4.5.9 TCP Congestion Control

- When the load offered to any network is more than it can handle, congestion builds up. When a connection is established, the sender initializes the congestion window to the size of the maximum segment in use on the connection.
- When the congestion window is 'n' segments, if all 'n' are acknowledged on time, the congestion window is increased by the byte count corresponding to 'n' segments. In effects, each burst acknowledged doubles the congestion window.
- The congestion window keeps growing exponentially until either a timeout occurs or the receiver's window is reached.
- The Internet congestion control algorithm uses the threshold parameter which is initially 64 kB, in addition to the receiver and congestion windows. When a timeout occurs, the threshold is set to half of the current congestion window, and the congestion window is reset to one maximum segment.

- Slow start is then used to determine what the network can handle, except that exponential growth stops when the threshold is hit. From that point on, successful transmissions grow the congestion window linearly instead of one per segment.
- Fig. 4.5.7 shows an example of the Internet congestion algorithm.

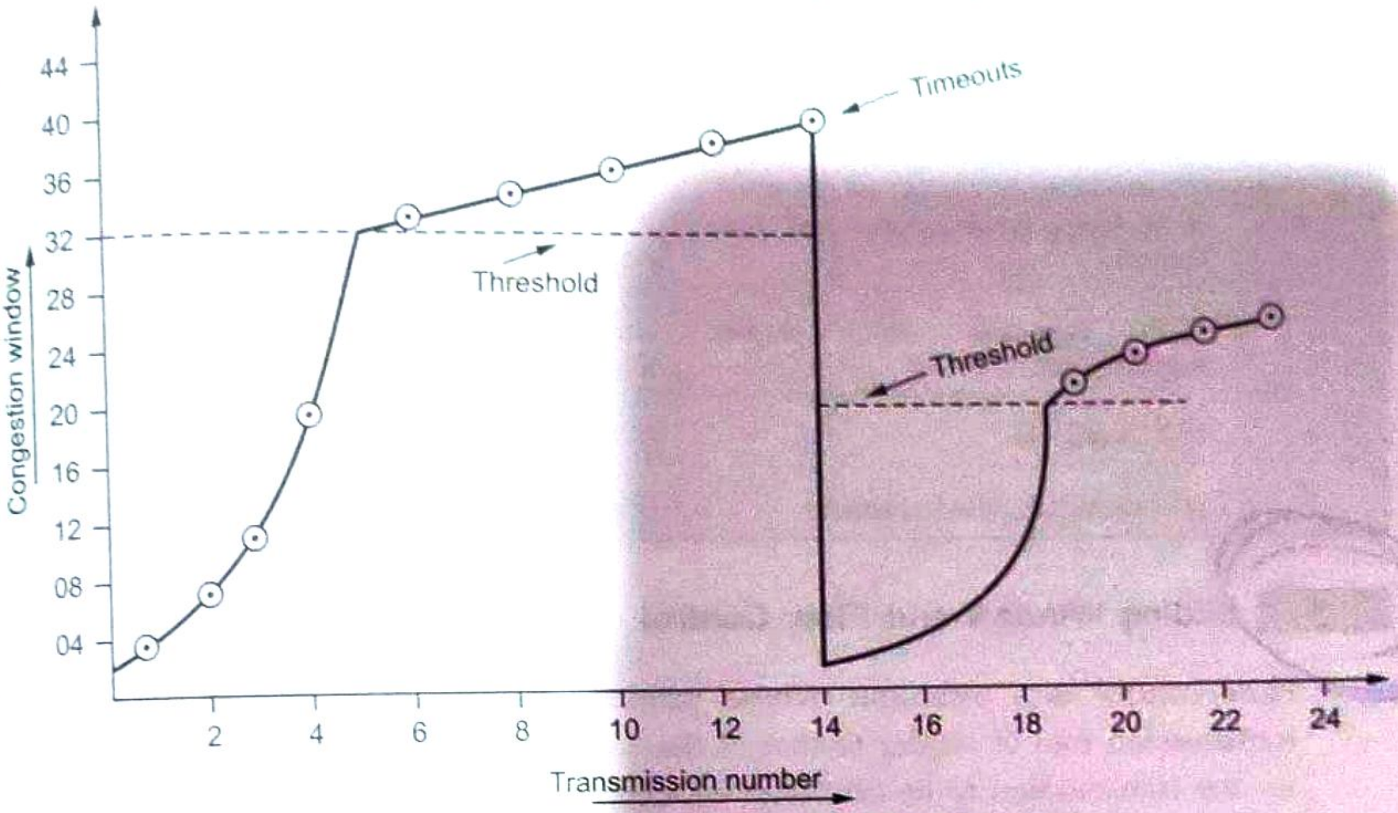

Fig. 4.5.7 Example of Internet congestion algorithm

- The maximum segment size is 1024 bytes. Initially the congestion window was 64 kB, but a timeout occurred, so the threshold is set to 32 kB and the congestion window to 1 kB for transmission 0 (zero) here.
- The congestion window then grows exponentially until it hits the threshold (32 kB). Starting then, it grows linearly.
- Transmission 13 is unlucky and a timeout occurs. The threshold is set to half the current window, and slow start is initiated all over again.
- When the acknowledgements from transmission 14 start coming in, the first four each double the congestion window, but after that, growth becomes linear again.
- If no more timeouts occur, the congestion window will continue to grow up to the size of the receiver's window. At that point, it will stop growing and remain constant as long as there are no more timeouts and the receiver's window does not change size.

4.8 Congestion Avoidance

AU : May-14, Marks 16

AU : Dec.-10,12,13, May-14

- A congestion avoidance scheme allows a network to operate in the region of low delay and high throughput. It is a prevention mechanism while congestion control is a recovery mechanism.

4.8.1 DECbit Scheme

