without the other's cooperation; essentially, both parties need to agree to end the communication before it is truly terminated.

TCP connection termination symmetry:

- **Four-way handshake:** Unlike the three-way handshake for connection establishment, terminating a TCP connection requires a four-way handshake where each side sends a FIN segment and acknowledges the other's FIN to properly close the connection.
- **Half-close possibility:** While the process is symmetric, one side can initiate the close by sending its FIN first, creating a "half-closed" state where one side can still receive data while no longer sending any.
- **Importance of symmetry:** This symmetry ensures that no data is lost during closure, as both sides need to confirm that they have received all outstanding data before fully shutting down.

The connection establishment process itself is considered asymmetric because one side (typically the client) initiates the connection by actively sending a SYN packet, while the other side (the server) passively waits for the connection request and responds with a SYN-ACK packet, making the initial connection setup not perfectly balanced between the two parties.

TCP connection establishment asymmetry:

- **Active Open vs. Passive Open:** The client performs an "active open" by sending the initial SYN packet, while the server performs a "passive open" by listening for incoming connection requests on a specific port.
- **Three-Way Handshake:** Even though the connection establishment involves a three-way handshake (SYN, SYN-ACK, ACK), the initiation of the handshake is clearly driven by the client.

Q11. What is Silly Window Syndrome? Explain how sender and receiver can avoid it using Nagle's and Clark's algorithms. **(AKTU 2024-25)**

Solution:

Silly Window Syndrome is a problem that arises due to poor implementation of TCP. It degrades the TCP performance and makes the data transmission extremely inefficient. The problem is called so because:

- It causes the sender window size to shrink to a silly value.
- The window size shrinks to such an extent that the data being transmitted is smaller than TCP Header.

This problem occurs when data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads data 1 byte at a time.

- ❖ The main purpose of Nagle's algorithm is to avoid inefficient use of bandwidth. When data come into the sender one byte at a time, just send the first byte and buffer all the rest until the outstanding byte is acknowledged. Then send all the buffered characters in the TCP segment and start buffering again until they are all acknowledged. If the user is typing quickly and the network is slow, a substantial number of characters may go in each segment, greatly reducing the bandwidth used.

  The algorithm additionally allows a new packet to be sent if enough data have trickled in to fill half the window or a maximum segment.
- ❖ Clark's solution is to prevent the receiver from sending a window update for 1 byte. Instead it is forced to wait until it has a decent amount of space available and advertise that instead. Specifically, the receiver should not send a window update until it can handle the maximum segment size it advertised when the connection was established or until its buffer is half empty, whichever is smaller.

The sender can also help by not sending tiny segments. Instead, it should wait until it has accumulated enough space in the window to send a full segment or at least one containing half of the receiver's buffer size.

Nagle's algorithm and Clark's solution to the silly window syndrome are complementary. Nagle's algorithm tries to solve the problem caused by the sending application delivering data to TCP a byte at a time. Clark's solution tries to solve the problem of the receiving application sucking the data up from TCP a byte at a time.

Both solutions can work together. The goal is for the sender not to send small segments and the receiver not to ask for them.

Q12. What is the role of stub in Remote Procedure Call (RPC) mechanism? (**AKTU 2024-25**)

Solution:

In a Remote Procedure Call (RPC) mechanism, the stub acts as a crucial intermediary (or proxy) that transparently handles all the low-level details of network communication, making a remote function call appear as if it were a local one. Stubs exist on both the client and server sides.

The main purpose of an RPC is to allow a local computer (client) to invoke procedures on a remote computer (server).

In a Remote Procedure Call (RPC) system, a stub is a piece of code that acts as a local representative or proxy for a remote procedure or service. Its primary function is to hide the complexities of network communication, making the remote call appear and behave like a standard, local function call to the developer.