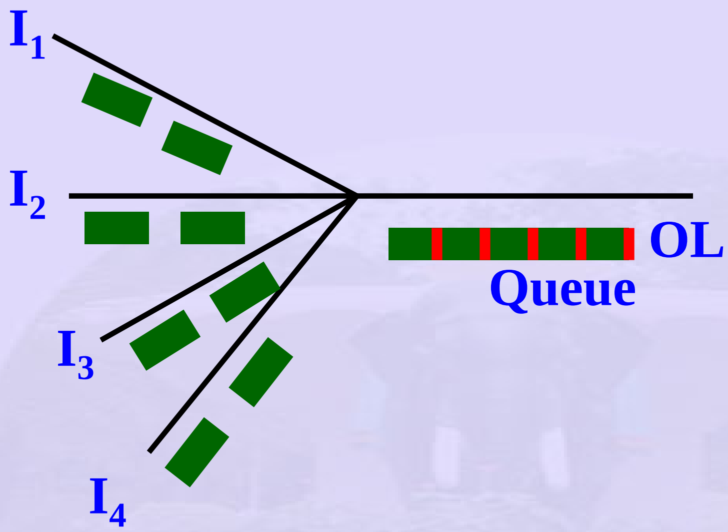
Congestion vs. Flow Control

- Flow control:
  - End-to-end
- Congestion control
  - Router to Router

Congestion control vs. Flow control:



- Congestion -
  - buffer length
    - Drop packets
  - Slow processor at the router even though line capacity is high
  - Mismatch between different parts of the system

Congestion vs. Flow Control

- **Router discards packets when it cannot serve**
  - **Sender retransmits until acknowledged**
    - **Congestion builds up**

- **Flow Control**
  - Pt – Pt links between a given sender and a given receiver
  - Fast sender does not overwhelm receiver
  - Receiver can tell sender directly to slow down

