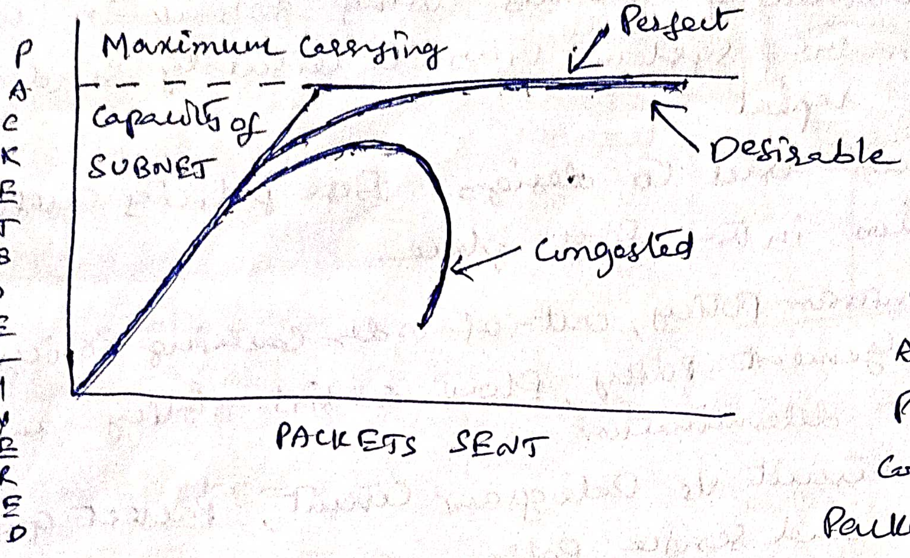
Congestion control :- A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

Effects of congestion :- i) As delay increases, performance decreases
ii) If delay increases, retransmission occurs, making situation worse



PROBLEM STATEMENT :- When too many packets are transmitted through a network congestion occurs. At very high traffic performance collapses completely and almost no packets are delivered.

CAUSES :- Bursty nature of traffic is the root cause → when part of N/w no longer can cope a sudden increase of traffic, congestion builds upon. Other facts such as lack of bandwidth, ill configuration and slow routers can also bring up congestion

SOLUTION :- Congestion control and two basic principles
a) Open loop :- Try to prevent congestion occurring by good design
b) Closed loop :- Monitor the system to detect congestion, pass this information to where action can be taken and adjust system operations to correct the problem (detect, defeat and correct)

Differences b/w Congestion and Flow Control :-
i) Congestion Control :- Try to make sure subnet can carry offered traffic, a global issue involving all the hosts and routers. It can be open-loop based or involving feed back
ii) Flow Control :- Is related to point-to-point traffic between given sender & receiver, it always involves direct feedback from receiver to sender

OPEN LOOP CONGESTION CONTROL :-

Prevention :- Different policies at various layers can affect Congestion and these are summarized in the table

Eg:- Retransmission policy at the data link layer affects congestion. A jumpy sender that times out quickly and retransmits all the outstanding frames using go back 'n' will put a heavy load on the system than a leisurely sender that uses selective repeat.

→ Congestion prevention tries to design these policies carefully to minimise congestion in the first place

Transport ⇒ Retransmission policy, out-of-order caching policy, Acknowledgement policy, flow control policy and Time out determination

Network ⇒ Virtual circuit vs Datagram circuit, Packet ~~queuing~~ queueing and service policy, Packet discard policy, Routing algorithm and Packet lifetime management

Data Link ⇒ Retransmission policy, out-of-order caching policy, Acknowledgement policy and flow control policy.

Traffic shaping :- As burstiness of traffic is a main cause of congestion, it is used to regulate average rate and burstiness of traffic

Eg:- i, When a virtual circuit is setup, the user and subnet first agree certain traffic shape for that circuit. Monitoring traffic flow, called traffic policing, is left to the subset

ii, Agreeing to a traffic shape and policing it afterward are easier with virtual circuit subnets, but the same ideas can be applied to datagram subnet at transport layer

LEAKY BUCKET ALGORITHM :- It consists of finite queue

→ When a packet arrives, if there is a room on the queue, it is ~~just~~ joined the queue, otherwise it is discarded.

→ At every (fixed) clock tick, one packet is transmitted unless the queue is empty.

→ It eliminates bursts completely, ie, packets passed to the subnet at the same rate

→ This may be a bit overdone and also packets gets lost (when bucket is full)