- DECbit means destination experiencing congestion bit.
- DECbit method is developed on the Digital Network Architecture (DNA). It split the responsibility between routers and end hosts. It is router-based congestion avoidance method.
- Uses a *congestion-indication bit* in packet header to provide feedback about congestion. Upon packet arrival, the average queue length is calculated for last (busy + idle) period plus current busy period. When the average queue length exceeds one, the router sets the congestion-indicator bit in arriving packet's header.
- If at least half of packets in source's last window have the bit set, decrease the congestion window exponentially.
- Queue length is counted over last busy period + idle + current busy period.
- Source machine adjust the packet flow rate. Source machine maintains a congestion window. It observes how many packets have the congestion bit set to 1 in the last window worth of packets.
- If less than 50 % of the ACKs have the DECbit set, then increase the window by 1 packet, otherwise, set the window to 0.875 times the original value.

4.8.2 RED

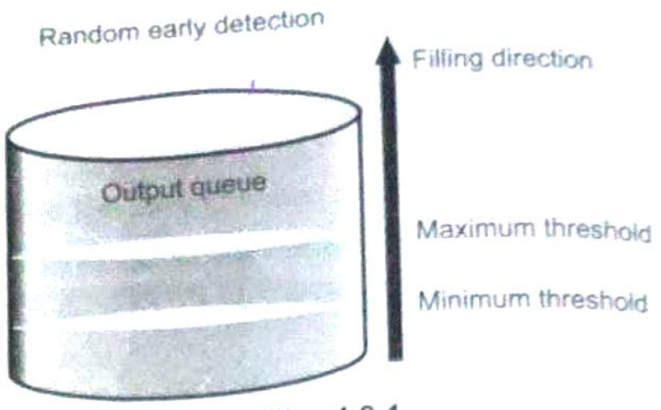

Fig. 4.8.1

- RED stands for Random Early Detection. The main idea is to provide congestion control at the router for TCP flows. RED is based on DECbit, and was designed to work well with TCP.
- RED implicitly notifies sender by dropping packets. Packet dropping probability is increased as the average queue length increases. The moving average of the queue length is used so as to detect long term congestion, yet allow short term bursts to arrive.
- Properties of RED :
  1. Drops packets before queue is full, in the hope of reducing the rates of some flows.
  2. Drops packet for each flow roughly in proportion to its rate.
  3. Drops are spaced out in time.
  4. Because it uses average queue length, RED is tolerant of bursts.
  5. Random drops hopefully desynchronize TCP sources.
- RED calculates the average queue length using a weighted running average. Following formula is used for calculating average queue length

Average length = (1 – Weight) × Average length + Weight × Sample Length

Where sample length is queue length each time a packet arrives. The weight parameter is in between 0 and 1 i.e. **0 < Weight < 1.**

- RED uses a packet drop profile to handle packet discarding. This profile defines a set of dropping probabilities according to the level of queue occupancy. A minimum and a maximum threshold are defined as shown in the Fig. 4.8.1.

