Network Layer

Network Layer

- Network layer
  - design issues - services to the transport layer
  - Routing algorithms-
    - adaptive, non adaptive algorithms,
    - optimality principle,
    - Dijkstra's - shortest path routing algorithm,
    - flow based routing, flooding, hierarchical routing,
    - congestion control algorithms –
      - the leaky bucket algorithm, the token bucket algorithm.

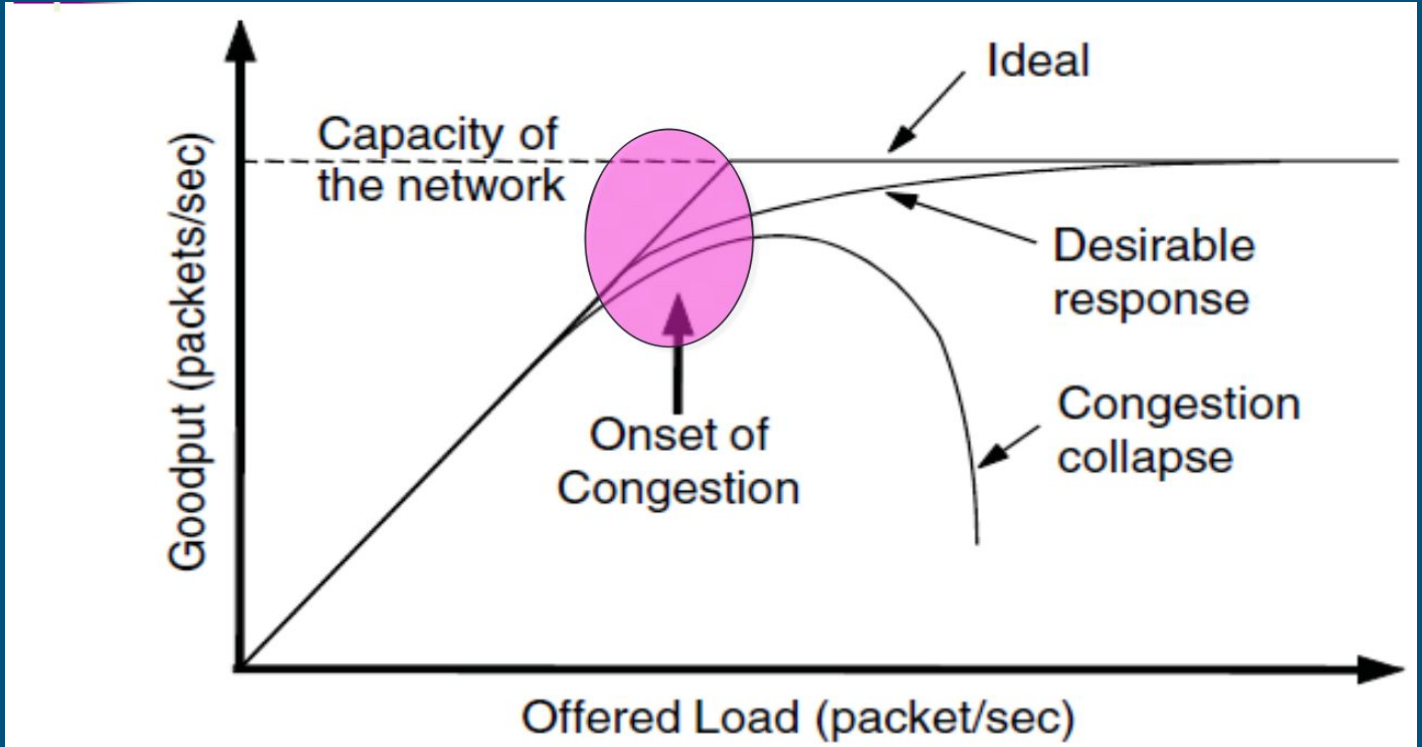
CONGESTION CONTROL ALGORITHMS

Congestion Control Intro.....

- When number of packets sent is within subnet carrying capacity, all are delivered
- As traffic increases, packet loss happens
- At very high traffic, performance collapses

- Too many packets present in (a part of) the network causes
  - packet delay and
  - loss that degrades performance.
    - This situation is called congestion
- The network and transport layers share the responsibility for handling congestion.
- Network layer is directly affected

With too much traffic, performance drops sharply.



Goodput: rate at which *useful* packets are delivered by the network.

How Congestions Happens

- Incoming packets from multiple inputs need to go to same output line;
  - queue builds up - congestion happens
- If insufficient memory, packets are lost.
  - Adding memory helps to some point
  - Even with enough memory, congestion gets worse
    - delayed packets timeout, retransmitted
    - duplicates increase load
- If the network load exceeds capacity congestion happens

- **Slow processors**
  - **CPU slow in doing bookkeeping tasks**
  - **queues build up**
- **Low bandwidth lines**
  - **can’t forward packets at the same arriving speeds**
- **Mismatch between system parts**
  - **upgrading some parts only shifts bottleneck**

Congestion control vs Flow control

- Congestion control
  - make sure subnet is able to carry offered traffic
  - global, involve behavior of all hosts
  - all factors that diminish carrying capacity
- Flow control
  - traffic between a given sender & given receiver
  - ensure fast sender not overwhelm slow receiver
  - involve feedback from receiver to sender

Example: Congestion VS Flow Control

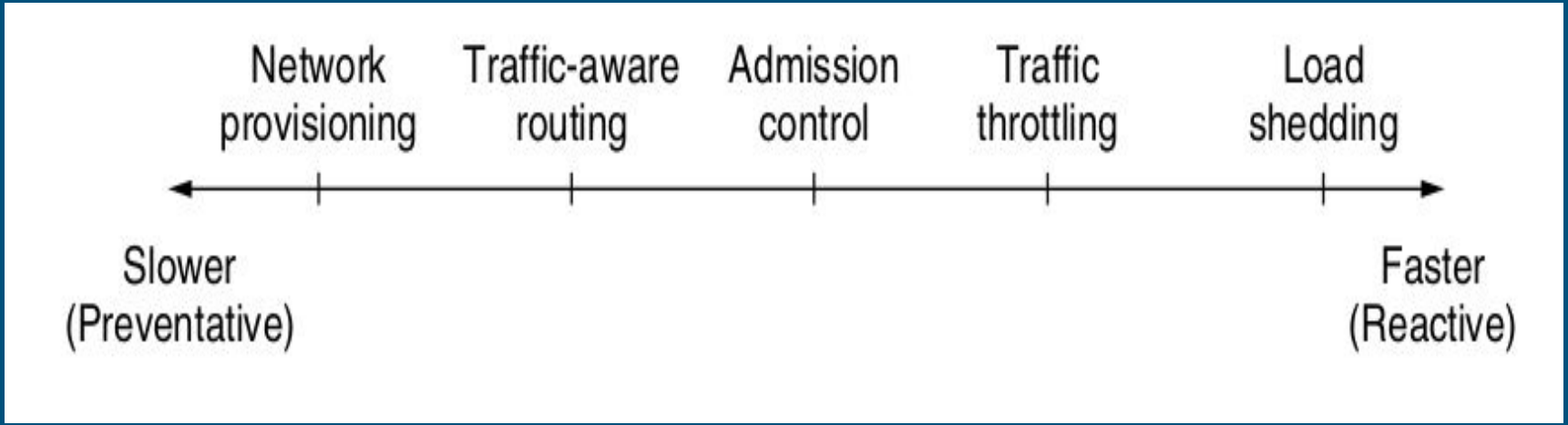
- Flow control
  - fiber optic network with 1000 Gbps
  - Source Computer try to transfer file to a PC @ 1Gbps
  - no congestion
  - flow control needed to slow Source Ccomputer
- Congestion control
  - network with 1 Mbps lines, 1000 computers
  - half of them trying to transfer @ 100 kbps
  - no overpowering problem
  - but total traffic exceed network capacity

