# Computer Network(CSC 503)

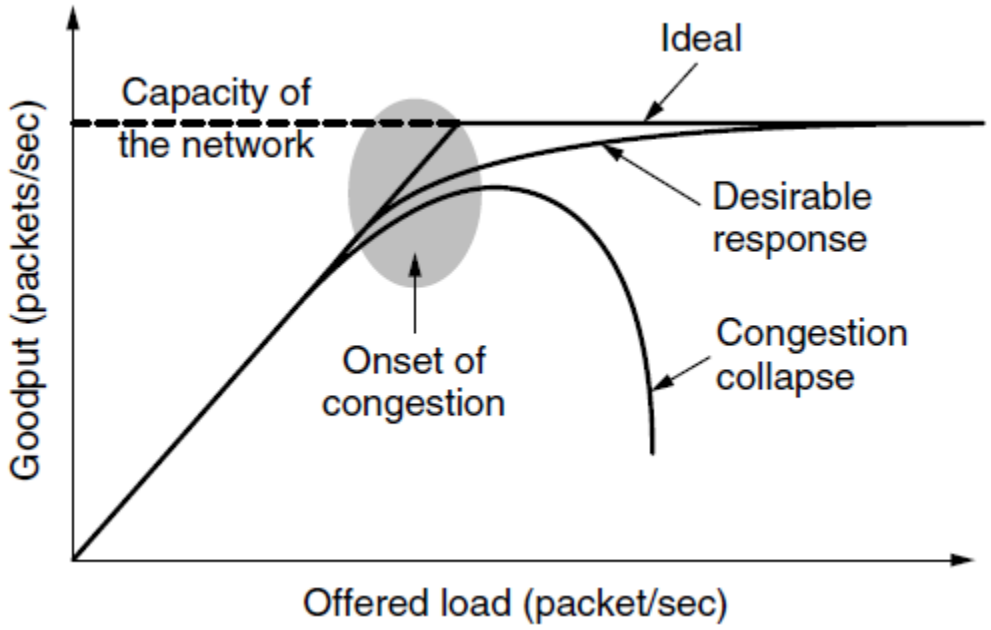
**Shilpa Ingoley**

Lecture 31

- 4.2 **Routing algorithms :** Shortest Path (Dijkastra‘s), Link state routing, Distance Vector Routing
- 4.3 **Protocols** - ARP,RARP, ICMP, IGMP
- 4.4 **Congestion control algorithms:** Open loop congestion control, Closed loop congestion control, QoS parameters, Token & Leaky bucket algorithms

# 4.4 Congestion control algorithms:

- In a network, the congestion occurs when the load on the respective network (i.e. the quantity of packets sent to the network) is comparatively more than its capacity (the total number of packets which the network is able to handle).




Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**.

**Fig. :Performance drops sharply, when more packet are sent in network than its capacity**

# Contd..

## Effects of Congestion

1. Packet  delivery delay
2. Undelivered packets
3. Packet loss
4. Retransmission

## Reasons or Factors can Cause Congestion

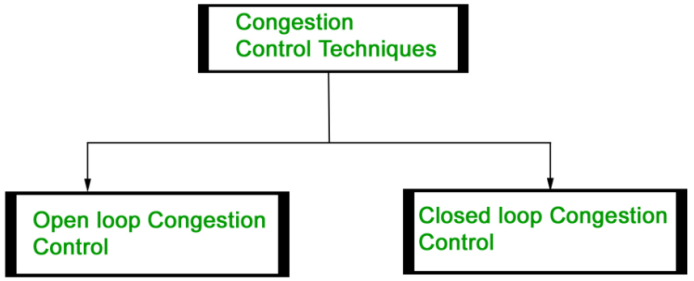
1. Packet arrival rate exceeds the outgoing link capacity of router/node.
2. Insufficient memory to store arriving packets.
3. Bursty (inconsistent) traffic.
4. Slow processor.

