

Computer Networks

BE Computer 7th Sem

Er. Anuj Sherchan

Assistant Professor , Department of Electronics and Computer
Engineering

Pokhara Engineering College

Unit 7 :Congestion Control and Quality of Services

- Outline :
- Congestion Control : Open Loop and Closed Loop
- Traffic Shaping : Leaky Bucket and Token Bucket
- TCP congestion control

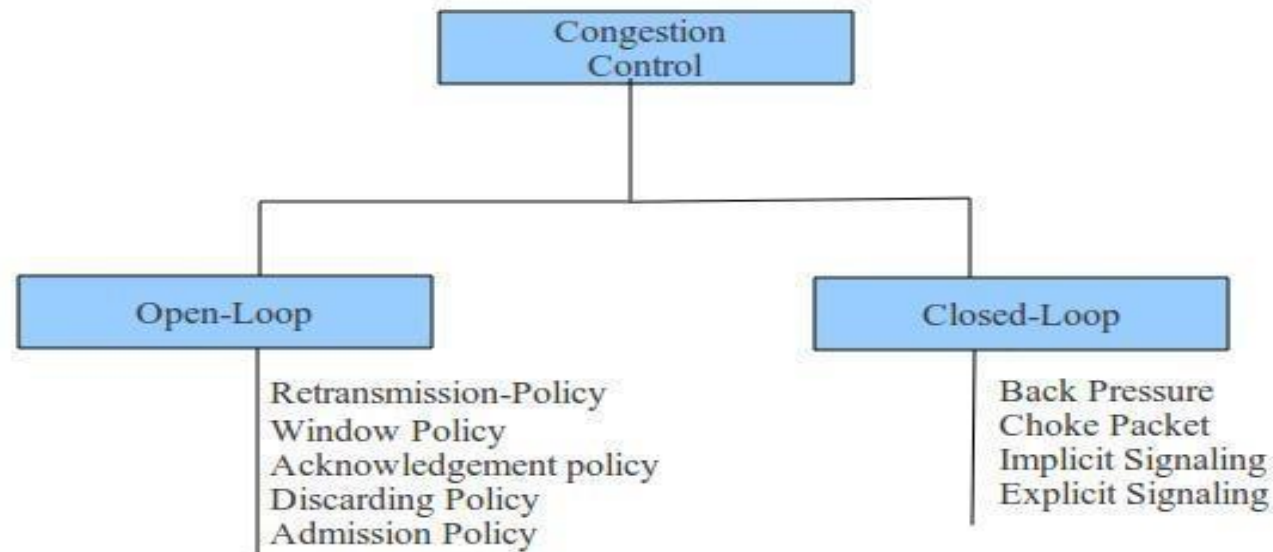
Congestion Control : Open Loop and Closed Loop

- **Congestion**
- 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 control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity .
- When too many packets are pumped into the system, congestion occur leading into degradation of performance.
- Congestion tends to feed upon itself and back ups.
- Congestion shows lack of balance between various networking equipment.
- It is a global issue.

Congestion Control : Open Loop and Closed Loop

- In general, we can divide congestion control mechanisms into two broad categories:
- open-loop congestion control (prevention) and closed-loop congestion control (removal) as shown in Figure

-



Congestion Control : Open Loop and Closed Loop

- **Open Loop Congestion Control:**
- In open-loop congestion control, policies are applied to prevent congestion before it happens. In these mechanisms, congestion control is handled by either the source or the destination
- **Retransmission Policy**
- Retransmission is sometimes unavoidable.
- If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. Retransmission in general may increase congestion in the network.
- However, a good retransmission policy can prevent congestion.
- The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion.
- For example, the retransmission policy used by TCP is designed to prevent or alleviate congestion.

Open Loop Congestion Control

- **Window Policy**
- The type of window at the sender may also affect congestion.
- The Selective Repeat window is better than the Go-Back-N window for congestion control.
- In the Go-Back-N window, when the timer for a packet times out, several packets may be resent, although some may have arrived safe and sound at the receiver.
- This duplication may make the congestion worse.
- The Selective Repeat window, on the other hand, tries to send the specific packets that have been lost or corrupted.

Open Loop Congestion Control

- **Acknowledgment Policy :**
- The acknowledgment policy imposed by the receiver may also affect congestion.
- If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion.
- Several approaches are used in this case.
- A receiver may send an acknowledgment only if it has a packet to be sent or a special timer expires.
- A receiver may decide to acknowledge only N packets at a time.
- We need to know that the acknowledgments are also part of the load in a network.
- Sending fewer acknowledgments means imposing less load on the network.

