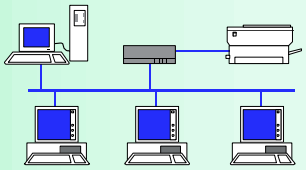


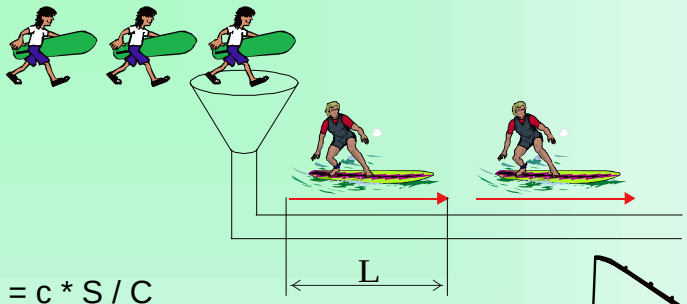
Distributed Systems
600.437
Advanced Networking Protocols

Department of Computer Science
The Johns Hopkins University

Advanced Networking Protocols

Lecture 10

Surfing On The Network Waves



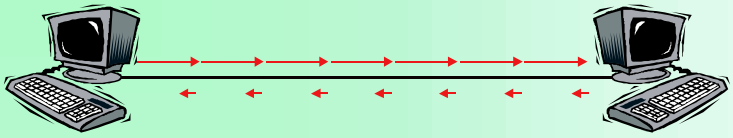
L = c * S / C

Where L = length of the packet
S = size of the packet
C = capacity of the network link
c = speed of light (electrons in the wire)

Surfing Packets…

ns2 Demo

Sliding Window Protocols



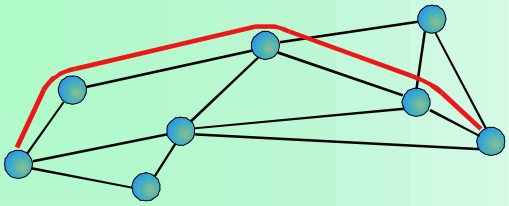
How big should the window size be ? (see exercise 1)

Ideally the window size will "occupy" the entire length of the network

- Maximum throughput (we can not send more than the network capacity)
- Minimum latency (no waiting time)

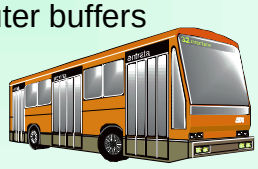
What if there are two senders, sending at the same time ?

Sliding Window Protocols (cont.)



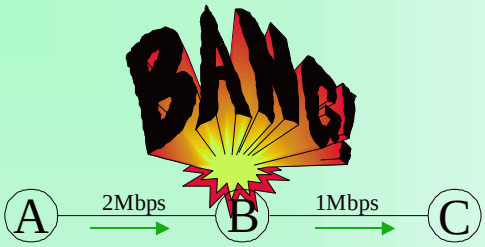
In reality we don't have a link to each destination in the network, but rather a set of different hops and intermediate routers (see traceroute)

If the window size is bigger than optimal, some packets will accumulate in the intermediate router buffers



Still maximum throughput
Increased latency (why ?)

What Is Congestion ?



Packets accumulate in intermediate buffers
Buffers have limited size => at some point packets will be dropped

Low throughput (why ?)
High latency

TCP Congestion Control

Go back n protocol [Jacobson 88]

cwnd – congestion window (dynamic)
ssthresh – slow start threshold

Two phases:

- **slow start** (cwnd < ssthresh)
  Multiplicative Increase Multiplicative Decrease
  cwnd initialized with 1
  for every ACK received, add **1** to cwnd

- **congestion avoidance** (cwnd > ssthresh)
  Additive Increase Multiplicative Decrease
  for every ACK received, add **1/cwnd** to cwnd