Approaches to Congestion Control

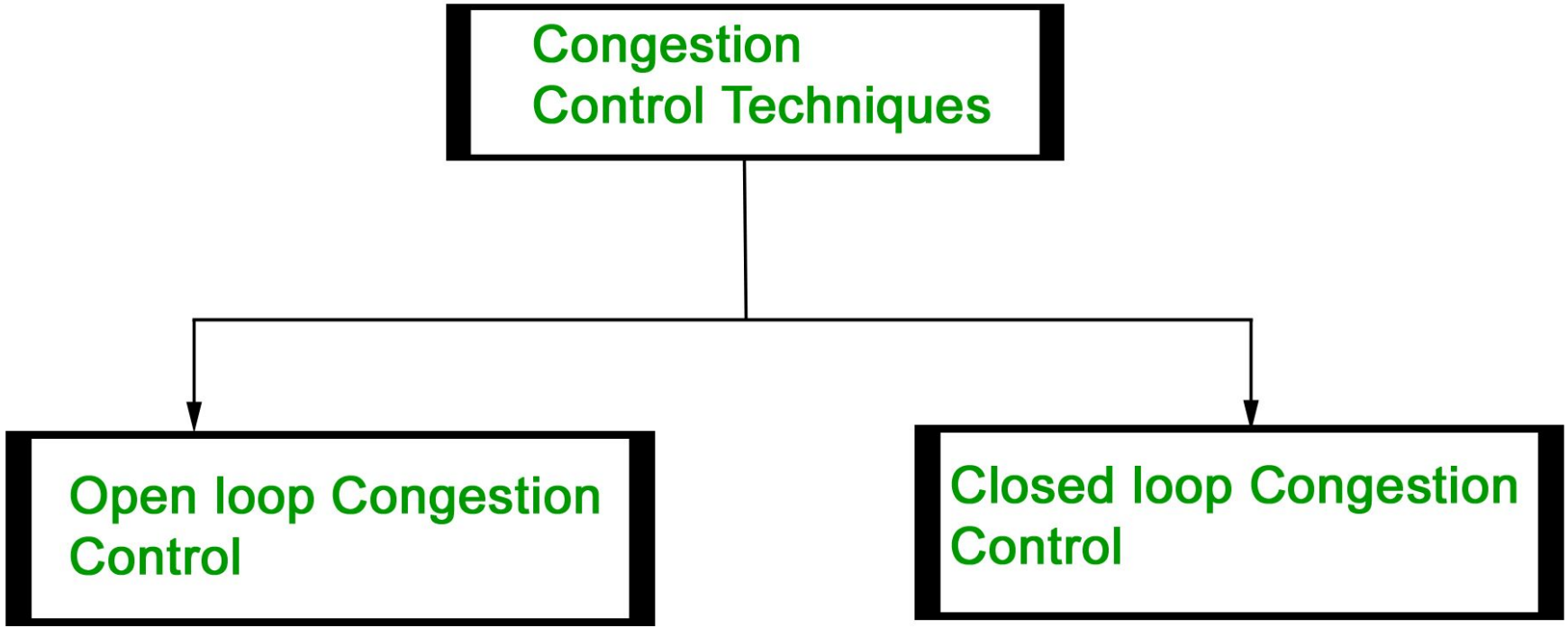
Congestion happens when *load (temporarily) > available resources*

- **Two solutions are:**
  - **increase the resources**
  - **decrease the load.**
- **these solutions are usually applied on different time scales**
  - **To either <u>prevent</u> congestion or**
  - **To <u>react</u> to it once it has occurred**

Solutions are applied in different time scales



Congestion control techniques can be broadly classified into two categories:



- **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.

- **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;

Common approaches/techniques for congestion control are.....