# Contd..

## Need of Congestion Control

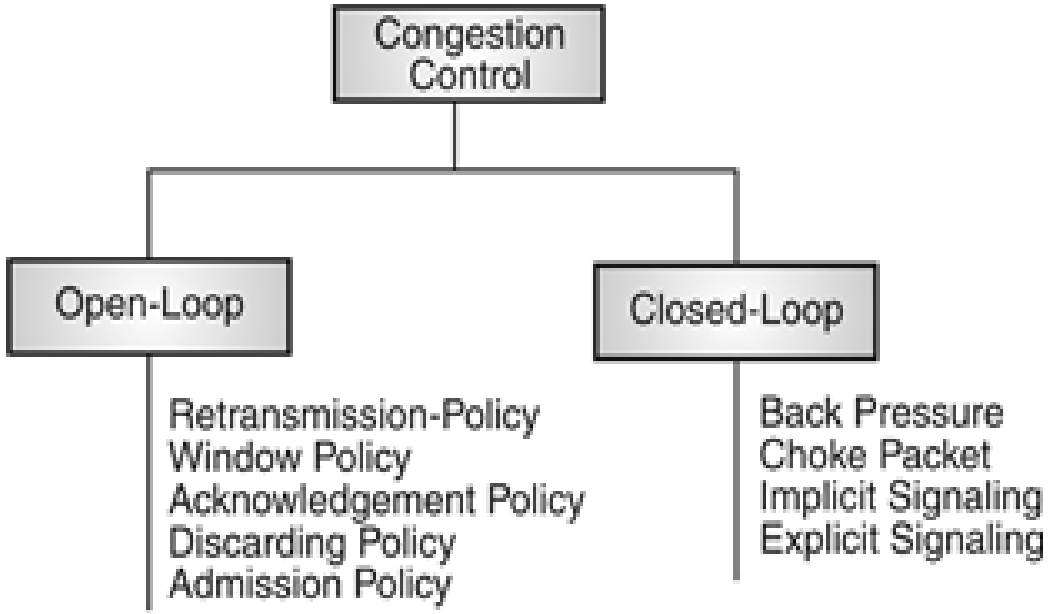
1. It is not possible to completely avoid the congestion but should be control.
2. Congestions built-up to a large Queue at routers.
3. Because of congestion Buffer Overflow & Loss of Packets happens.
4. Should receive negotiated Quality of Services.

# Congestion control mechanisms

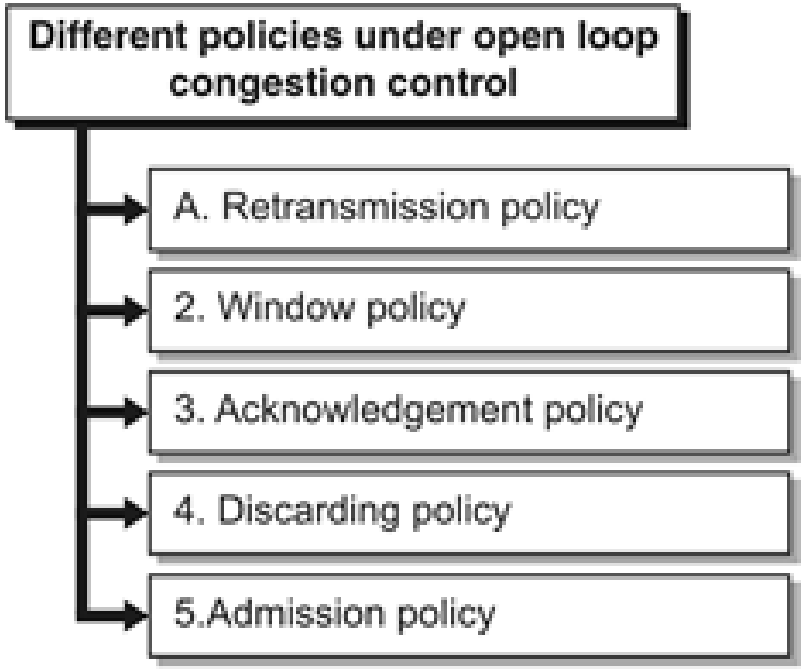



Congestion control mechanisms/technique are categorized into **two** parts :

1. **Open-loop congestion control (prevention)**
2. **Closed-loop congestion control (removal)**



# Open loop congestion control



# Contd..

## 1. Retransmission Policy

- Retransmission considered as a process which is sometimes unavoidable.
- . If the sender has doubt that a packet which has been sent is lost or may be corrupted, there is need to retransmit the packet.
- Usually retransmission may leads to increase in congestion of the network.
- However, a good retransmission policy has ability to avoid the congestion. The design of the retransmission policy and the retransmission timers should focus on optimization of efficiency and also prevent congestion at the same time.

## 2. Window Policy

- Selective Repeat window
- Go-Back-N window
- (For controlling the congestion, the Selective Repeat window is considered as better compared to the Go-Back-N window)

# Contd..

## 3. Acknowledgment Policy

- Sending less number of acknowledgments leads to impose fewer loads on the network
- If the receiver of packets does not give any acknowledgement regarding each packet which has been received by it, the sender may get slow down and helps to avoid congestion.
- A receiver has choice to acknowledge only when it has a packet to be sent or a special timer expires.
- A receiver may take decision to provide acknowledgment for only N packets at a time.