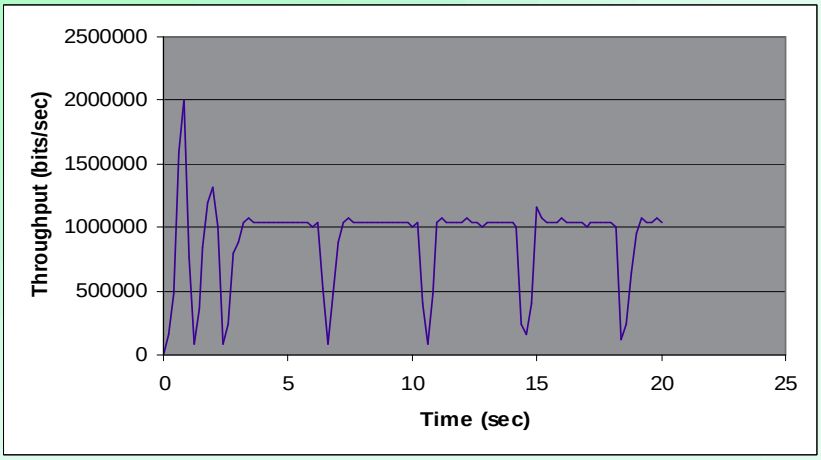
Packet loss: ssthresh = cwnd / 2
cwnd = cwnd / 2

Timeout: ssthresh = cwnd / 2
cwnd = 1

TCP Congestion Control (cont.)

ns2 Demo

TCP throughput



TCP throughput model

[Padhye 98]

Basic idea: $$Throughput = \frac{Window\_size}{Rond\_trip\_time}$$

More accurate: $$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)}$$

Where:

T: sending throughput
s: packet size
R: round-trip time
p: loss rate
t_{RTO}: retransmit timeout

Rate Based Congestion Control

[Floyd 00]

Receiver: computes probability of error and sends feedback to the sender
Sender: computes round-trip time, and adjust rate according to the formula

Advantages:
• smooth, almost no oscillations (as opposed to TCP)
• easy to implement at the sender

Disadvantages:
• difficult to compute p (error rate)
• p averaged => low responsiveness to changes

Flow Control

What if the receiver doesn't read fast enough ?

TCP flow control

- receiver sends to the sender its "advertised window" i.e. the empty slots in its receiving buffer
- sender uses the minimum between congestion window and advertised window

Outline

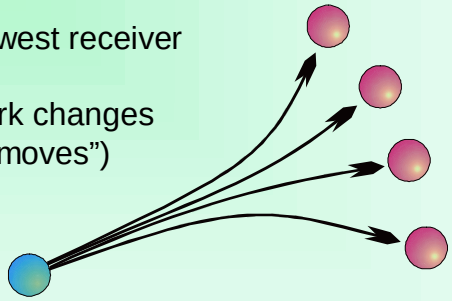
- **Unicast Congestion and Flow control**
  - The choice of a window size
  - TCP congestion control
- **Multicast Congestion Control**
  - PGMCC
- **Overlay Networks**
  - Benefits
  - Low latency reliable unicast
- **Multicast in Overlay Networks**
  - Online flow control for multi-session multicast

Congestion control for multicast

Can we use the TCP protocol for multicast ?

Problems:

- different receivers have different rates
  => send at the slowest receiver speed
- how to choose the slowest receiver
- how to adapt to network changes
  (the slowest receiver “moves”)
- ACK flooding



PGMCC

[Rizzo 00]

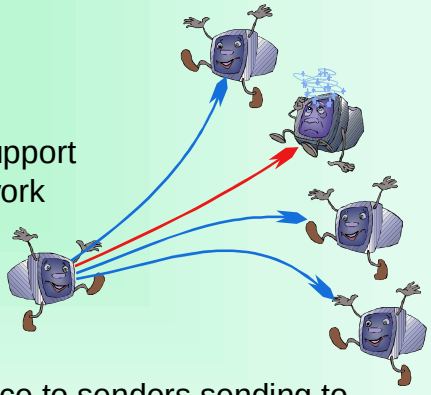
Congestion **C**ontrol based on **PGM** (Pragmatic General Multicast) Reliable Transport Specification

- Each receiver estimates **RTT** and **error rate**
- Receivers send feedback together with **NACK**s
- Based on RTT and error rate, the sender selects the slowest receiver
- Only the slowest receiver acknowledges every packet.
- The rest of them send only **NACK**s
- The sender performs as if it was running **TCP** with the slowest receiver

PGMCC (cont.)