Open Loop Congestion Control

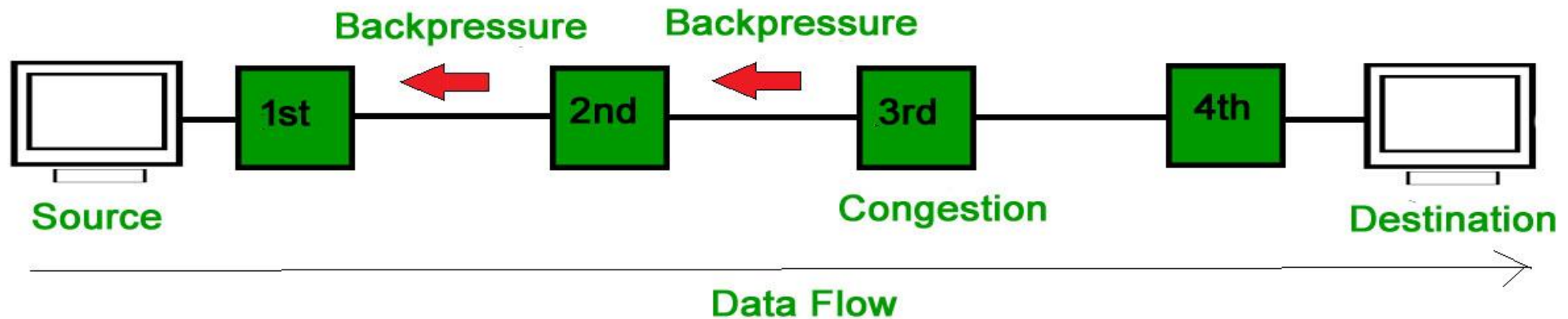
- **Discarding Policy :**
 - A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission.
 - For example, in audio transmission, if the policy is to discard less sensitive packets when congestion is likely to happen, the quality of sound is still preserved and congestion is prevented or alleviated.
- **Admission Policy :**
 - An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual- circuit networks.
 - Switches in a flow first check the resource requirement of a flow before admitting it to the network.
 - A router can deny establishing a virtual- circuit connection if there is congestion in the network or if there is a possibility of future congestion.

Closed Loop Congestion Control

- **Closed-Loop Congestion Control**
- Closed-loop congestion control mechanisms try to alleviate congestion after it happens.
- Several mechanisms have been used by different protocols.
- **Back-pressure:**
- The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes.
- This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes. And so on.

Closed Loop Congestion Control

- Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow, to the source.
- The backpressure technique can be applied only to virtual circuit networks, in which each node knows the upstream node from which a flow of data is coming.