## 4. Discarding Policy

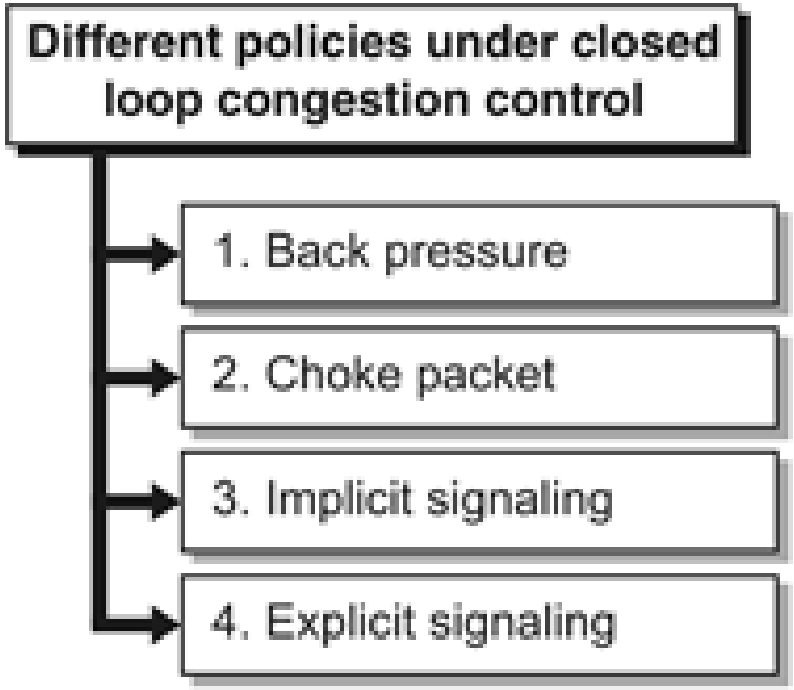
- A better discarding policy applied by the routers may helps to avoid congestion by keeping the integrity of the transmission.
- To abandon less sensitive packets when there is possibility of congestion

# Contd..

## 5. Admission Policy

- An admission policy that is considered as a quality-of-service technique is also able to prevent congestion in virtual-circuit networks.
- A router has option to refuse establishing a virtual- circuit connection if there is occurrence of congestion in the respective network or if future possibility of congestion is seen.

