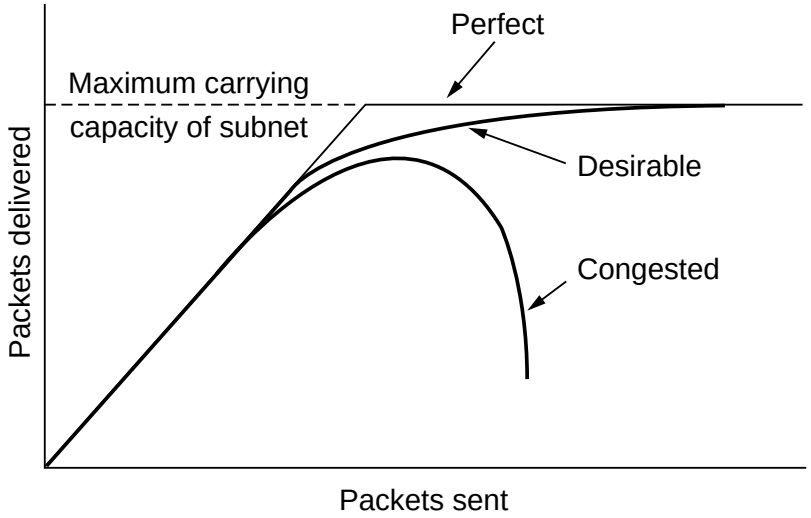
# Congestion Control Overview

- **Problem**: 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 $\longrightarrow$ When part of the network no longer can cope a sudden increase of traffic, congestion builds upon. Other factors, such as lack of bandwidth, ill-configuration and slow routers can also bring up congestion




- **Solution**: congestion control, and two basic approaches
  - Open-loop: try to prevent congestion occurring by good design
  - Closed-loop: monitor the system to detect congestion, pass this information to where action can be taken, and adjust system operation to correct the problem (detect, feedback and correct)

- Differences between congestion control and flow control:
  - **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 feedback
  - **Flow control** is related to point-to-point traffic between given sender and 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 summarised in the table

  e.g. retransmission policy at 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

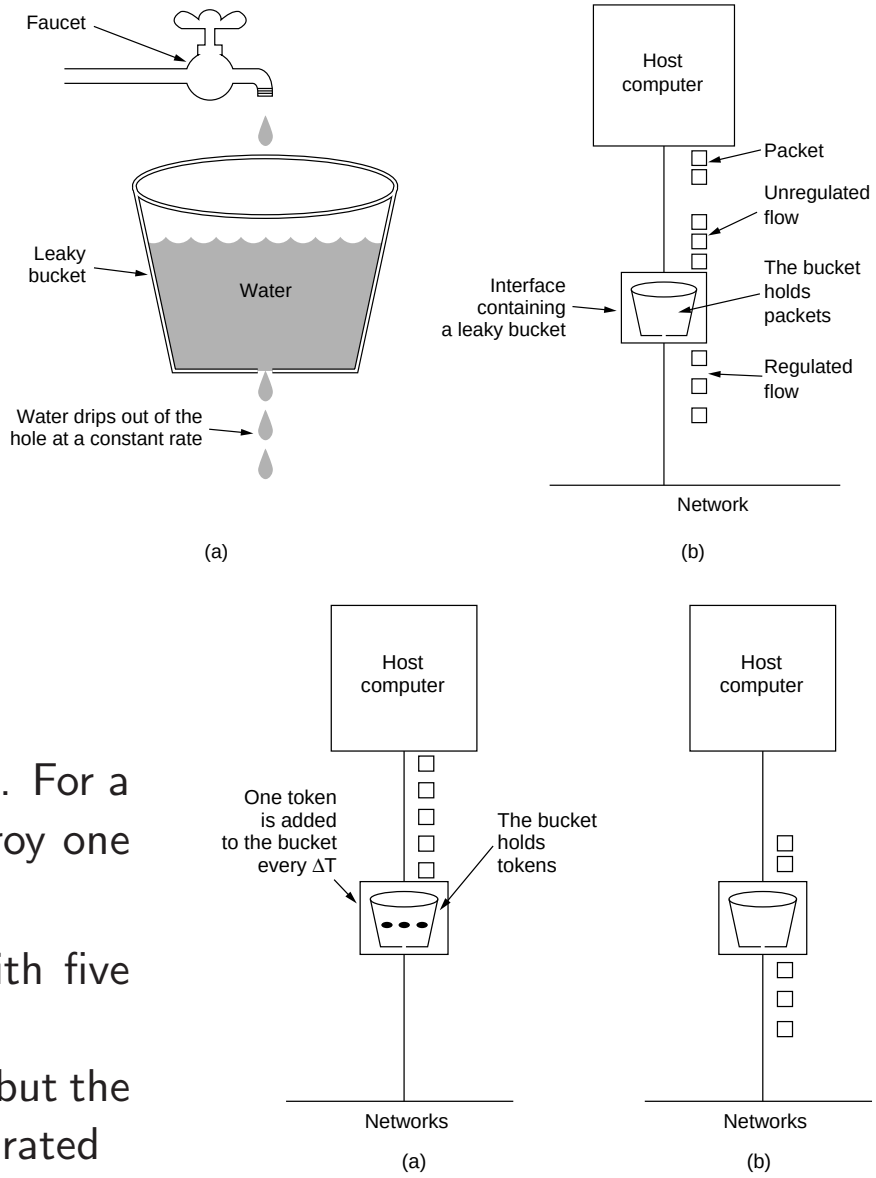
| | |
|---|---|
| Transport | • Retransmission policy<br>• Out-of-order caching policy<br>• Acknowledgement policy<br>• Flow control policy<br>• Timeout determination |
| Network | • Virtual circuit versus datagram<br>• Packet queueing and service policy<br>• Packet discard policy<br>• Routing algorithm<br>• Packet lifetime management |
| Data link | • Retransmission policy<br>• Out-of-order caching policy<br>• Acknowledgement policy<br>• Flow control policy |