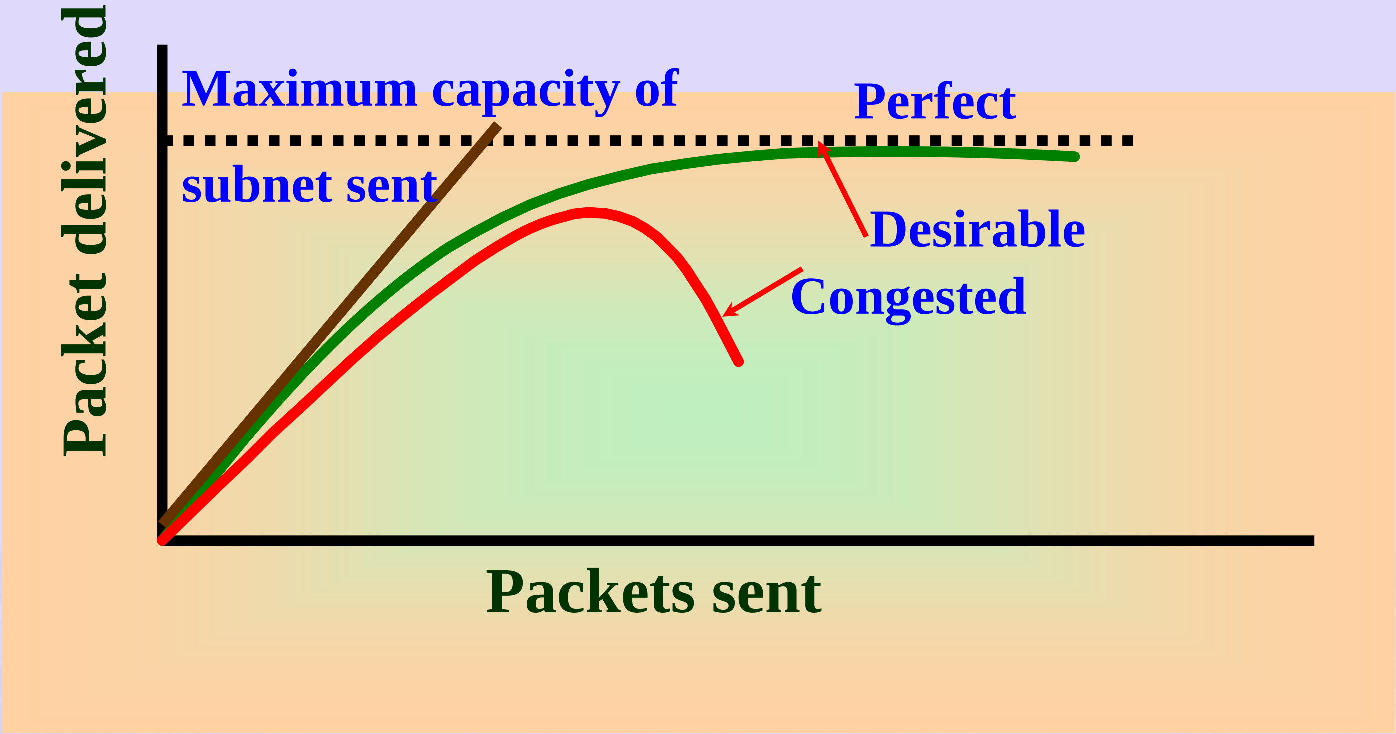
Congestion Control

- General principle of congestion:
  - Monitor system to detect when and where congestion occurs
  - Pass this information to places where action can be taken
  - Adjust system operation to correct the problem

Congestion vs. Flow Control

- Policing traffic at routers
  - Token bucket / leaky bucket
  - non trivial
- Alternative flow specifications:
  - Agreed between sender and receiver
  - pattern of injected traffic
  - QoS desired by Application

Congestion Control Algorithm:



- Routers loose packets
- Buffering?
  - No use
  - Packet reaches front of Queue, duplicate generated

Traffic Shaping

- Traffic monitoring:
  - Monitoring a traffic flow
  - VC no problem
    - Can be done for each VC separately since connection oriented
- DG - Transport layer

Congestion: Reasons

Congestion causing policies:

- Transport Layer
  - Retransmission
  - Out of other caching policy
  - Ack policy
  - Flow control policy
  - Time out
- Network Layer:
  - VC versus datagram inside subnet
  - Packet queuing and service policy
  - Packet discard policy
  - Routing algorithm policy
  - Packet lifetime management policy

Congestion Control (contd.)

- Solution:
  - Traffic prediction?
  - Router informs neighbour of possible congestion
  - Traffic shaping
  - Regulate the packet rate
  - VC - traffic characteristics
    - Not too important for file transfer but important for audio and video

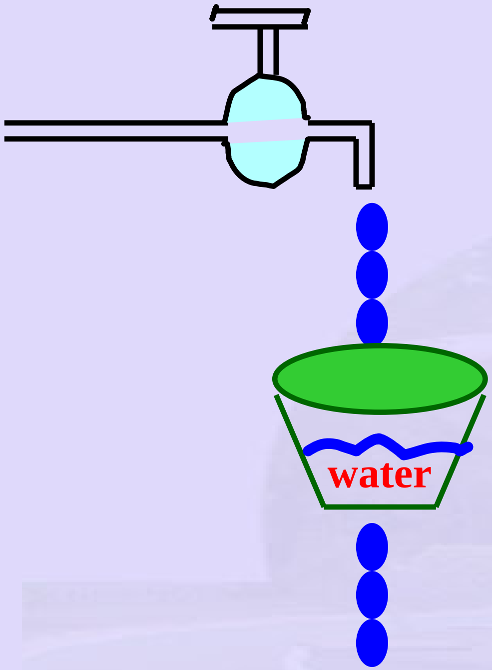
Congestion Control (contd.)