1. If the queue occupancy lies beneath the minimum threshold, then packet drop does not occur.
2. If the queue occupancy is somewhere between minimum and maximum thresholds then packets are dropped according to the configured drop probability.
3. When the queue occupancy crosses maximum threshold, then all new packets attempting to enter the queue are discarded.

4.4 User Datagram Protocol

AU : May-11, Dec.-11,13

- UDP is a simple, datagram-oriented, transport layer protocol. This protocol is used in place of TCP. UDP is connectionless protocol provides no reliability or flow control mechanisms. It also has no error recovery procedures.
- Several application layer protocols such as TFTP (Trivial File Transfer Protocol) and the RPC use UDP. UDP makes use of the port concept to direct the datagrams to the proper upper-layer applications. UDP serves as a simple application interface to the IP.
- Fig. 4.4.1 (a) shows the encapsulation of a UDP datagram as an IP datagram.

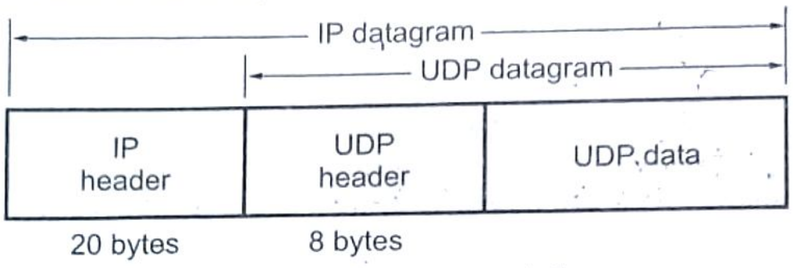

Fig. 4.4.1 (a) UDP encapsulation

- Fig. 4.4.1 (b) shows the format of the UDP header. The port number identify the sending process and the receiving process.

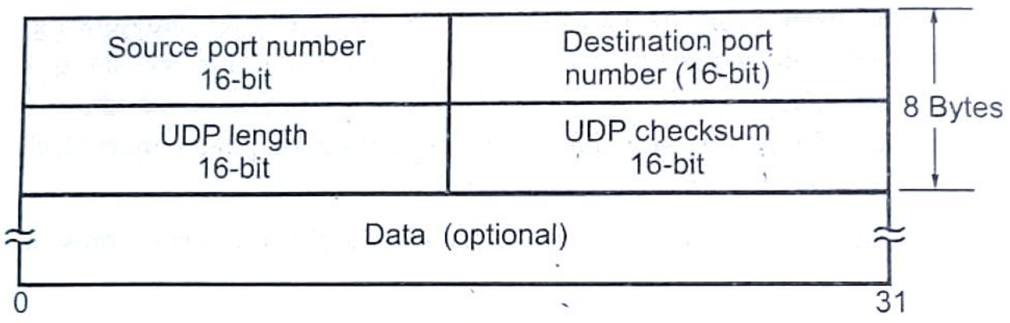

Fig. 4.4.1 (b) UDP header

- The UDP datagram contains a source port number and destination port number. Source port number identifies the port of the sending application process. The destination port number identifies the receiving process on the destination host machine.
- The UDP length field is the length of the UDP header and the UDP data in bytes. The minimum value for this field is 8 bytes.
- UDP checksum covers the UDP header and the UDP data. Both UDP and TCP include a 12 byte pseudo-header with the UDP datagram just for the checksum computation. This pseudo_header includes certain fields from the IP header. The

purpose is to let UDP double check that the data has arrived at the correct destination.

- UDP checksum is end-to-end checksum. It is calculated by the sender, and then verified by receiver. It is designed to catch any modification of the UDP header or data anywhere between sender and receiver.

4.4.1 Port Numbers

- UDP uses port numbers as the addressing mechanism in the transport layer.
- Following is the list of well-known port number used by UDP.

Port No.	Protocol	Description
7	Echo	Echoes a received datagram back to the sender.
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns the quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol

Applications of UDP

1. UDP is used for some route updating protocols such as RIP.
2. UDP is used for multicasting.
3. It is suitable for a process with internal flow and error control mechanisms.