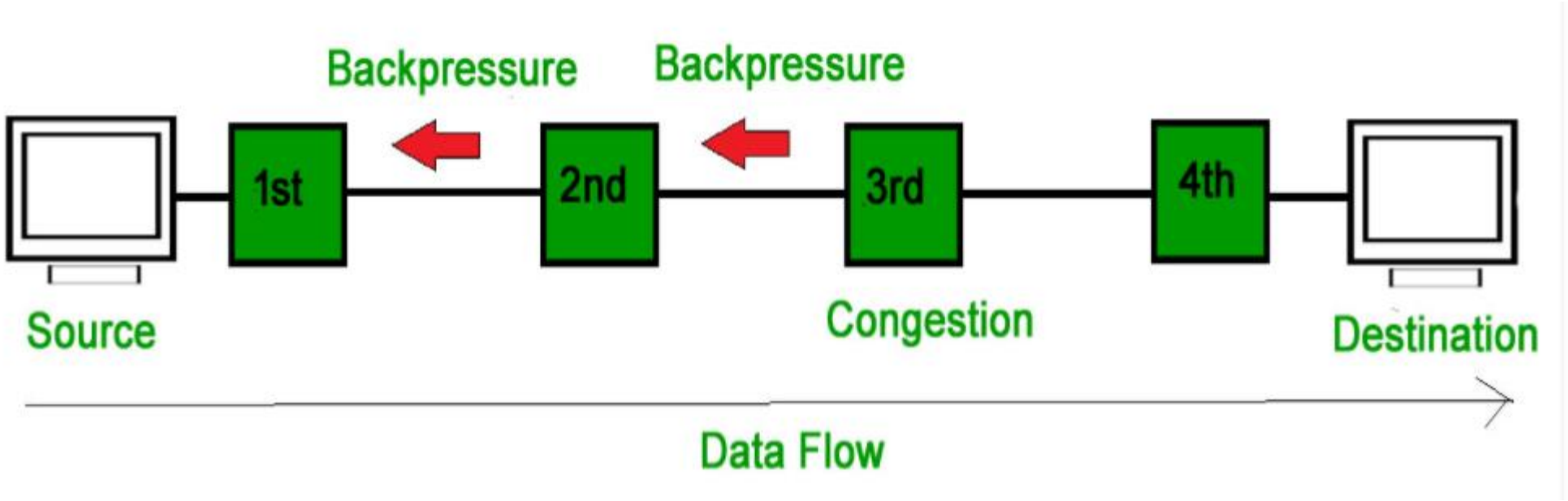
# Closed loop congestion control



# Contd..

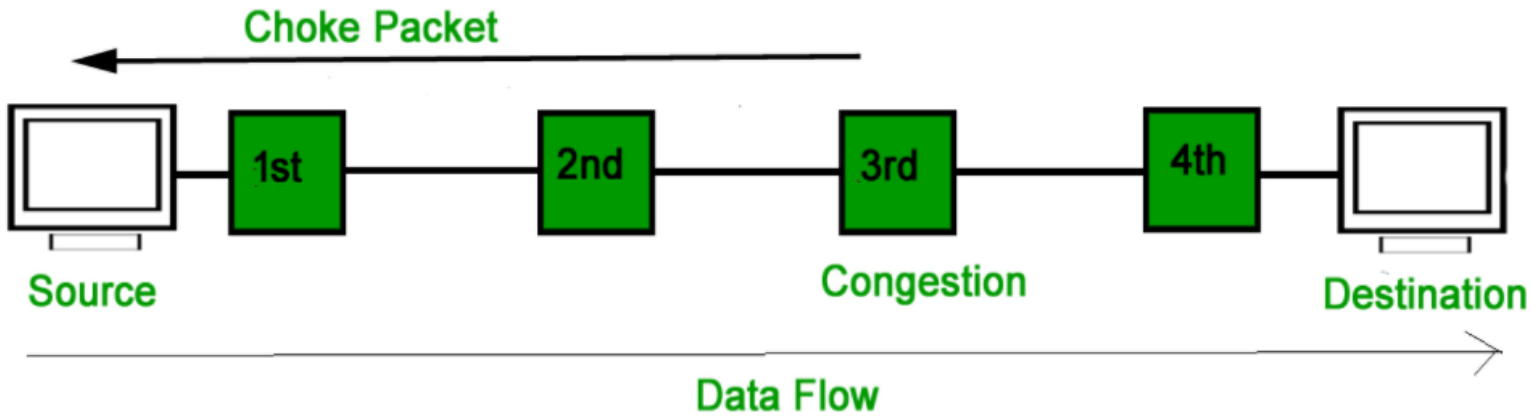
## 1. Back-pressure/ Warning bit

- A special bit in the packet header is set by the router to warn the source when congestion is detected.



# Contd..

- **2. Choke Packet**
- A choke packet is a packet which a node sends to the source for the purpose of informing it regarding the congestion.




- In back-pressure, one node gives warning just to its upstream node, even though the warning may ultimately reach the source station.
- In the choke packet technique, router which has found the congestion gives the warning to the source station directly. The instant nodes by which the packet has traveled do not receive any warning.

# Contd..

## 3. Implicit Signaling

- The source assumes possibility of congestion somewhere in the network with the help of other symptoms.
- For example, when several packets are sent by a source and it do not get any acknowledgment from the receiver for some time period then one consideration is that the network is congested; the speed of sending packets of source should slow down.

## 4. Explicit Signaling

- The node which encounters congestion has option to explicitly send a signal to the source or destination. The explicit signaling technique is considered as different from the choke packet technique.
- In the choke packet technique, an independent packet is referred for this task whereas in the explicit signaling technique, the signal is added in the packets which carry data. Explicit signaling can happen in either the forward or the backward direction.

# Comparison between Open and Closed loop

| Parameter | Open loop congestion control | Closed loop congestion control |
|---|---|---|
| Based on | Based on prevention of congestion | Based on solution for removing the congestion |
| Work performed | It prevents congestion from happening | It removes congestion after it took place |
| End-to-end feedback | It does not need End-to-end feedback | It adjusts cell rate depending upon some kind of feedback |
| Policies used | Retransmission Policy<br>Window Policy | Back Pressure<br>Choke Packet |

# Quality of Service (QoS)

- QoS is an important parameter for network performance
- It is specific to the application
- A stream of packets from a source to a destination is called a **flow.**
- In a connection-oriented network, all the packets belonging to a flow follow the same route
- In a connectionless network, they may follow different routes.
- The needs of each flow can be characterized by **four** primary parameters called as **QoS parameters**

# QoS Characteristics

PARAMETERS

- Reliability
- Delay
- Jitter
- Bandwidth

# Contd…

➢**Reliability:**

➢It is a characteristic that a flow needs in order to deliver the packets safe and sound to the destination. Lack of reliability means losing a packet or acknowledgment, which leads to retransmission.

- **Example:** Reliable transmission is more important for e-mails, file transfer, and Internet access than for telephony or audio conferencing.

➢**Delay:**

➢ Applications can tolerate delay in different degrees

- **Example:** Telephony, audio conferencing, video conferencing, and remote logging need **minimum delay**, while delay in file transfer or e-mail is less important.