Network provisioning

Traffic-aware routing

Admission control

Traffic throttling

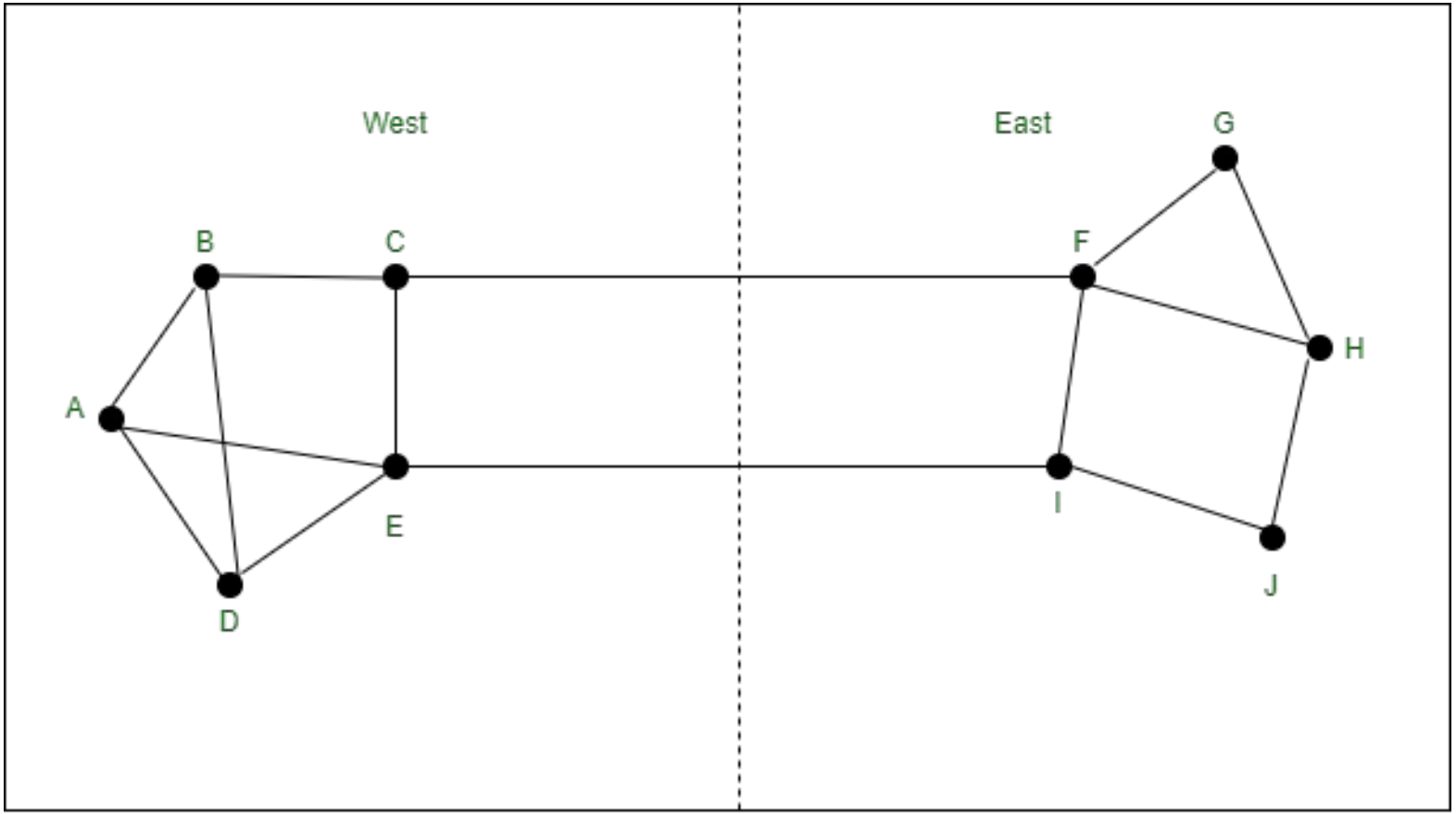
Load shedding

Network provisioning

- The most basic way to avoid congestion is to **build a network that is well matched to the traffic** that it carries.
- Bcos, If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely.
- Sometimes resources can be added dynamically
  - for example, turning on spare routers
  - enabling lines that are normally used only as backups
  - purchasing bandwidth
- More often, **links and routers** that are regularly heavily utilized are **upgraded**
- This is called *Network Provisioning* and happens on a timescale of months, driven by long-term traffic trends.

Traffic-aware routing.

- To make the most of the existing network capacity, routes can be updated to traffic patterns that change during network usage
- Traffic-aware routing adapted to change in topology but not


West
East
A
B
C
D
E
F
G
H
I
J

- Suppose most traffic is between east and west use connection CF the result in this connection is heavily loaded with a long delay.
    a. Even for shortest path calculation, the EI makes it more attractive.
- After New routing tables have been installed, most of the traffic will go over EI
- in the next update, CF will appear to be the shortest path

- As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems
- Two techniques can contribute to a successful solution.
  - The first is *multipath routing*, in which there can be multiple paths from a source to a destination.
  - The second one is for the routing scheme to *shift traffic* across routes slowly enough that it is able to converge
- Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called traffic engineering

Features :

- It is a congestion technique.
- In these routes can be altered because these traffic patterns change during network usage.
- As there are heavily used paths, routes can be changed to shift traffic away.
  a. For example, routes may be changed to shift traffic away from heavily used paths by changing the shortest path weights.
- This is called **traffic-aware routing**
- Splitting traffic across multiple paths is also helpful.

Admission Control

- This is the scheme used in VC subnets for congestion control.
- In this scheme,
  - it does not set up a new virtual circuit
    - unless the network can carry the added traffic without becoming congested.

- Traffic is often described in terms of its rate and shape.
  - The traffic that varies while browsing the Web is more difficult to handle than a streaming movie with the same long-term throughput because the bursts of Web traffic are more likely to congest routers in the network.
- A commonly used descriptor that captures this effect is the leaky bucket or token bucket
- Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure.
- For example, consider the network...

Here two routers are congested ……..

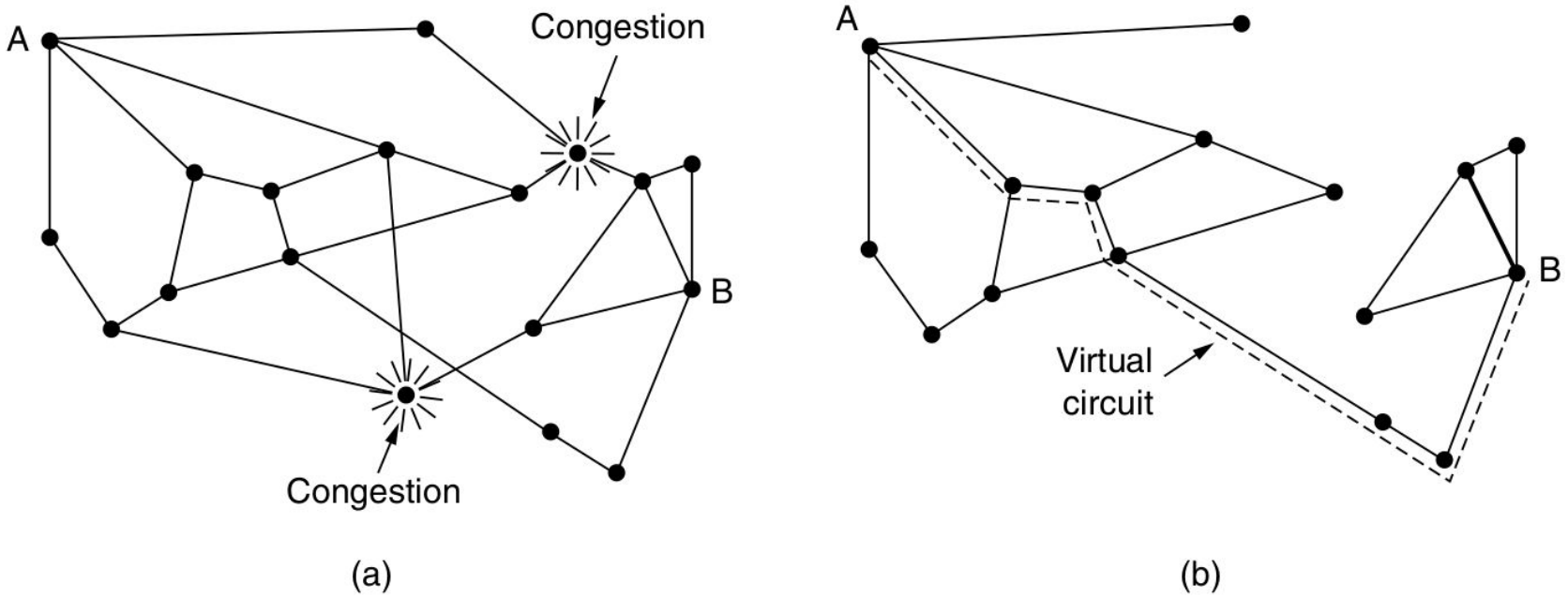


Figure 5-24. (a) A congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

- **Suppose that a host attached to router A wants to set up a connection to a host attached to router B.**
- **Normally, this connection would pass through one of the congested routers.**
- **To avoid this situation, we can redraw the network by omitting the congested routers and all of their lines.**
- **The dashed line in the figure shows a possible route for the virtual circuit that avoids the congested routers.**
- **It is a kind of *load-sensitive routing*.**