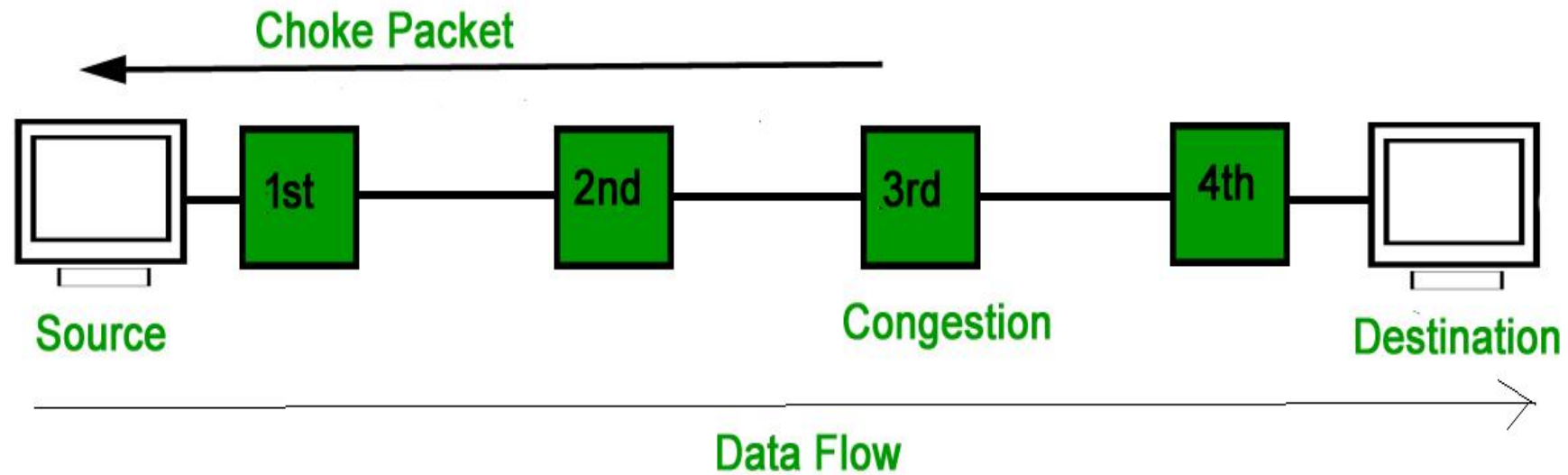
Closed-Loop Congestion Control

- 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**
- A choke packet is a packet sent by a node to the source to inform it of congestion.
- Note the difference between the backpressure and choke packet methods.
- In back pressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station.
- In the choke packet method, the warning is from the router, which has encountered congestion, to the source station directly.

Closed-Loop Congestion Control

- The intermediate nodes through which the packet has traveled are not warned.
- We have seen an example of this type of control in ICMP.
- When a router in the Internet is overwhelmed datagrams, it may discard some of them; but it informs the source . host, using a source quench ICMP message.
- The warning message goes directly to the source station; the intermediate routers, and does not take any action.
- Figure shows the idea of a choke packet.

Choke Packet



Closed-Loop Congestion Control

- **Implicit Signaling**
- In implicit signaling, there is no communication between the congested node or nodes and the source.
- The source guesses that there is a congestion somewhere in the network from other symptoms.
- For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested.
- The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down.
- We will see this type of signaling when we discuss TCP congestion control later in the chapter.

Closed-Loop Congestion Control

- **Explicit Signaling**
- The node that experiences congestion can explicitly send a signal to the source or destination. The explicit signaling method, however, is different from the choke packet method.
- In the choke packet method, a separate packet is used for this purpose; in the explicit signaling method, the signal is included in the packets that carry data.
- Explicit signaling, as we will see in Frame Relay congestion control, can occur in either the forward or the backward direction.
- **Backward Signaling**
- **Forward Signaling**

Closed-Loop Congestion Control

- **Backward Signaling**

- A bit can be set in a packet moving in the direction opposite to the congestion.
- This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets.

- **Forward Signaling**

- A bit can be set in a packet moving in the direction of the congestion.
- This bit can warn the destination that there is congestion.
- The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion.

Traffic Shaping

- **Traffic Shaping**
- Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network.
- Approach of congestion management is called Traffic shaping.
- Traffic shaping helps to regulate rate of data transmission and reduces congestion.
- Two techniques can shape traffic:
- **Leaky bucket and**
- **Token bucket**

Traffic Shaping

- **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 shows a leaky bucket and its effects.