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

- **Traffic shaping**: As burstiness of traffic is a main cause of congestion, it is used to regulate average rate and burstiness of traffic
  - e.g. when a virtual circuit is set up, the user and the subnet first agree certain traffic shape for that circuit. Monitoring traffic flow, called traffic policing, is left to the subset
  - 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 / Token Bucket

- **Leaky bucket**: consists of a finite queue
  - When a packet arrives, if there is a room on the queue it is 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: packets passed to the subnet at the same rate
- This may be a bit overdone, and also packets can get lost (when bucket is full)
- **Token bucket**: Tokens are added at a constant rate. For a packet to be transmitted, it must capture and destroy one token
  - (a) shows that the bucket holds three tokens with five packets waiting to be transmitted
  - (b) shows that three packets have gotten through but the other two are stuck waiting for tokens to be generated



# Token Bucket (continue)

- 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 packetswhile leaky bucket discards packets when the bucket is full
- Let token bucket capacity be $C$ (bits), token arrival rate $\rho$ (bps), maximum output rate $M$ (bps), and burst length $S$ (s)
  - During burst length of $S$ (s), tokens generated are $\rho S$ (bits), and output burst contains a maximum of $C + \rho S$ (bits)
  - Also output in a maximum burst of length $S$ (s) is $M \cdot S$ (bits), thus

$$C + \rho S = MS \quad \text{or} \quad S = \frac{C}{M - \rho}$$

- Token bucket still allows large bursts, even though the maximum burst length $S$ can be regulated by careful selection of $\rho$ 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

# Illustration

(a) The input to a leaky bucket, 25 Mbps for 40 ms, i.e. $40 \times 25$ Kbits $= 1$ Mbits burst

(b) The output of leaky bucket at 2 Mbps uniform rate for 500 ms (assuming no packet is discarded)