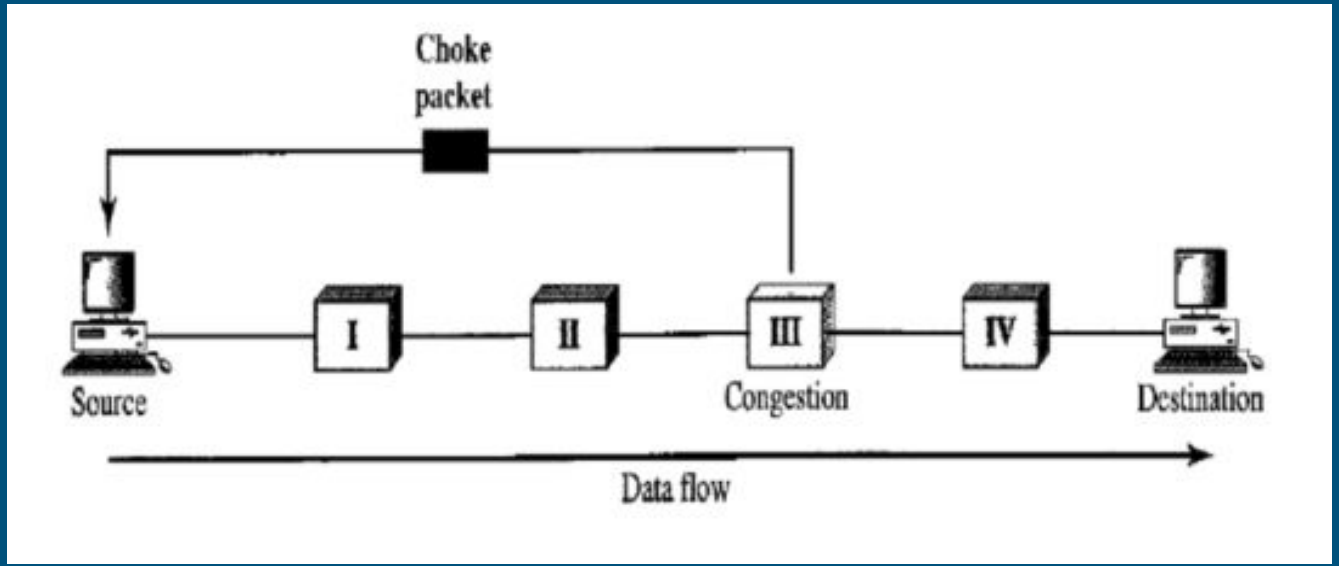
Traffic Throttling

- When congestion is imminent, it must tell the senders to throttle back their transmissions and slow down
- Approaches to throttling traffic that can be used in both datagram networks and virtual-circuit networks.

- **Each approach must solve two problems.**
  - **First, routers must determine when congestion happens.**
  - **To do so, each router can continuously monitor the resources it is using.**
  - **Three possibilities are**
    - **the utilization of the output links,**
    - **the buffering of queued packets inside the router,**
    - **the number of packets that are lost due to insufficient buffering.**

- Second problem is that routers must deliver timely feedback to the senders that are causing the congestion.
- To deliver feedback, the router must identify the appropriate senders.
- It must then warn them carefully, without sending many more packets into the already congested network.
- Different schemes use different feedback mechanisms

- ***Choke Packets***
  - **In this approach, the router sends a choke packet back to the source host**
  - **The original packet may be tagged (a header bit is turned on) so that it will not generate any more choke packets along the path and then forwarded in the usual way.**
  - **To avoid increasing load on the network during a time of congestion, the router may only send choke packets at a low rate.**
  - **When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination**


Choke
packet
I
II
III
IV
Source
Congestion
Destination
Data flow

- The source node is addressed directly by the router, forcing it to decrease
- ... through which the packets has traveled are not
- warned about congestion.

- **Explicit Congestion Notification**
  - Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion.
  - When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet.
  - The sender can then throttle its transmissions as before.
  - This design is called ECN (Explicit Congestion Notification) and is used in the Internet

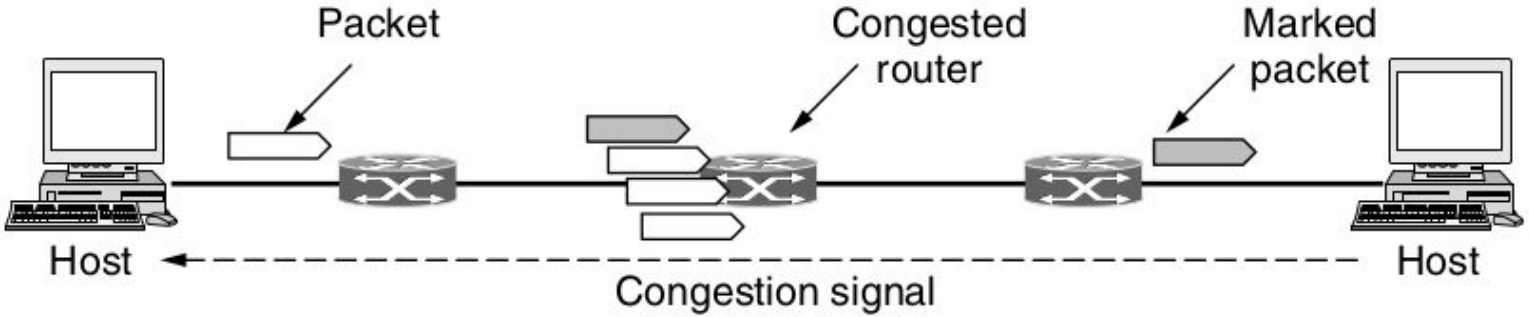


Figure 5-25. Explicit congestion notification

- *Hop-by-Hop Backpressure*
  - This technique is an advancement over the Choked packet technique and helps alleviate congestion even before the choke packet arrives at the source.