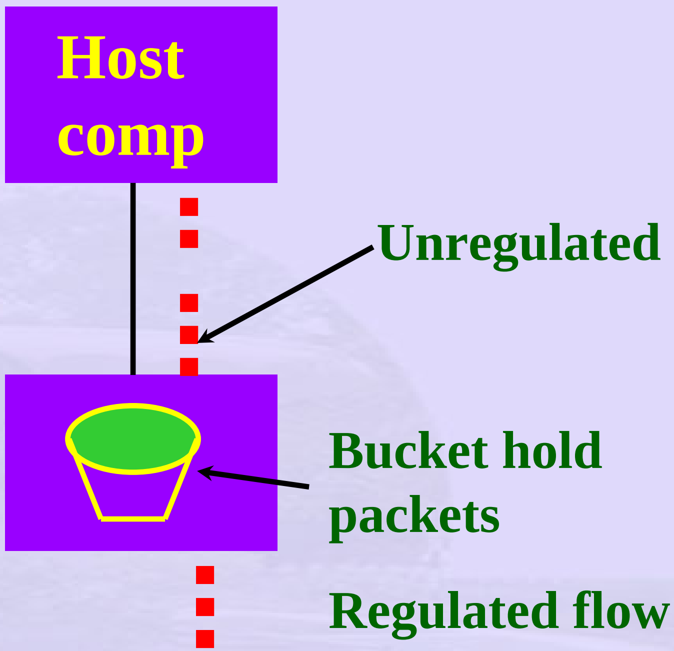
- Send probe packets periodically ask about congestion
  - Road congestion – use helicopters flying over cities
  - Bang bang operation of router – how does one prevent it
  - Feed back and control required

Congestion Control Algorithms

- Leaky Bucket Algorithm
  - Regulate output flow
    - Packets lost if buffer is full
- Token Bucket Algorithm
  - Buffer filled with tokens
    - transmit ONLY if tokens available

Leaky bucket algorithm:



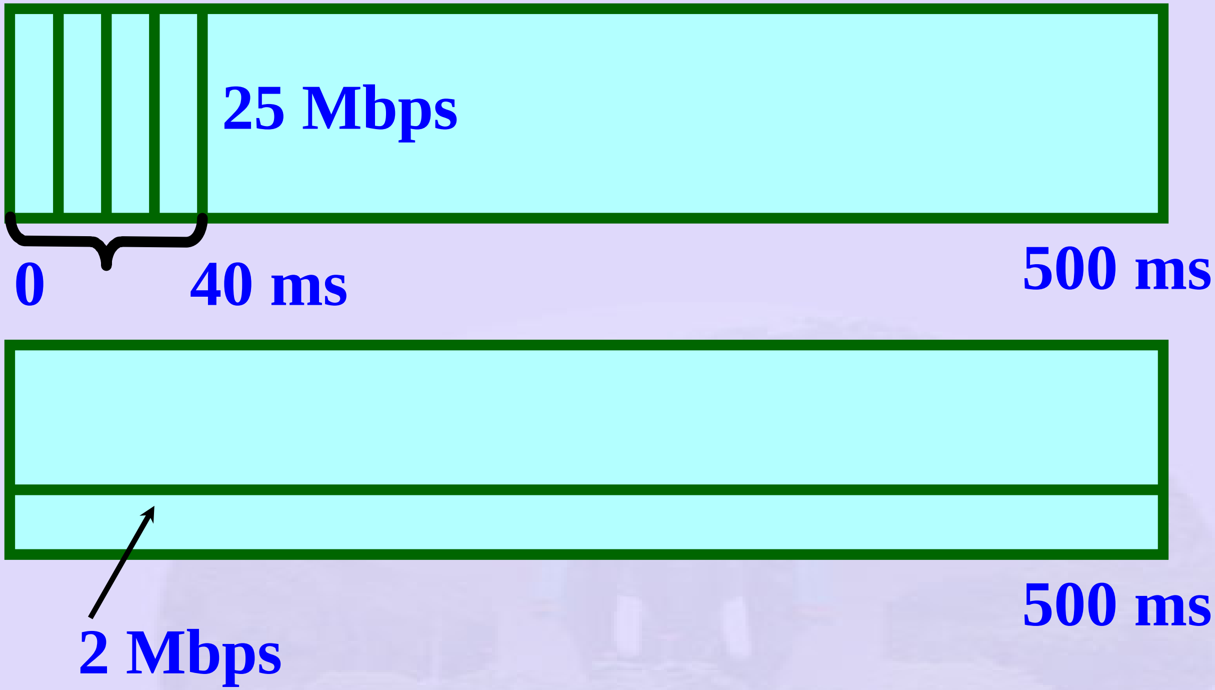


Bucket full – lost packets

- Output flow constant
  - when water in bucket – zero when no water
- Converts uneven flow to even flow
  - Packets Queued
  - Packets output at regular intervals only

Leaky Bucket Algorithm

- Queue full, packet discarded.
  - What if packets are different size and fixed bytes/ unit time.
- Leaky bucket example
  - Input burst 25 Mb/s every 40 ms
    - Network speed 25 Mbps – every second
    - Capacity of bucket C – 1 Mb
    - Reduce average rate – 2 Mbps
    - bucket can hold upto 1 Mb without data loss,
    - burst spread over 500 ms irrespective of how fast they come