# Contd..

- **Jitter:**
- Jitter is the variation in delay for packets belonging to the same flow
  - **Example:** For applications such as audio and video it does not matter if the packets arrive with a short or long delay as long as the delay is the same for all packets. These types of applications do not tolerate jitter.
- **Bandwidth:**
- Different applications need different bandwidths.
  - **Example:** In video conferencing we need to send millions of bits per second to refresh a color screen while the total number of bits in an e-mail may not reach even a million.

# Contd..

| Application | Bandwidth | Delay | Jitter | Loss |
|---|---|---|---|---|
| Email | Low | Low | Low | Medium |
| File sharing | High | Low | Low | Medium |
| Web access | Medium | Medium | Low | Medium |
| Remote login | Low | Medium | Medium | Medium |
| Audio on demand | Low | Low | High | Low |
| Video on demand | High | Low | High | Low |
| Telephony | Low | High | High | Low |
| Videoconferencing | High | High | High | Low |

# Techniques for achieving good QoS

- **Overprovisioning:**
- Provide extra router capacity, buffer space, and bandwidth.
  - Problem: It is expensive. To some extent, the telephone system is overprovisioned.
- **Buffering:**
- Flows can be buffered on the receiving side before being delivered.
  - Buffering them does not affect the reliability or bandwidth, and increases the delay, but it **smooths out the jitter**.
  - For audio and video on demand, jitter is the main problem, so this technique helps a lot.

# Contd..

- **Traffic Shaping:**
- It smooths out the traffic on the **server side**, rather than on the client side
  - Traffic shaping is about **regulating the average rate of data transmission**.
  - In contrast, the **sliding window protocols** limit the **amount of data in transit** at once, **not the rate at which it is sent**.
  - When a connection is set up, the user and the subnet (i.e., the customer and the carrier) **agree on a certain traffic pattern** (i.e., shape) for that circuit. This is called a **service level agreement**.
  - Such agreements are not so important for file transfers but are of great importance for **real-time data, such as audio and video connections**, which have stringent quality-of-service requirements.
  - Monitoring a traffic flow is called **traffic policing.**

# Contd…

## Traffic Shaping:

- Traffic shaping reduces congestion
- Two ways of achieving traffic shaping:
  - **Leaky Bucket algorithm**
  - **Token Bucket algorithm**

# Contd…

- **Resource Reservation**: Three different kinds of resources can potentially be reserved:
  - 1. Bandwidth.
  - 2. Buffer space
  - 3. CPU cycles.
- **Admission Control:** Once the incoming traffic from some flow is well shaped and can potentially follow a **single route** in which capacity can be reserved in advance on the routers along the path, the router, has to decide, based on its capacity and how many **commitments it has already made** for other flows, whether to **admit or reject the flow**.
- The decision to **accept or reject a flow is decided by** some flow specifications like **data rate, max. packet size, min. packet size**

# Contd…

- **Proportional Routing**: Split the traffic for each destination over multiple paths.
  - A simple method is to divide the traffic equally or in proportion to the capacity of the outgoing links.
- **Packet Scheduling:** Various packet scheduling algorithms have been devised. e.g.-**Fair queuing algorithm**
  - The essence of the algorithm is that routers have separate queues for each output line, one for each flow.
  - When a line becomes idle, the router scans the queues in a round robin fashion, taking the first packet on the next queue.