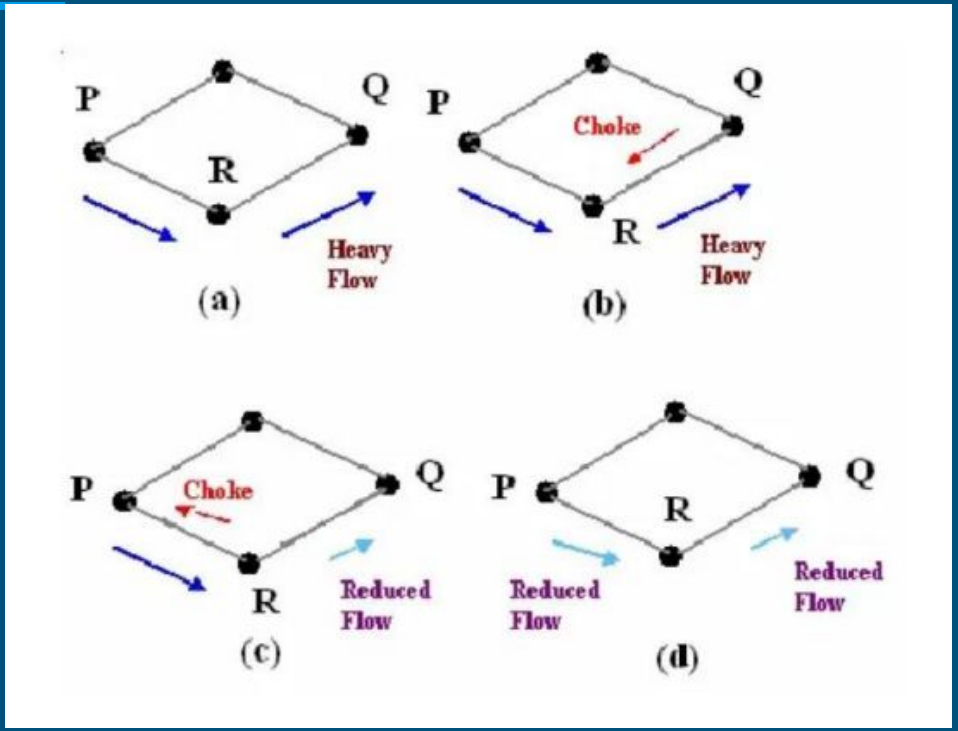
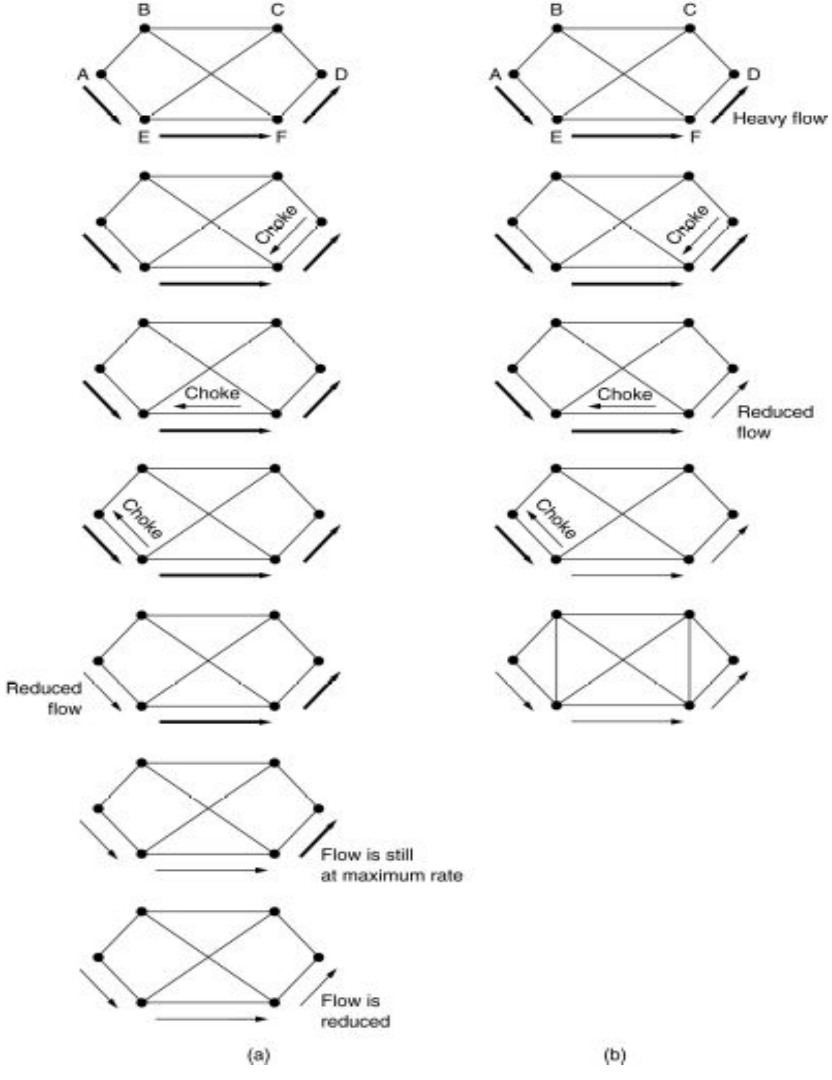


Figure Depicts the functioning of choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to node P, (c) Choke packet reaches P and throttles down, between P and Q is (d) The Choke packet reaches P and Produces the flow cut.

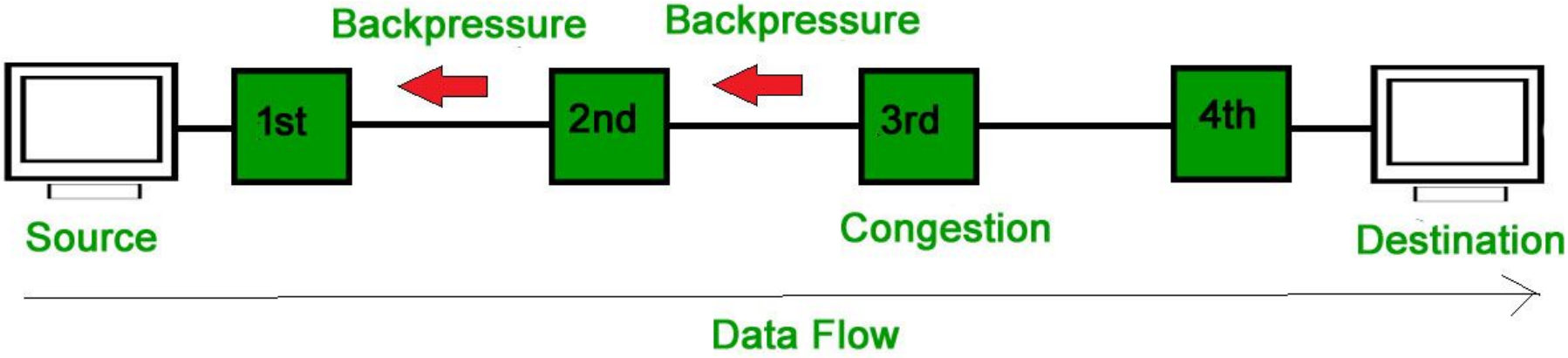
Hop-by-Hop Choke Packets



(a) A choke packet that affects only the source.