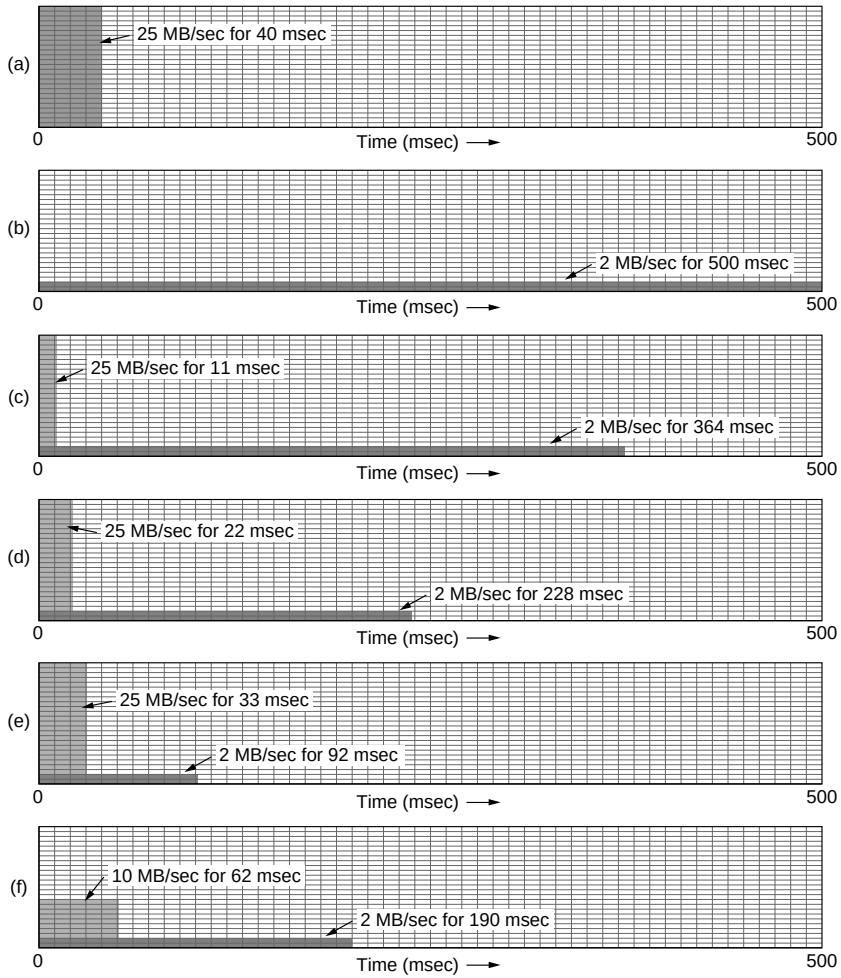
(c) The output of token bucket with 250 Kbits capacity and token arrival rate 2 Mbps, assuming that the token bucket is full when the 1 Mbits burst arrives

Output at first is at the full burst rate of 25 Mbps for about 11 ms ($S = 10.87$ ms). During this burst period, 272 Kbits are send. The remaining $1000 - 272 = 728$ Kbits have to be cleared at the rate 2 Mbps for a duration of $728/2 = 364$ ms

(d) and (e) The outputs of token buckets for capacities of 500 Kbits and 750 Kbits, respectively, with token arrival rate 2 Mbps

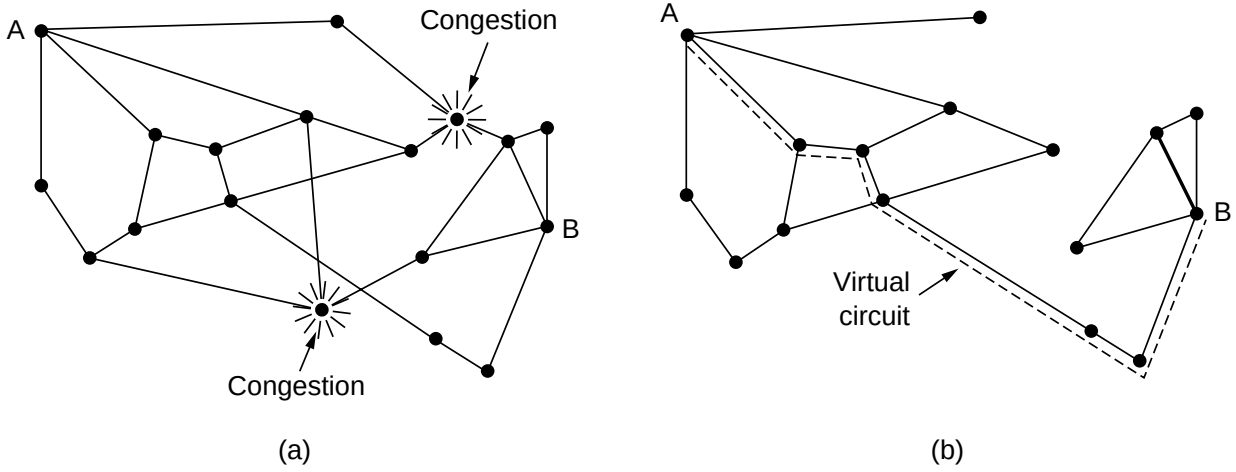
(f) The output for 500 Kbits token bucket followed by a 10 Mbps leaky bucket