Advantages:

- Fair with TCP
- Scalable
- Does not need router support
- Adapts very fast to network changes



Potential problems:

- Gives the same preference to senders sending to different number of receivers

Congestion Control Summary

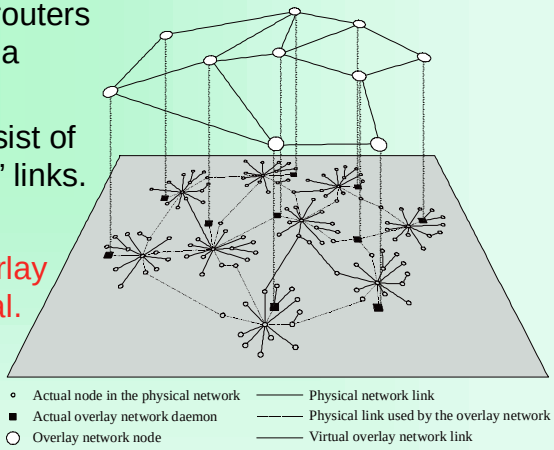
- Congestion occurs when the incoming traffic is **higher** than the capacity of the network – intermediate buffers drop packets
- Senders **slow down** by adjusting their sending rate
- All the participants should slow down **equally** in order to share the network resources fairly
- Single rate reliable multicast should send at the rate of the **slowest receiver**

Outline

- **Unicast Congestion and Flow control**
  - The choice of a window size
  - TCP congestion control
- **Multicast Congestion Control**
  - PGMCC
- **Overlay Networks**
  - Benefits
  - Low latency reliable unicast
- **Multicast in Overlay Networks**
  - Online flow control for multi-session multicast

Overlay Networks

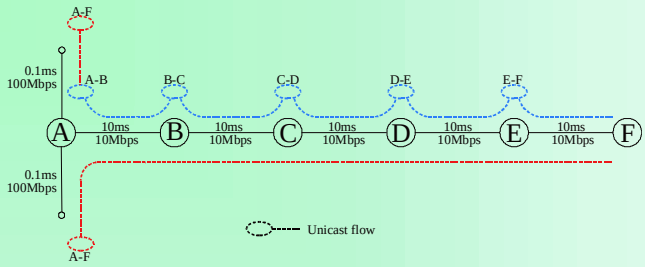
- Application-level routers working on top of a physical network.
- Overlay links consist of multiple “physical” links.
- Incurs overhead.
- Placement of overlay routers not optimal.
- **Flexible use of peer-protocols.**
- **Provides added value.**



The Case for Overlay Networks

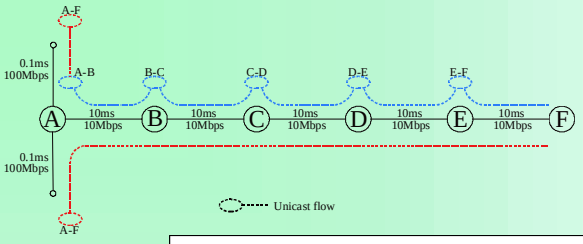
- Better network protocols can be developed.
  - Better performance, higher availability, higher security, etc.
- Real networks take a lot of time to change.
  - Especially once standards take hold (IP routers, 802.11 cards).
- Most distributed applications use only a small part of the Internet.
  - Overlay protocols do not need to scale to Internet size (number of nodes).
- Security advantages.
  - Having it built in.

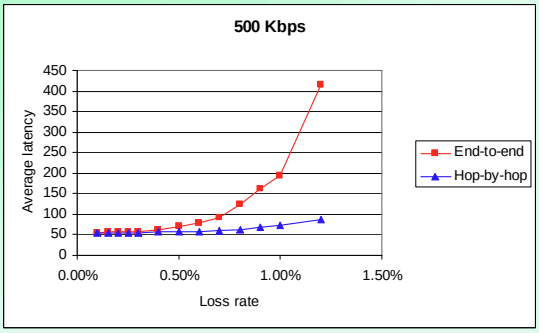
Low Latency Reliability



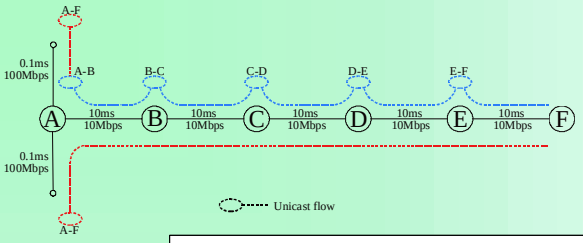
- Hop-by-hop reliability (”hop” is overlay hop)
  - **Packets are forwarded from hop to hop immediately.**
  - **No regard to FIFO ordering.**
- End-to-end ordering.
  - **For TCP user channels.**
  - **End-to-end reliability in case of network disconnections.**
- TCP congestion control on every overlay hop.
  - **Fair with respect to the rest of the Internet.**