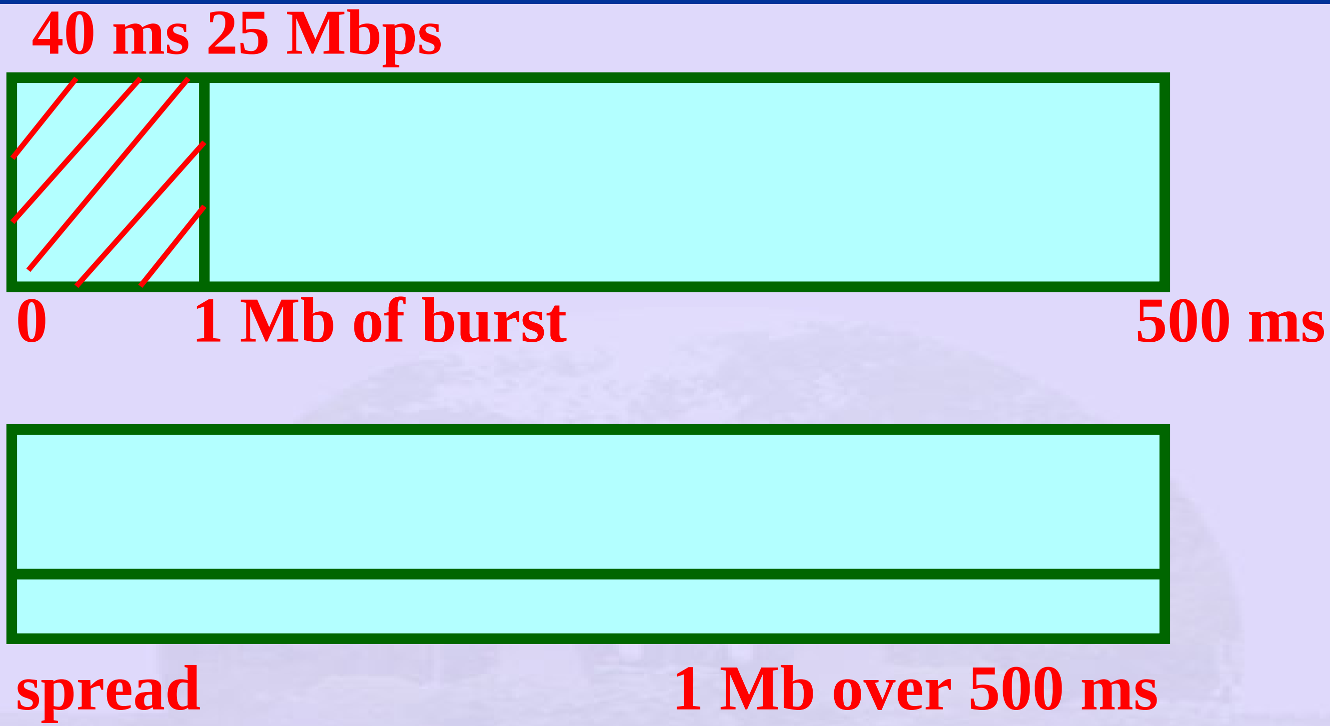


$\frac{25 \times 40}{1000}$ = 1 Mb

25 - 1000 $\qquad \frac{40 \times 25}{1000}$

? - 40 = 1 Mb every secs

- spread it over 500 ms

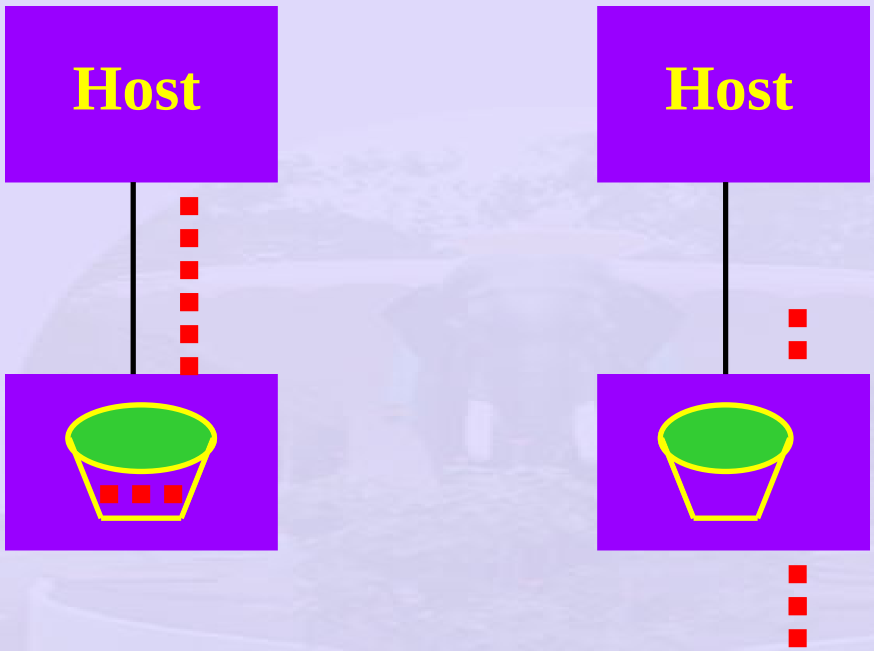


→ output rate 2 Mbps

Leaky bucket issues:

- * Drops packets
- * Does not allow host to save permission to transmit large burst later

Token bucket Algorithm



- Host save packets upto maximum size of bucket, n
- n packets send at once – some burstiness
- Host captures token
- Never loose data
- Tokens not available packets queue up! – not discarded

Δ

Token Bucket Algorithm

- Packet gets tokens and only then transmitted
  - A variant – packets sent only if enough token available - token - fixed byte size
  - Token bucket holds up n tokens
    - Host captures tokens
    - Each token can hold some bytes
    - Token generated every T seconds
    - Allows bursts of packets to be sent - max n
    - Responds fast to sudden bursts
    - If bucket full – thrown token packets not lost

Token Bucket Algorithm (example)

Calculation of length of maximum rate burst:

- Tokens arrive while burst output

Example

S – burst length in S

M – Maximum output rate

MS – Maximum byte in lengths

ρ **– Token arrival rate**

C – Capacity of token bucket in byte

Token Bucket Algorithm (Example)