(b) A choke packet that affects each hop it passes through.

- *Backpressure*
  - Backpressure is a technique in which a congested node


Backpressure
Backpressure
1st
2nd
3rd
4th
Source
Congestion
Destination
Data Flow

Load Shedding

- When none of the above methods make the congestion disappear, routers can use
  - load shedding.
- When routers are being inundated by packets that they cannot handle, they just throw them away.
- To implement an intelligent discard policy, applications must mark their packets to indicate to the network how important they are.
- Then, when packets have to be discarded, routers can first drop packets from the least important class, then the next most important class, and so on.

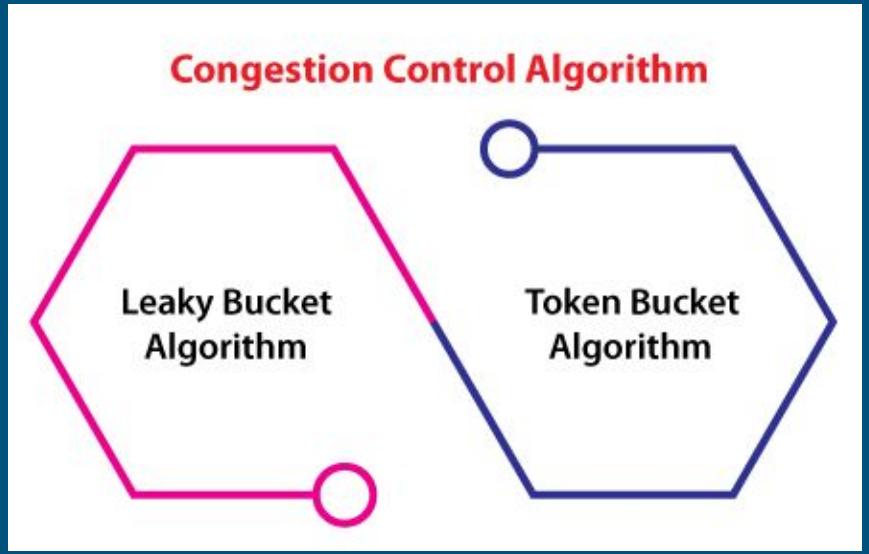
Quality of Service

- Before the network can make QoS guarantees, it must know what traffic is being guaranteed
- Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network.
- The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network

- When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow.
- Sometimes this agreement is called an SLA (Service Level Agreement),
  - especially when it is made over aggregate flows and long periods of time
- Packets in excess of the agreed pattern might be dropped by the network, or they might be marked as having lower priority.

- Monitoring a traffic flow is called **traffic policing.**
- Shaping and policing are not so important for peer-to-peer and other transfers that will consume any and all available bandwidth,
  - but they are of great **importance for real-time data, such as audio and video connections**, which have stringent quality-of-service requirements.

more general way to characterize traffic, with the leaky bucket and token bucket algorithms,
The formulations are slightly different but give an equivalent result.



