Chapter 7
Congestion Control and Quality of services

Congestion Control

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.

In general, we can divide congestion control mechanisms into two broad categories:

1. open-loop congestion control (prevention) and
2. closed-loop congestion control (removal)

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.

1. 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.

2. 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.

3. 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 fewer loads on the network.

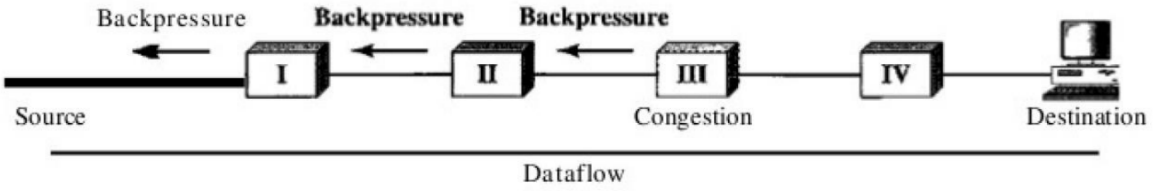
Closed-Loop Congestion Control

Closed-loop congestion control mechanisms try to alleviate congestion after it happens.

Several mechanisms have been used by different protocols.

1. 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. 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 corning.

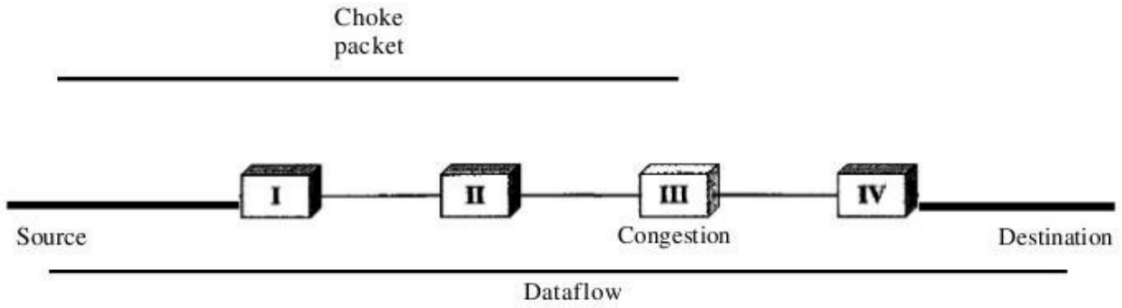


Node III in the figure has more input data than it can handle. It drops some packets in its input buffer and informs node II to slow down. Node II, in turn, may be congested because it is slowing down the output flow of data. If node II is congested, it informs node I to slow down, which in turn may create congestion. If so, node I inform the source of data to slow down. This, in time, alleviates the congestion.

2. 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 backpressure, 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. The intermediate nodes through which the packet has travelled are not warned. When a router in the Internet is overwhelmed datagrams, it may discard some of them; but it informs the source, host. 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.



3. Implicit Signaling

In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is 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.

4. 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.

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.

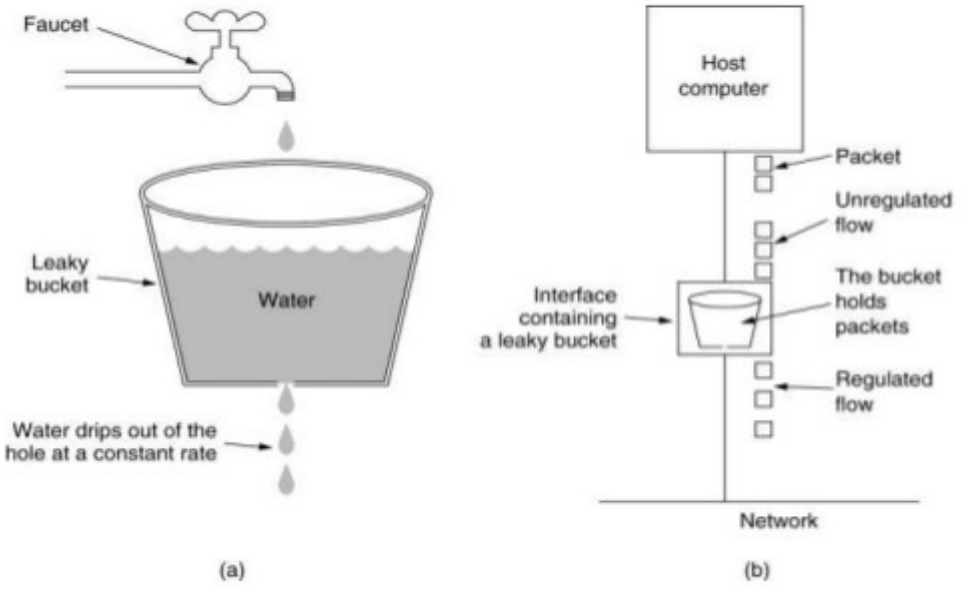
Traffic Shaping

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

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.