Leaky Bucket Algorithm

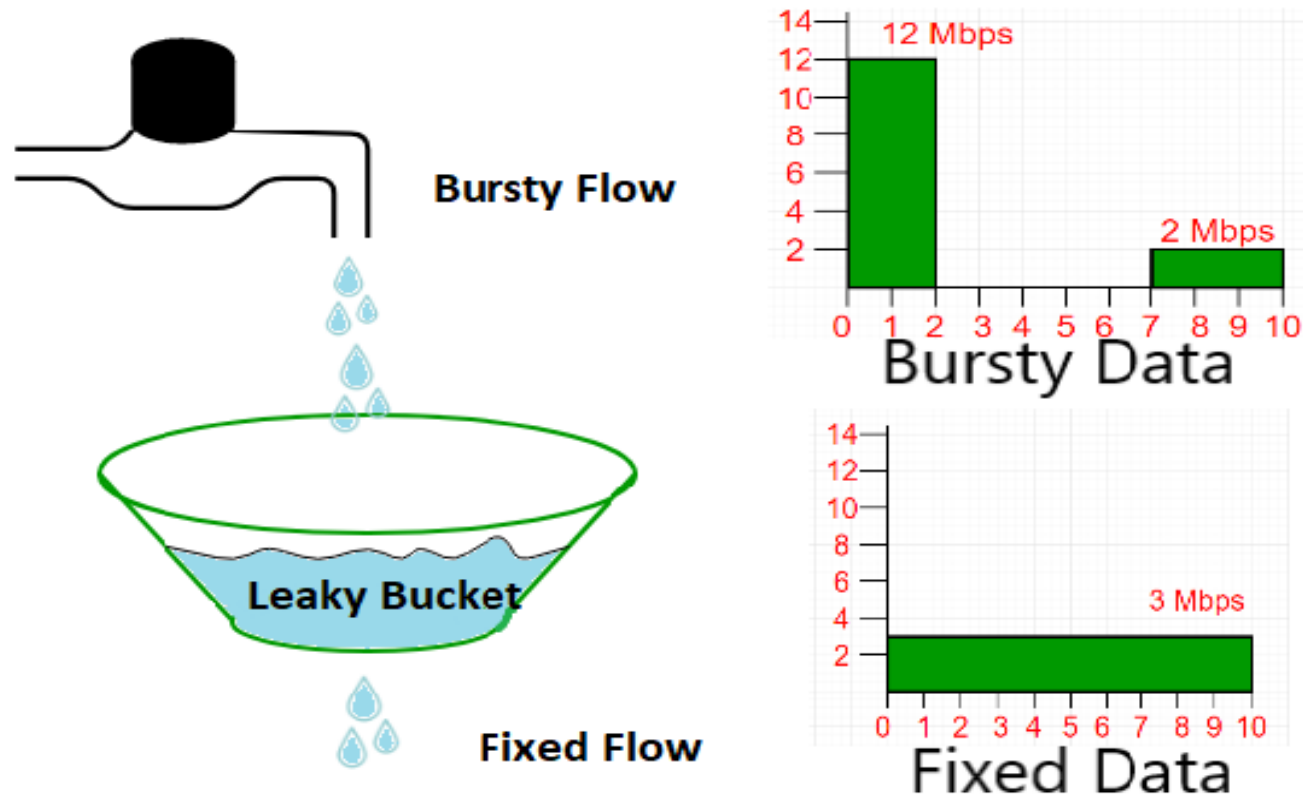


Fig :Leaky Bucket

Leaky Bucket Algorithm

- 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 Mbits 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 Mbits of data.
- In all, the host has sent 30 Mbits of data in 10s.
- The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s.
- Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host.
- We can also see that the leaky bucket may prevent congestion.

Leaky Bucket Algorithm

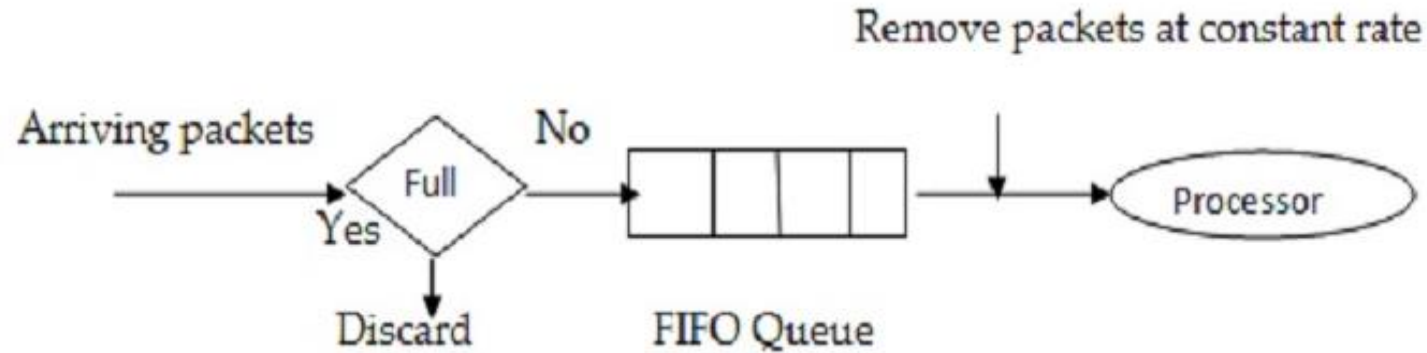


Fig : Leaky Bucket implementation

- A simple leaky bucket implementation is shown in Figure.
- A FIFO queue holds the packets.
- If the traffic consists of fixed-size packets (e.g., cells in ATM networks), 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.

Leaky Bucket Algorithm

- The following is an algorithm for variable-length packets:
 1. Initialize a counter to n at the tick of the clock.
 2. 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.
 3. Reset the counter and go to step 1.
- *A leaky bucket algorithm shapes bursty traffic into fixed-rate traffic by averaging the data rate. It may drop the packets if the bucket is full.*

Token Bucket Algorithm

- Token bucket algorithm is one of the techniques for congestion control algorithms.
- The leaky bucket algorithm enforces output patterns at the average rate, no matter how busy the traffic is.
- So, to deal with the more traffic, we need a flexible algorithm so that the data is not lost.
- One such approach is the token bucket algorithm.