Leaky Bucket Algorithm

- Leaky Bucket Algorithm mainly controls the total amount

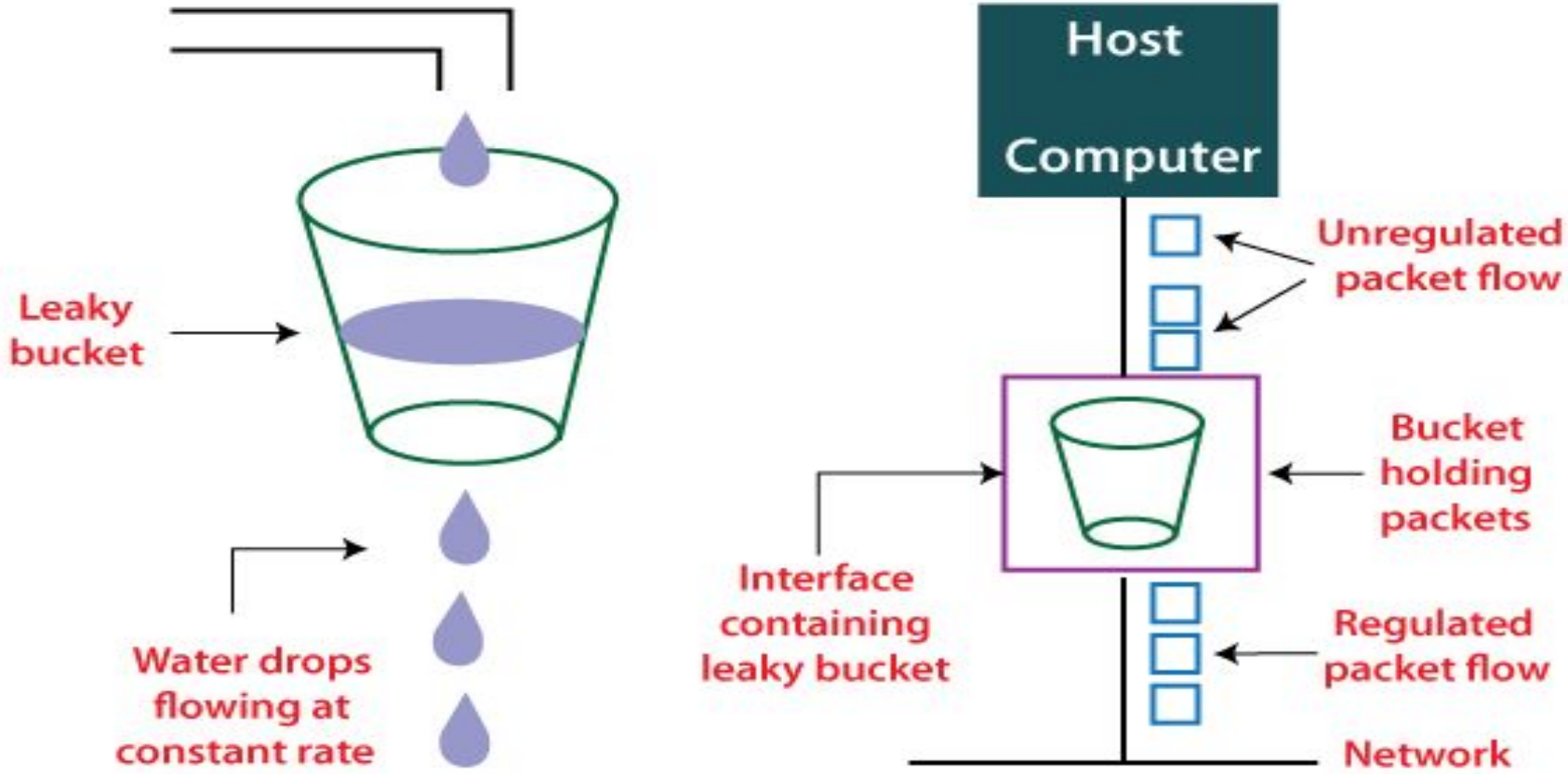
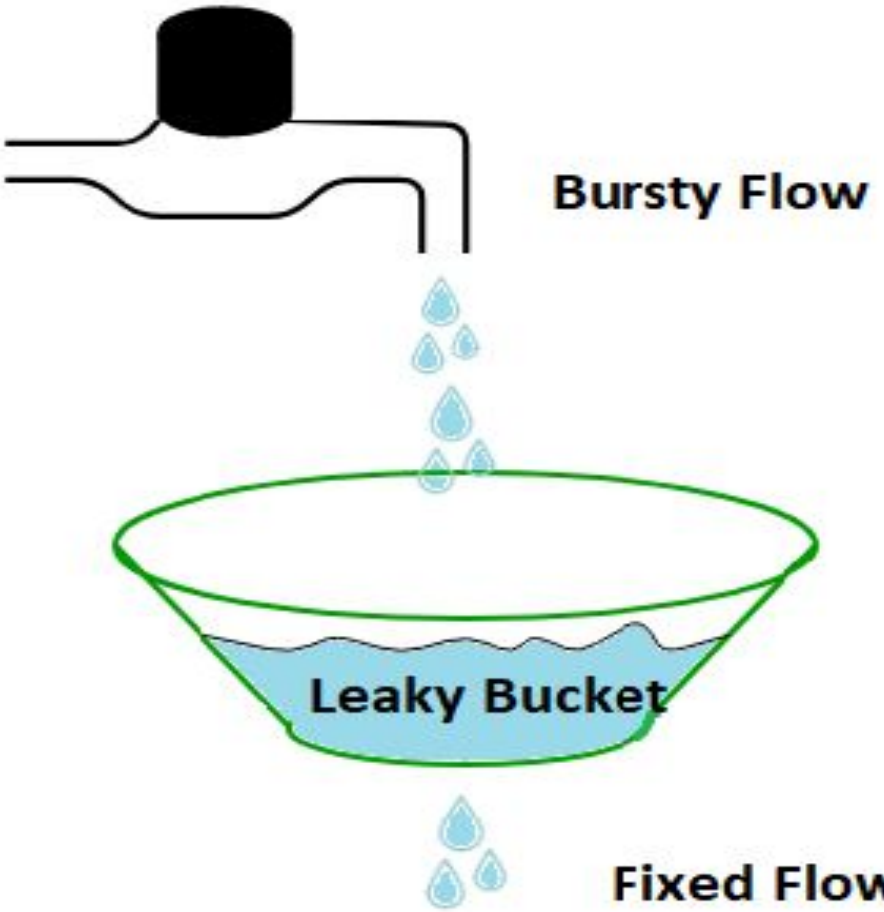
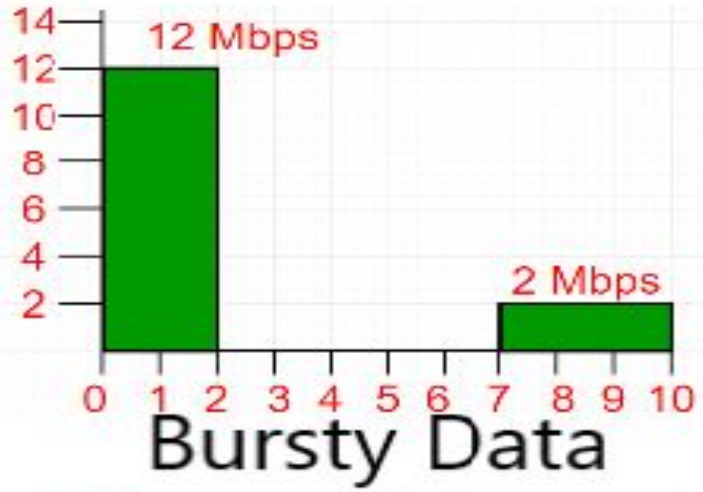
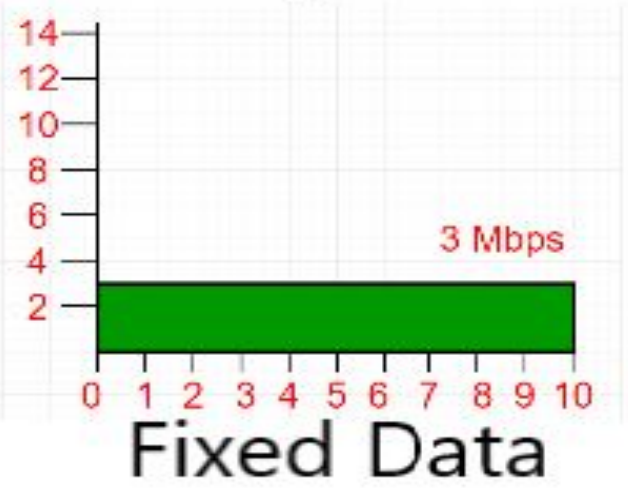


Fig: Leaky Bucket Algorithm


Bursty Flow
Leaky Bucket
Fixed Flow


14
12
10
8
6
4
2
12 Mbps
2 Mbps
0 1 2 3 4 5 6 7 8 9 10
Bursty Data


14
12
10
8
6
4
2
3 Mbps
0 1 2 3 4 5 6 7 8 9 10
Fixed Data

In the above figure, we assume that the network has a bandwidth of 3 Mbps for a host.
Take use of the leaky bucket shapes the input traffic to make it conform to this bandwidth.
In the above figure, the host sends a burst of data at a rate of 12 Mbps for 2s, 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 10 s.

The leaky bucket enforces an average output rate of 3 Mbps during the 10 s.

Without the leaky bucket, the beginning burst may have sent data at a rate of 10 Mbps during the first second.

In practice, bursts of traffic are normally followed by periods in which little traffic is sent.

We can also see that the leaky bucket may prevent congestion.

- A simple leaky bucket algorithm can be implemented using FIFO queue. A FIFO queue is checked to see if there is a packet to transmit **at each tick of the clock**. If there is a packet in the queue, the packet is removed from the queue and sent into the network.

- If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes

The following is an algorithm for variable-length packets:

1. Initialize a counter to n at the tick of the clock.
2. Repeat until n is smaller than the packet size of the packet at the head of the queue.
   1. Pop a packet out of the head of the queue, say P.
   2. Send the packet P, into the network
   3. Decrement the counter by the size of packet P.
3. Reset the counter and go to step 1.