The Leaky Bucket Algorithm

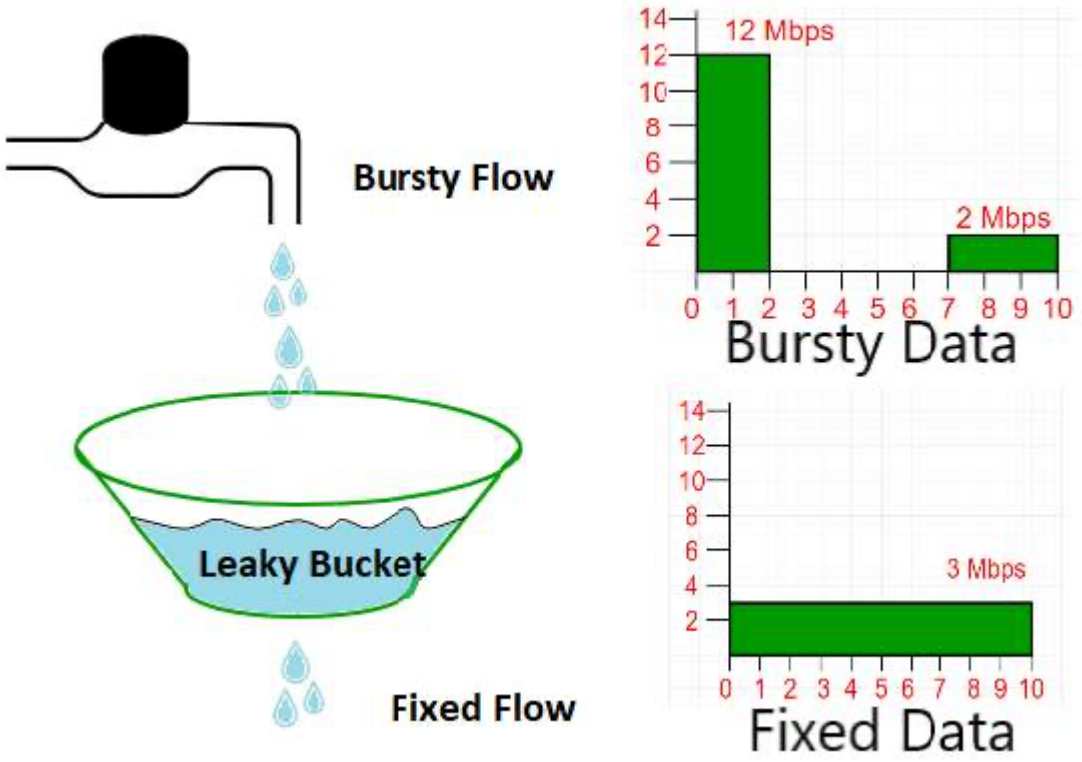


(a) A leaky bucket with water.

(b) a leaky bucket with packets.

Illustration (a) shows how an actual leaky bucket works. The water leaks out of the bucket at a constant rate independent of the water that comes from the faucet.

Illustration (b) shows the transportation of packets using the Leaky Bucket Algorithm. The host computer sends out a packet of data (usually unregulated), it passes through an interface where the Leaky Bucket is situated, this bucket holds the packets in and begins to release them in a regulated manner to the network. It helps to ensure that the network isn't laden with packets. An overloaded network often leads to packet loss.



The following steps are performed:

Step 1. When the host has to send a packet, the packet is thrown into the bucket.

Step 2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.

Step 3. Bursty traffic is converted to a uniform traffic by the leaky bucket.

Step 4. In practice the bucket is a finite queue that outputs at a finite rate.

Step 5. 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:

(i) Initialise a counter to n at the tick of the clock.

(ii) It 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.

(iii) Reset the counter and go to step 1.

In leaky bucket algorithms, there are chances of loss of packets as the packet is filled in the bucket and overflow if the bucket is full. To minimize such limitation of packets, Token Bucket Algorithm is introduced. Here the bucket holds a token not a packet. Tokens are generated by clocks at the rate of one token per clock tick second. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero the host cannot send data.

This algorithm follows the following steps:

Step 1. In regular intervals tokens are thrown into the bucket.

Step 2. The bucket has a maximum capacity.

Step 3. If there is a ready packet, a token is removed from the bucket and the packet can be send.

Step 4. If there is no token in the bucket, the packet cannot be send.