Maximum output burst = C + ρS = MS

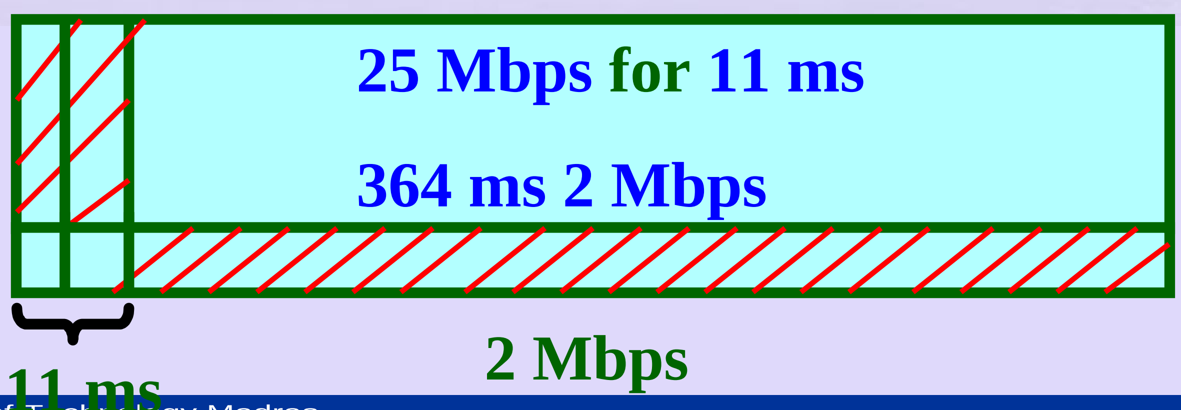
$$S = \left(\frac{C}{M-\rho}\right)$$

C = 250 Kb

M = 25 Mbps

ρ = 2 Mbps

S = 11 ml



Example specifications

Application to subnet	by Application
IP character	Services desired
Max packet size (bytes)	Loss sensitivity (bytes) / unit time
IP character	Loss interval time (bytes)
Token bucket Rate (r bytes/s)	Burst sensitivity
Token bucket size (b bytes	Min delay noticed
Max transmission rate	Max delay variation
	Quality of guarantee

Maximum rate possible

Shortest time in which token bucket empties

Does application mean it?

How many packets in seq lost

Maximum delay for a packets

Congestion Control in VCs

- Congestion control in VCs
  - Admission control
  - Allow VCs to avoid problem areas – avoid routers that are known to congest.
  - Negotiate agreement between host and subnet
    - Volume of traffic

Congestion Control in VCs

- Flow specification: Response from subnet to application
  - Issues – Sometimes application may not know what it wants
- iitm ⟷ imsc - fast
- iitm ⟷ thajavan – slow

Congestion Control in VCs

- Shape of traffic
- QoS required
  - Subnet – reserves resources along the entire path when VC is setup
- Issues:
  - 3 Mbps link
  - 4 VCs each requiring 0.75 Mbps
  - Wastes bandwidth
  - Unlikely that all VCs are simultaneously used

Congestion Control in VCs

- **Monitor utilisation on output lines**
  - **Associate a value for each line – recent utilisation update**
- $u_{new} = a\, u_{old} + (1-a)\, f$
- **instantaneous line utilisation**
  - **a – memory parameter**
  - **u > threshold implies output lines congests**
-

Congestion Control (Other mechanisms)