This algorithm is also known as **Byte-counting leaky bucket algorithm**

Example: Let n=1000

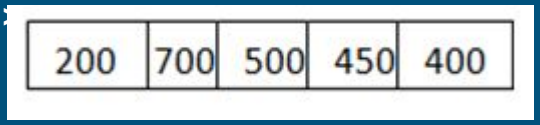
Packet=

Since n > size of the packet at the head of the Queue, i.e. n > 200

Therefore, n = 1000-200 = 800

Packet size of 200 is sent into the network.

Now, again n > size of the packet at the head of the Queue, i.e. n >



Therefore, n = 800-400 = 400

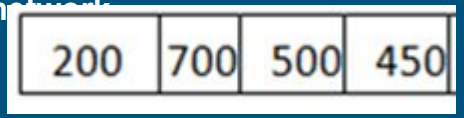
Packet size of 400 is sent into the network.

Since, n < size of the packet at the head of the Queue, i.e. n < 450

Therefore, the procedure is stopped.

Initialise n = 1000 on another tick of the clock.

This procedure is repeated until all the packets are sent into the network.

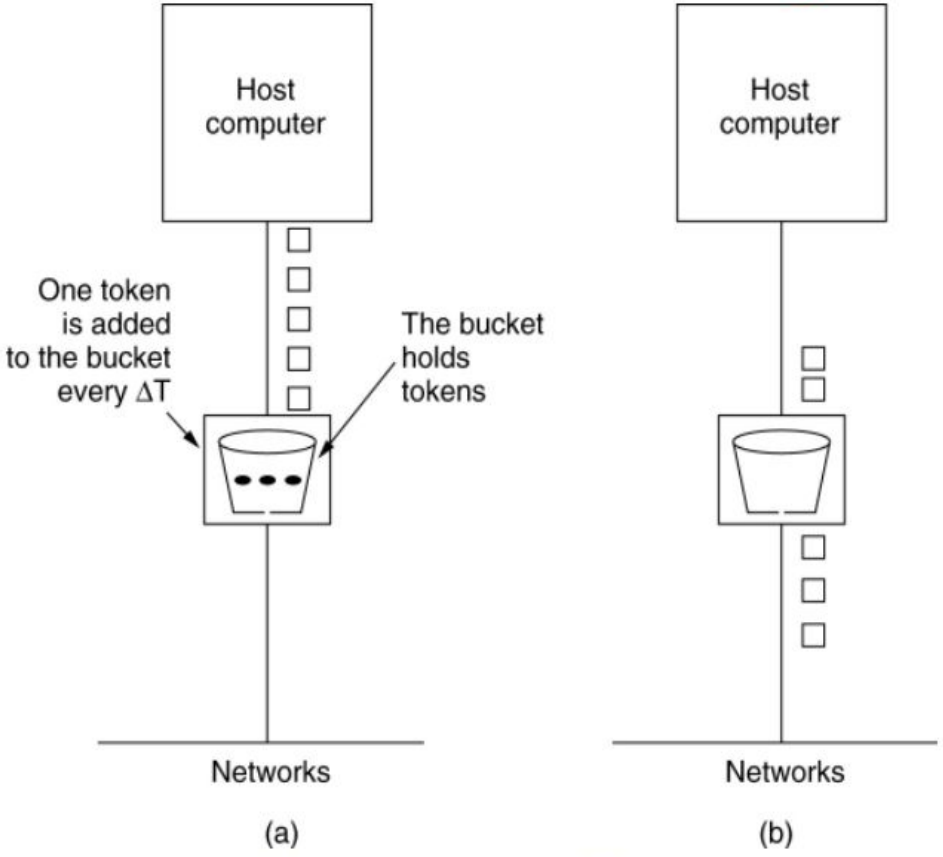


Token Bucket Algorithm

- The token bucket algorithm is based on an analogy is the token bucket algorithm.

- In this algorithm the leaky bucket holds TOKENS
- Tokens are generated at the rate of one token for every ΔT sec

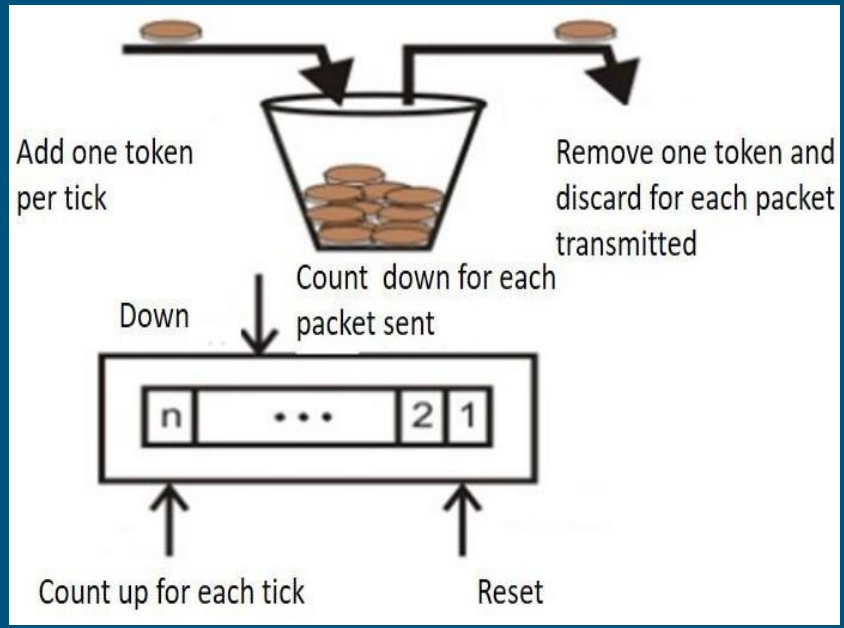
The Token Bucket Algorithm



(a) Before. (b) After.

- The bucket holds 3 tokens, five packets waiting to be transmitted,
- For a packet to be transmitted , it must capture and destroy one token
- In fig (b) three packet transmitted , other two waiting for 2 more tokens to be generated.

- 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. The counter is reset for further transmit of packets
- In the byte count variant, the counter is incremented by k bytes every ΔT sec and decremented by the length of each packet sent.



Difference between Leaky and Token buckets –

Leaky Bucket	Token Bucket
When the host has to send a packet , packet is thrown in bucket.	In this, the bucket holds tokens generated at regular intervals of time.
Bucket leaks at constant rate	Bucket has maximum capacity.
Bursty traffic is converted into uniform traffic by leaky bucket.	If there is a ready packet , a token is removed from Bucket and packet is send.
In practice bucket is a finite queue outputs at finite rate	If there is no token in the bucket, then the packet cannot be sent.