$21.7 \times 25 + (x - 21.7) \times 2 = 10x \;\rightarrow\; x = 62.4$ ms



# Congestion Control in Virtual Circuits

- These are closed-loop based designed for virtual circuits subnets, which are connection oriented → during connection set up, something can be done to help congestion control
- The basic principle is obvious: When setting up a virtual circuit, make sure that congestion can be avoided
  - **Admission control**: Once congestion has been signaled, no more new virtual circuits can be set up until the problem has gone away. This is crude but simple and easy to do




  - **Select alternative routes** to avoid part of the network that is overloaded, i.e. temporarily rebuild your view of network

    e.g. Normally, when router $A$ sets a connection to $B$, it would pass through one of the two congested routers, as this would result in a minimum-hop route (4 and 5 hops respectively). To avoid congestion, a temporary subnet is redrawn by eliminating congested routers. A virtual circuit can then be established to avoid congestion

  - **Negotiate quality of connection in advance**, so that network provider can reserve buffers and other resources, guaranteed to be there

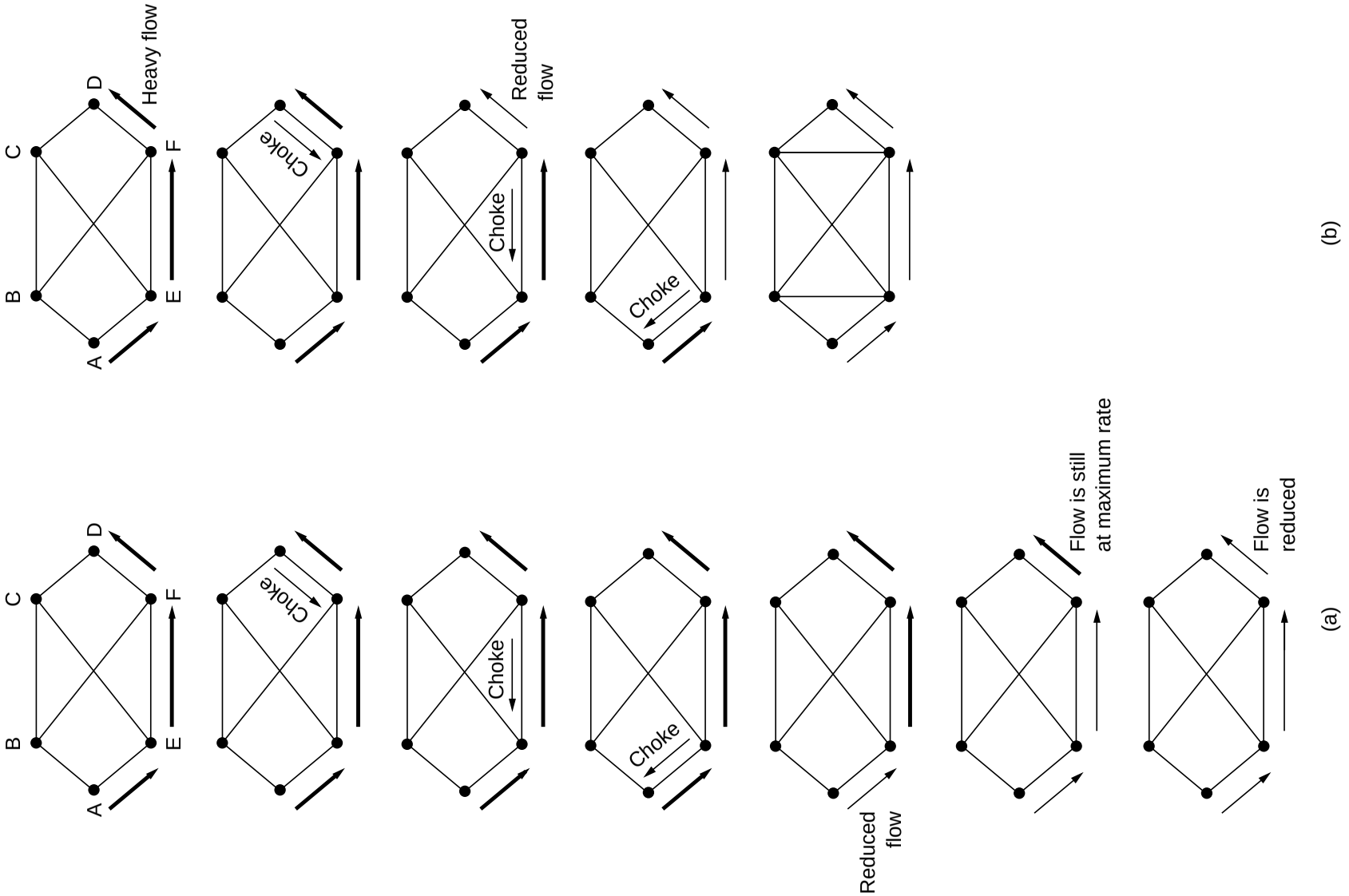
# Choke Packets

- This closed-loop congestion control is applicable to both virtual circuits and datagram subnets
- **Basic idea**: Router checks the status of each output line: if it is too occupied, sends a **choke packet** to the source. The host is assumed to be cooperative and will slow down
  - When the source gets a chock packet, it cuts rate by half, and ignores further choke packets coming from the same destination for a fixed period
  - After that period has expired, the host listens for more choke packets. If one arrives, the host cut rate by half again. If no choke packet arrives, the host may increase rate