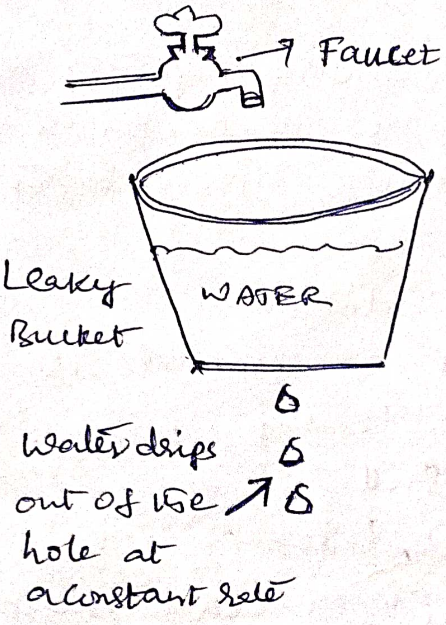


fig. a,

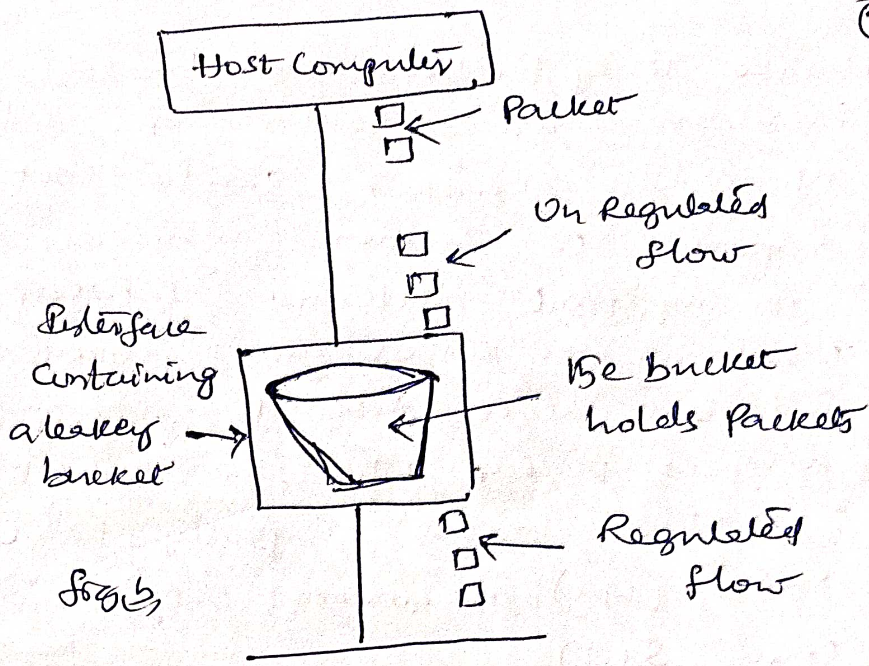


fig. b,

Here fig. a, & fig. b, rep's LEAKY BUCKET ALGORITHM

TOKEN BUCKET :- Tokens are added at a constant rate. For a packet to be transmitted, it must capture and destroy one token

→ shows that the bucket holds three tokens with five packets waiting to be transmitted

→ shows that three packets have gotten through but the other two are stuck waiting for tokens to be generated

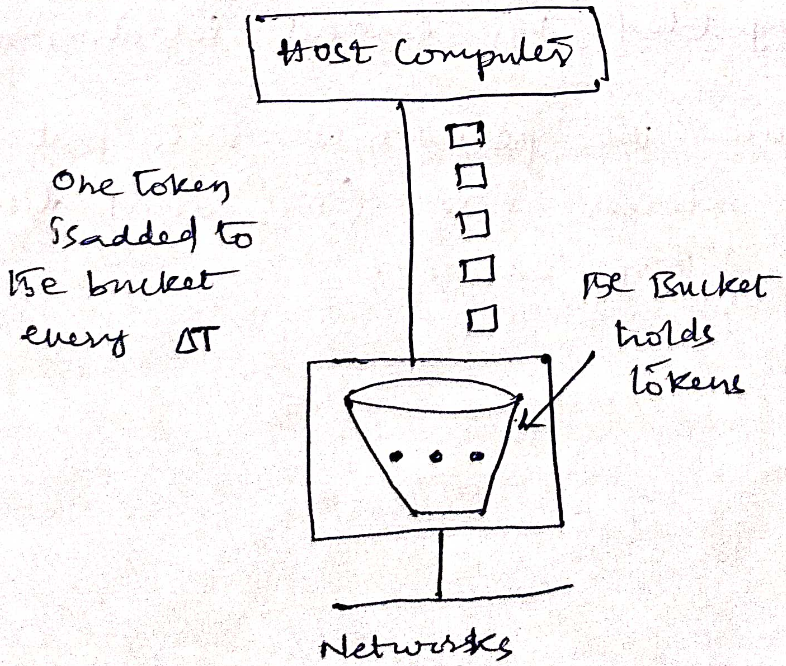


fig. a,

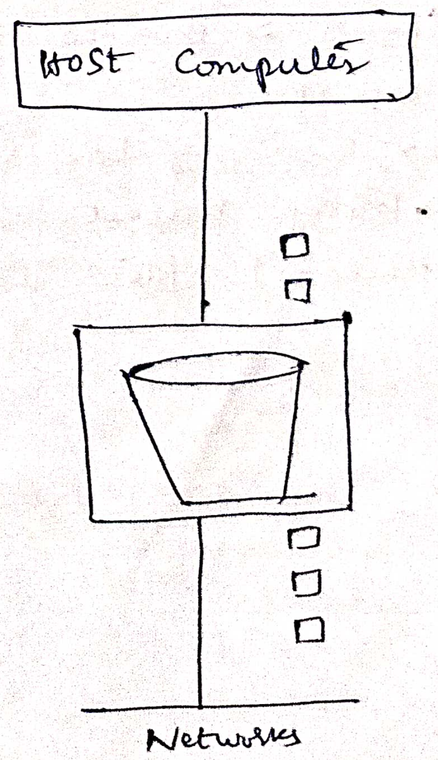


fig. b,

→ Unlike leaky bucket, token bucket allows saving up to maximum size of bucket 'n'. This means that bursts of up to n packets can be sent at once, giving faster response to sudden bursts of input.

→ An important difference between two algorithms: Token bucket throws away tokens when the bucket is full but never discards packets while leaky bucket discards packets when the bucket is full.

→ Let token bucket capacity be C (bits), token arrival rate p (bps) maximum output rate M (bps) and burst length S (s)

→ During burst length of S (s) tokens generated are pS (bits) and output burst contains a maximum of C + pS (bits)

→ Also output in a maximum burst of length S (s) is M.S (bits), thus $C + pS = MS$ or,

$$S = C / M-p$$

→ Token bucket still allows large bursts, even though the maximum burst length S can be regulated by careful selection of p and M

→ One way to reduce the peak rate is to put a leaky bucket of a larger rate (to avoid discarding packets) after the token bucket.