In Fig. (a), token bucket holding two tokens, before packets are send out.

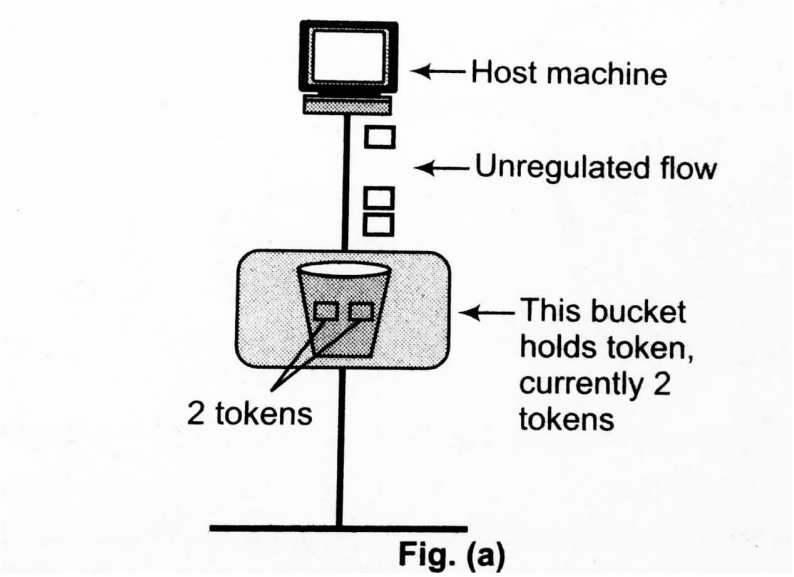
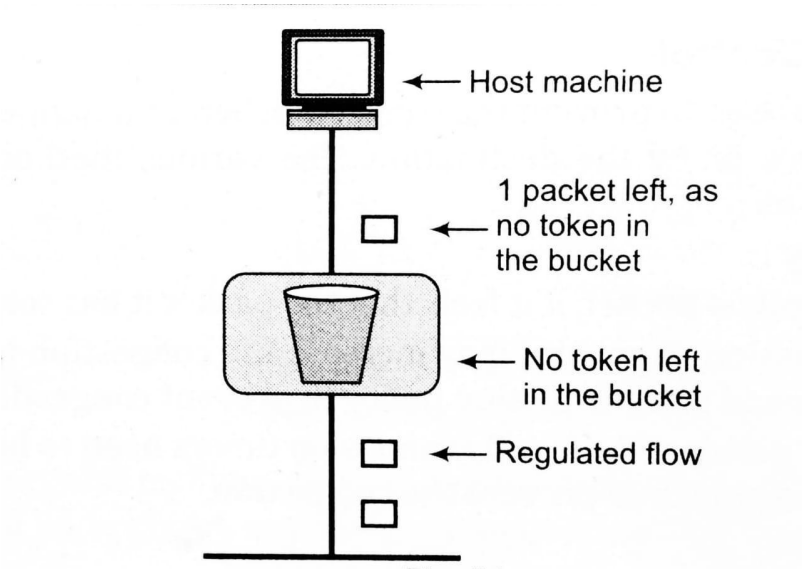


Fig. (a)

When token bucket after two packets are send. One packet still remains as no token is left. As shown in Fig(b).



Now, if the host wants to send bursty data, it can consume all 10,000 tokens at once for sending 10,000 cells or bytes. Thus, a host can send bursty data as long as bucket is not empty. As shown in Fig(c).

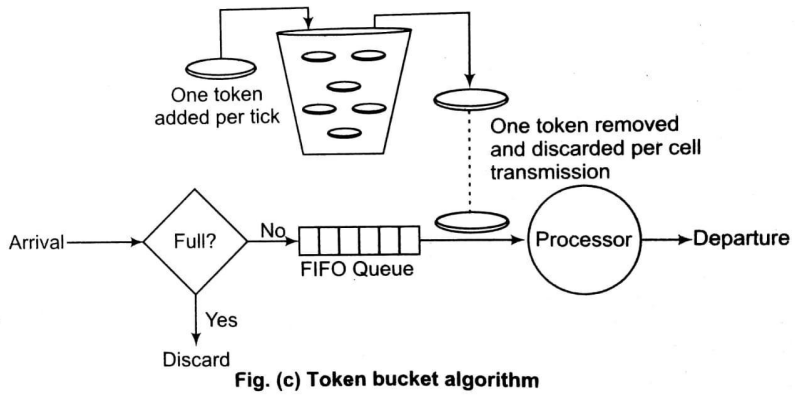


Fig. (c) Token bucket algorithm