- Uncooperative cheating host may get all bandwidth while cooperative honest host gets penalised $\longrightarrow$ Use weighted fair queueing to enforce cooperation and assign priority
- **Problem of basic Choke Packets**: For high-speed WANs, return path for a choke packet may be so long that too many packets have already been sent by the source before the source notes congestion and takes action

  Host in San Francisco (router $A$) is sending heavy traffic to a host in New York (router $D$), and $D$ is in trouble. It sends a choke packet to $A$. Note how long it takes for $A$ to reduce the rate and eventually to relieve $D$
- **Solution**: Use "push-back" or **hop-by-hop** choke packets

  When choke packet reaches router $F$, it forwards choke packet to router $E$ as well as reduces its traffic to $D$. Thus the problem that $D$ has is "push-back" to $F$ and $D$ gets relief quickly. This process is repeated down the route until the "ball" is back to the "root" source $A$

# Choke packets in WANs: (a) basic, (b) hope-by-hope



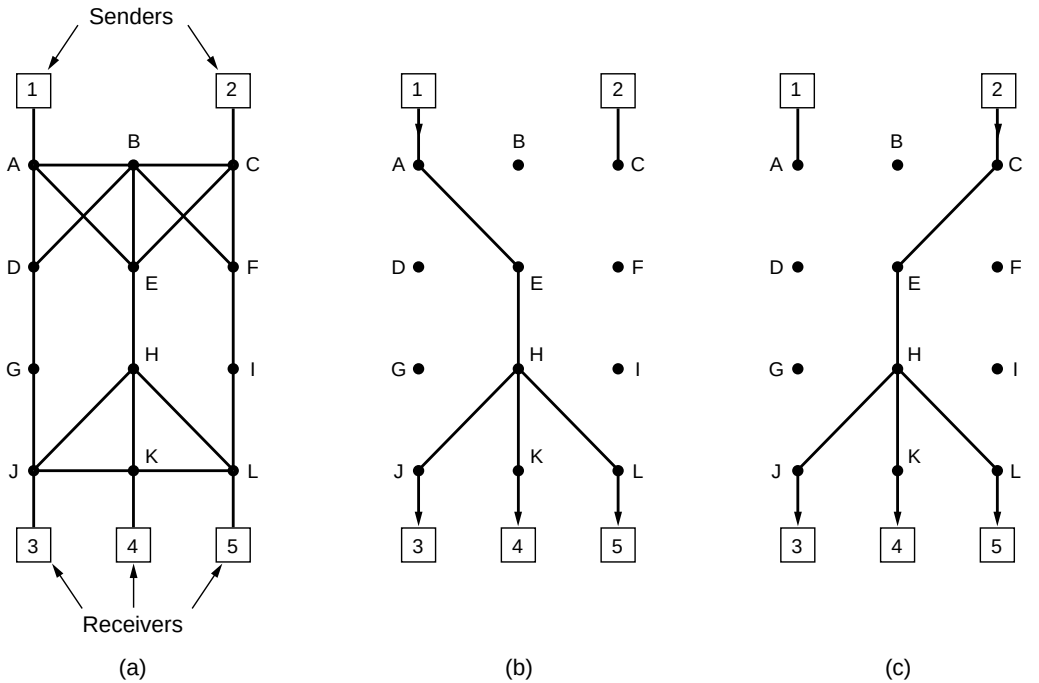
# Last Resort

- **Load Shedding**: When all the other methods cannot make congestion disappear, routers may force to use this last resort, but how to drop or discard packets?
- A router may randomly pick packets to drop but it can usually do better than this
- Which packet to discard depends on the applications running
  - **Wine policy**: For file transfer, an old packet is worth more than a new one. This is because dropping an old packet may force more packets to be retransmitted (since receiver will discard out-of-order packets). For this kind of applications, "the older the better"
  - **Milk policy**: For multimedia, a new packet is more important than an old one. Thus, "fresher is better"
- Implementing some sort of **intelligent discard policy**: For some applications, some packets are more important than others