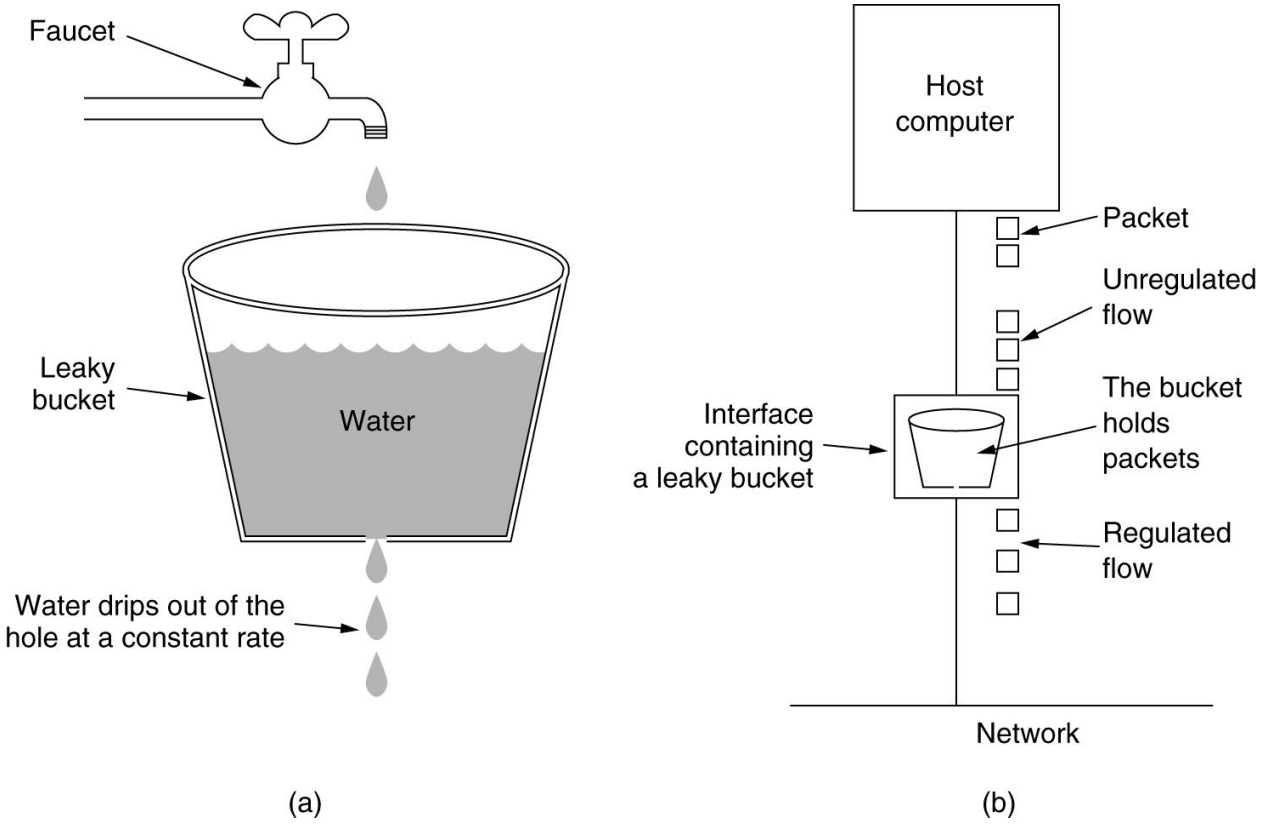
# Leaky Bucket Algorithm

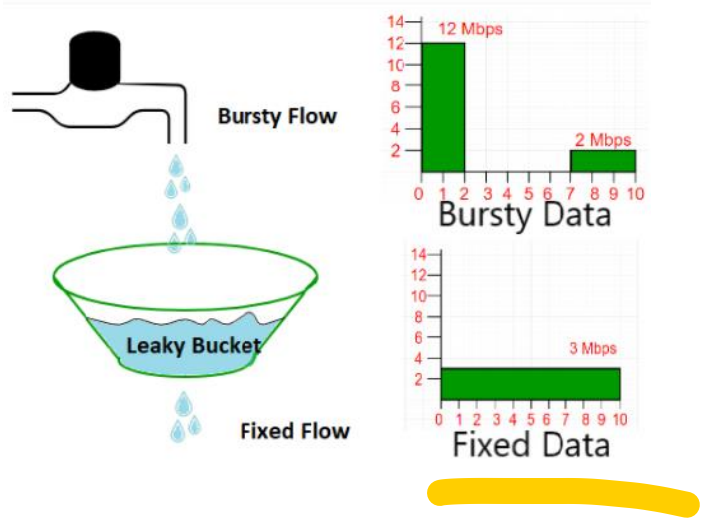
- It was proposed by **Turner (1986)** and is called the **leaky bucket algorithm.**
- No matter the rate at which water enters the bucket, the **outflow is at a constant** rate, ρ, when there is any water in the bucket and zero when the bucket is empty.
- Also, once the **bucket is full**, any additional water entering it **spills over** the sides and is **lost** (i.e., does not appear in the output stream under the hole).
- Each host is connected to the network by an interface containing a leaky bucket, that is, **a finite internal queue**.
- If a packet arrives at the queue when it is full, the packet is discarded.
- In other words, if one or more processes within the host try to send a packet when the **maximum number** is already queued, the **new packet is discarded**.

# Contd…

- This arrangement can be built into the hardware interface or simulated by the host operating system. It is nothing other than a **single-server queueing system** with constant service time
- The host is allowed to put **one packet per clock tick** onto the network.
- This mechanism turns an **uneven flow** of packets from the user processes inside the host into an **even flow of packets** onto the network, smoothing out bursts and greatly reducing the chances of congestion.
- This method is applicable when the packets are all the same size

# Congestion Avoidance policies The Leaky Bucket Algorithm




(a) A leaky bucket with water. (b) a leaky bucket with packets.