Token Bucket Algorithm

- **Algorithm**
- **Step 1** – In regular intervals tokens are thrown into the bucket f .
- **Step 2** – The bucket has a maximum capacity f .
- **Step 3** – If the packet is ready, then a token is removed from the bucket, and the packet is sent.
- **Step 4** – Suppose, if there is no token in the bucket, the packet cannot be sent

Token Bucket Algorithm

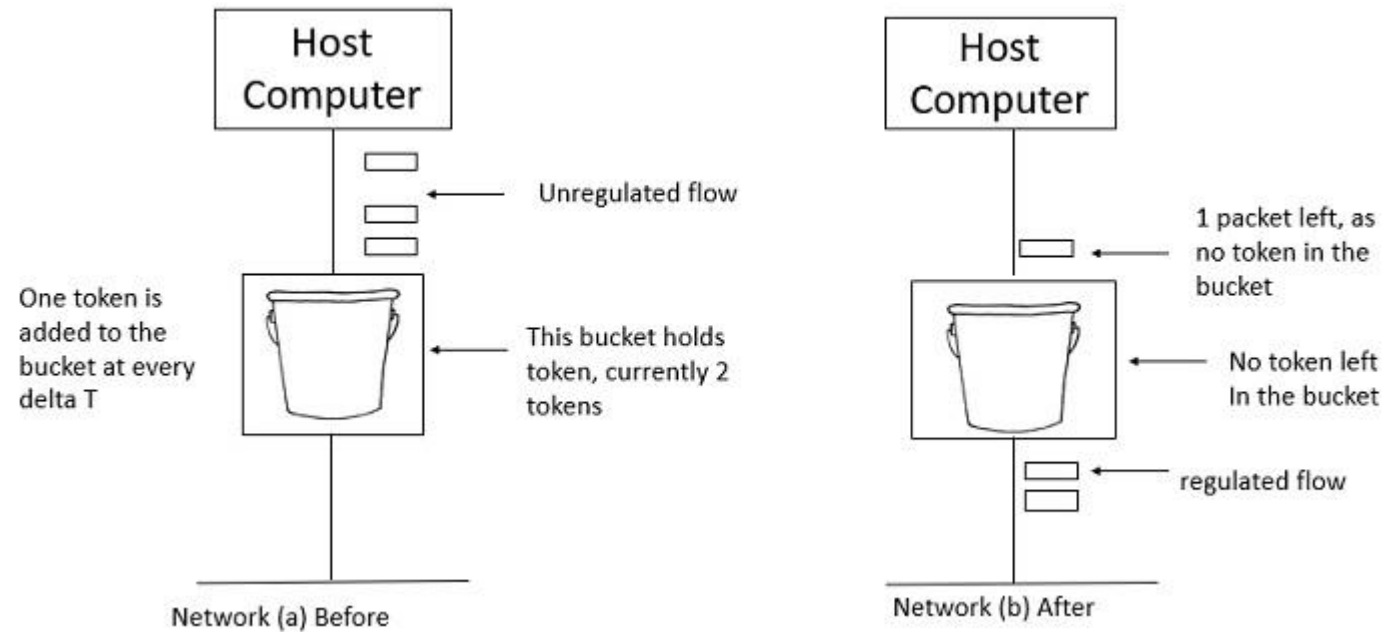


Fig : Token bucket implementation

Token Bucket Algorithm

- In figure (a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface.
- In Figure (b) two packets have been sent out by consuming two tokens, and 1 packet is still left.
- When compared to Leaky bucket the token bucket algorithm is less restrictive that means it allows more traffic.
- The limit of busyness is restricted by the number of tokens available in the bucket at a particular instant of time.
- The implementation of the token bucket algorithm is easy – a variable is used to count the tokens.
- For every t seconds the counter is incremented and then it is decremented whenever a packet is sent.
- When the counter reaches zero, no further packet is sent out.

Leaky Bucket vs Token Bucket Algorithm

Parameter	Leaky Bucket	Token Bucket
Token Dependency	Token independent.	Dependent on Token.
Filled bucket for token	When bucket is full, data or packets are discarded.	If bucket is full, token are discard not packets.
Packet transmission	Leaky bucket sends packets at constant rate.	Token bucket can send large burst of packets at faster rate.
Condition for packet transmission	In Leaky bucket algorithm, Packets are transmitted continuously.	In Token bucket algorithm, Packets can only transmit when there is enough token.
Token saving	It does not save any token.	It saves token for the burst of packet transmission.
Restrictive Algorithm	Leaky bucket algorithm is more restrictive as compared to Token bucket algorithm.	Token bucket algorithm is less restrictive as compared to Leaky bucket algorithm.

THANK YOU