  e.g. in MPEG video standard, periodically, an entire frame is transmitted and this is followed by subsequent frames as differences from the full reference frame $\longrightarrow$ drop packets that is part of a difference is preferred to drop one that contains part of the last full reference frame
  - Applications mark their packets in priority classes to indicate how important they are, such as very important – never discard, or lower priority, etc.

# Congestion Control for Multicasting

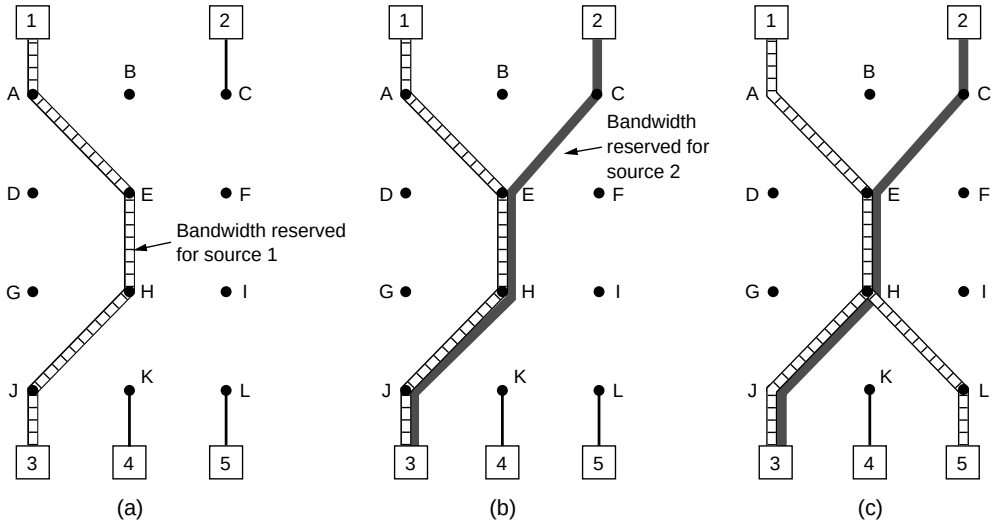
- Congestion control algorithms discussed so far deal with single-source to single-destination case
- In the advent of all kinds of services on the Internet that deal with broadcasting streams of data (voice and video) with a limited bandwidth, managing multicast flows from multiple sources to multiple destinations becomes critical
- Multicast routing uses spanning trees
  - Hosts 1 and 2 are multicast senders, and hosts 3, 4 and 5 are multicast receivers
  - (a) shows network topology, multicast trees from hosts 1 and 2 are shown in (b) and (c)




- **Resource reservation protocol**: The basic idea is that, to avoid congestion, extra information can be broadcasted to the group periodically to tell the routers along the tree to maintain certain data structures in their memories

# RSVP (continue)

- Any receiver can send a reservation message up the tree to the sender, using the reverse path forwarding routing algorithm
  - At each hop, router notes reservation and reserve necessary bandwidth
  - If insufficient bandwidth is available, it reports back failure
  - By the time the message gets back to source, bandwidth has been reserved all the way from sender to receiver along the spanning tree




- Host 3 has requested a channel to host 1. Once it has been established, packets can flow from 1 to 3 without congestion. Next host 3 decides to reserve a channel to the other sender, host 2, and 2nd path is reserved
- Now, host 5 makes a reservation to host 1. First, dedicated bandwidth has to be reserved as far as router $H$. Router $H$ can see that it already has a feed from host 1
- Assume bandwidth requested for host 1 to host 5 is no more than that reserved for host 1 to host 3. As the necessary bandwidth has already been reserved, it does not have to reserve any more

# Summary

- Congestion: the root cause is the bursty nature of traffic
- Open-loop based congestion control

  Prevention and traffic shaping, leaky bucket, token bucket
- Closed-loop based congestion control

  For single source to single destination: congestion control in virtual circuits subnets and (hop-by-hop) choke packets approach

  For multiple sources to multiple destinations: resource reservation protocol