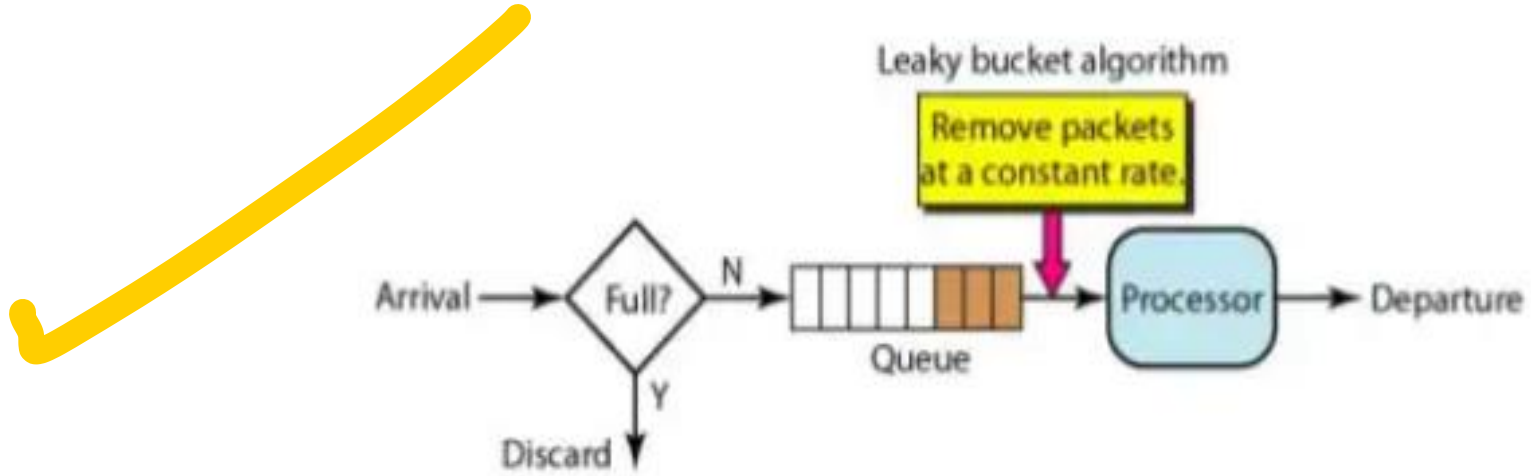
# Leaky Bucket Algorithm-Example

- A computer can produce data at 25-million-byte (200 Mbps)
- Network Speed: 200 Mbps, Router speed: 2Mbps
- Now suppose data comes in 1-million-byte bursts, one 40-msec burst every second. To reduce the average rate to 2 MB/sec, we could use a leaky bucket with $\rho$=2 MB/sec and a capacity, C, of 1 MB. This means that bursts of up to 1 MB can be handled without data loss and that such bursts are spread out over 500 msec, no matter how fast they come in.

# Contd…

- For a **variable size packet**, a **fixed number of bytes per tick** is allowed rather than just one packet.
- At each tick, a **counter** is initialized to **n**.
- If the first packet on the queue has fewer bytes than the current value of the counter, it is transmitted, and the counter is decremented by that number of bytes.
- Additional packets may also be sent, as long as the counter is high enough.
- When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.
- **Example:** If the rule is **1024** bytes per tick, a **single 1024-byte** packet can be admitted on a tick, **two 512-byte** packets, **four 256-byte packets**, and so on. If the residual byte count is too low, the next packet must wait until the next tick.

# Contd...




- 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 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