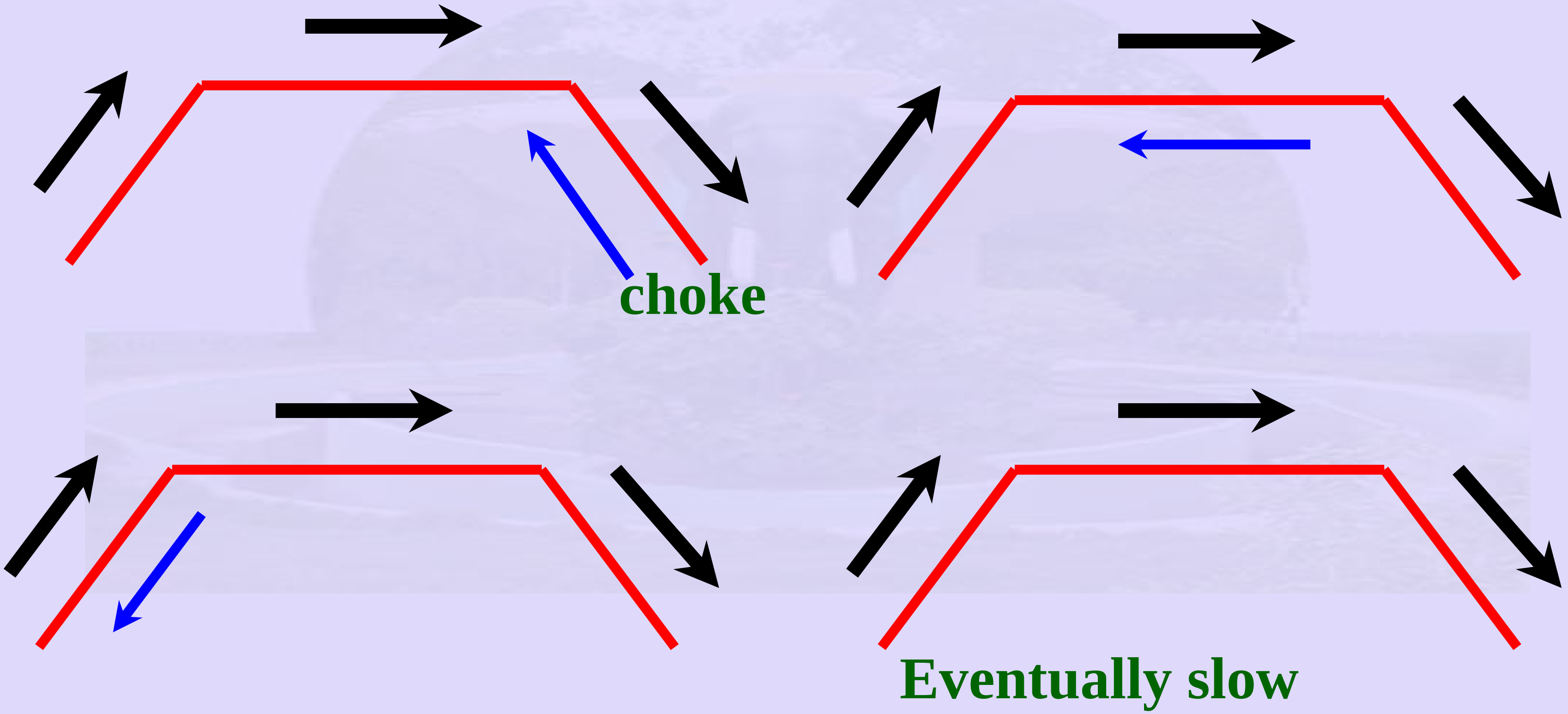
- Fair Queuing:
  - Multiple Queues for each output line, for each source
  - Router scans queues in RR fashion
- Issues:
  - More bandwidth to router with large packets
  - Byte by byte round robin

Congestion Control

  - Scan queue repeatedly until tick found at which packet done
  - Reorder packets in terms of time completion
- Weighted fair queuing:
  - Servers vs Clients
- Hop-by-hop choke packets

Hop-by-Hop Choke Packets

Too many choke packets → Congests link



Congestion Control

- Load shedding
  - Discard packets
    - question what to discard?
    - ftp – Keep old, discard new
    - audio/ video – keep new, discard old
    - need more intelligence:(?)
      - Some packets are more important
        - » Video – full frame(don't discard)- difference frame (discard)
        - » Sender prioritises packets!

Congestion Control

- Jitter Control Parameters:
  - Packets ahead/ delayed
  - Strategy flush packet furthest from it schedule first
- Multicast Routing Congestion ?
  - Single source multiple destination
  - RSVP - Resource reSerVation Protocol

Multicast Routing: Congestion

- Standard multicast
- Spanning tree covering all group members

- For better reception
  - Any receiver in a group can send message up spanning tree
  - Use reverse path forwarding
  - Reserve bandwidth at each hop

Flow Control

- Flow Control is specified end to end
  - Sliding window protocol
  - Fast sender vs. slow receiver
    - Sender does not overwhelm receiver
  - Advertisement of window size
    - receiver tells sender DIRECTLY
  - Process to process
- See More about flow control in TCP