# Contd..

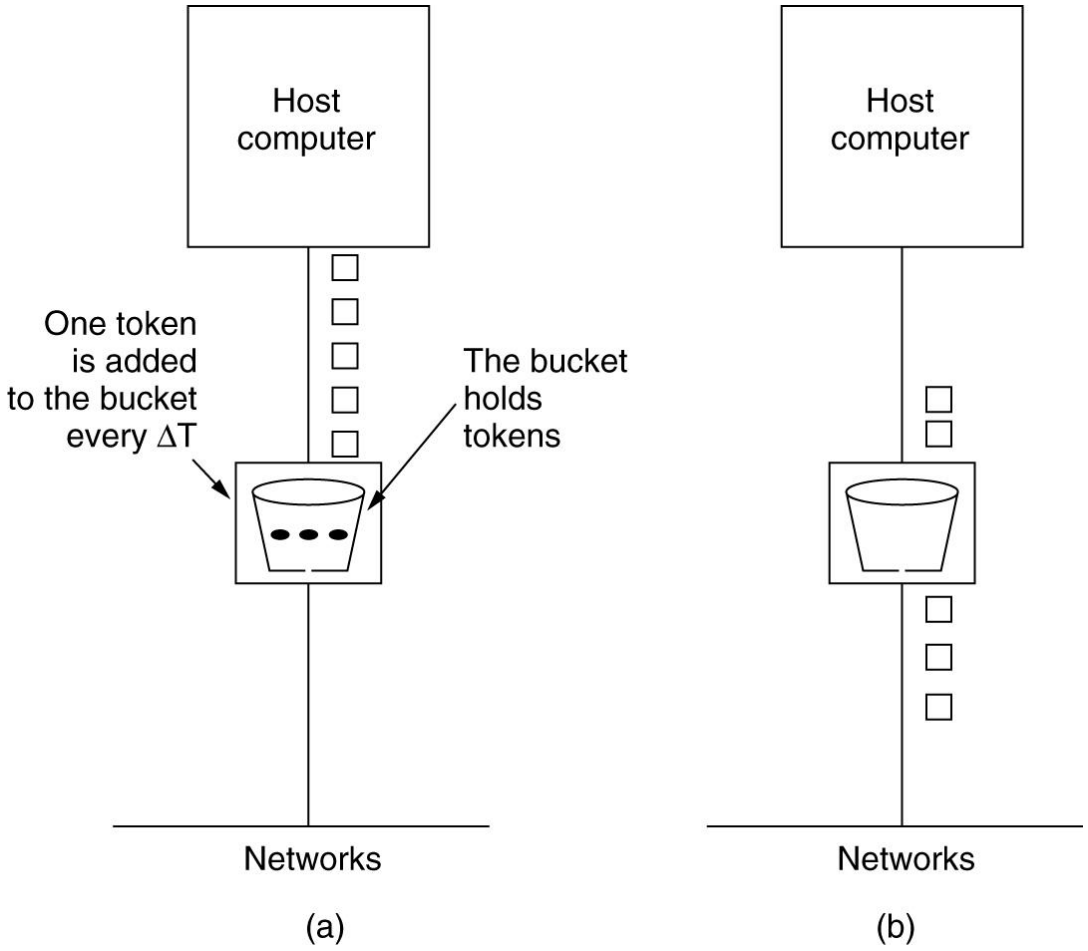
- **Dis/Advantage of Leaky Bucket Algorithm:** It enforces a rigid output pattern at the average rate, no matter how bursty the traffic is.

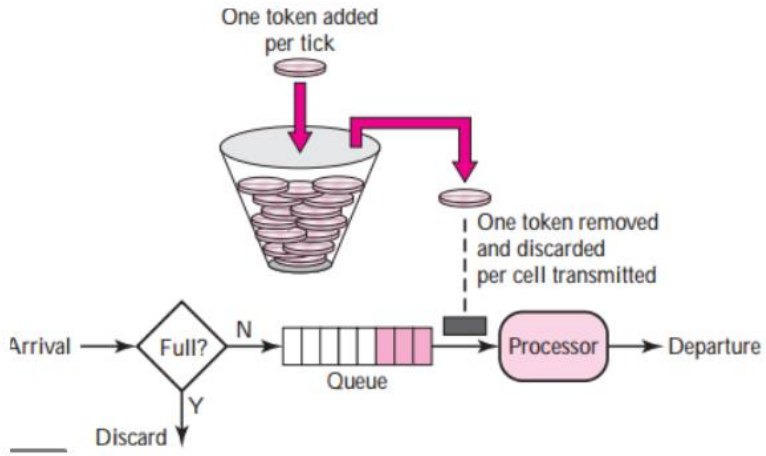
- **Advantage of Token Bucket Algorithm:** It is more flexible.
- For many applications, it is better to allow the output to **speed up somewhat when large bursts arrive**, so a more flexible algorithm is needed, preferably one that never loses data.
- In this algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every $\Delta T$ sec.

# Token Bucket Algorithm

- The token bucket algorithm provides a **different kind of traffic shaping** than that of the leaky bucket algorithm.
- The leaky bucket algorithm does not allow **idle hosts to save up permission to send large bursts later.**
- **Bursts of up to n packets can be sent at once**, allowing some burstiness in the output stream and giving **faster response** to sudden bursts of input.
- The token bucket algorithm **throws away tokens** (i.e., transmission capacity) when the bucket **fills up but never discards packets**. In contrast, the leaky bucket algorithm discards packets when the bucket fills up.

# The Token Bucket Algorithm




(a) Before. (b) After.

# Contd..

- **Disadvantage of token bucket**: It allows large bursts again, even though the maximum burst interval can be regulated by careful selection of ρ and M.
- **Solution**: To get smoother traffic, insert a **leaky bucket after the token bucket**.
- The rate of the **leaky bucket should be higher than the token bucket's ρ** but lower than the maximum rate of the network.

# Comparison of Token Bucket and Leaky Bucket

| Parameter | Leaky Bucket | Token Bucket |
|---|---|---|
| Token dependency | Not dependent | Dependent |
| When bucket is full | Packet or data is discarded | Tokens are discarded but not packet |
| Packet transmission | Continues | Packet transmission is possible only when there are sufficient tokens |
| Tokens | Does not save tokens | Save token to send large bursts |
| Transfer rate | Faster after constant | Constant |