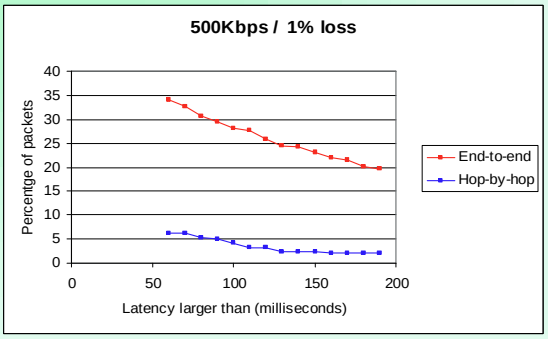
The Average-Latency Advantage





Latency Under Loss





Multicast Using Overlay Networks

One to many

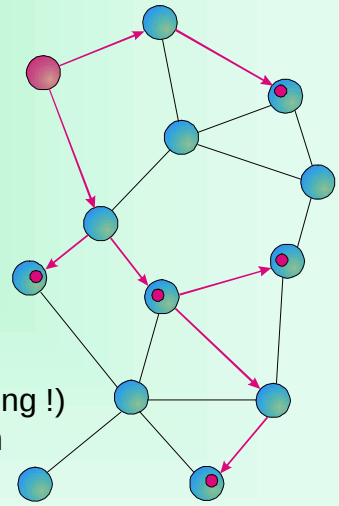
- used for broadcasting data

Many to many (more interesting)

- used in conferencing

Multi group, multi sender (amazing !)

- used in group communication



Multi-group Multi-sender Multicast in Overlay Networks

- Each node chooses its own (different) multicast tree
- Big number of (open) groups
- Big number of clients connected to each of the overlay nodes
- Each client can join any number of groups and send simultaneously to any number / combination of groups
- Flow control required !!!

Flow Control for Multi-group Multi-sender Multicast

[AADS 02]

End-to-end window

- Requires a window per sender per receiver per group
- Control traffic dependent on number of participants and groups
- Not dependent on number of intermediate links

Link-state

- Each sender is aware about the state of the overlay links that it uses, and adjusts its rate accordingly
- Control traffic dependent on number of links in the overlay network
- Not dependent on number of participants or groups

Online Cost-Benefit Flow Control

Summary:

- Each link is associated with a **cost** based on current resource utilization
- Each application stream is associated with application **benefit** for sending its messages
- Benefit is given to each application periodically
- Goal: maximize the **benefit of all applications** (number of messages sent on the network)
- **Online** decision of accepting/forwarding each packet
- **Competitive ratio** is the ratio between benefits achieved by the **Offline** and **Online** algorithms

Resource - Cost Function

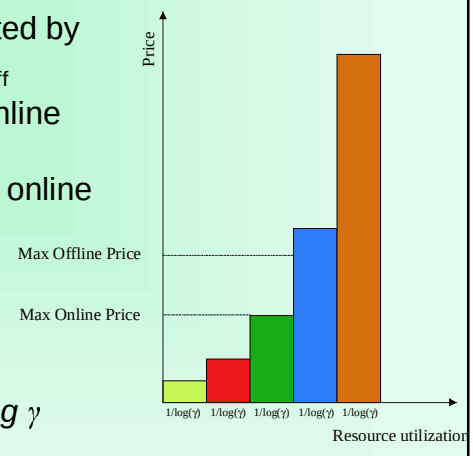
- Opportunity cost = benefit lost by *high benefit* connections as a result of consuming the resource by a *lower benefit* connection
- $C(u_l) = \alpha \cdot \gamma^{u_l}$
  has a $\Omega(\log \gamma)$ on competitive ratio (*Awerbuch, Azar, Potkin, 1993*)

u_l = utilization of link l
α = minimum benefit
β = maximum benefit
$\gamma = \beta / \alpha$

- Achievable if every *1/log* γ fraction of the utilized resource necessitates doubling the price

An Online Auction Example

- The maximum bid accepted by the offline algorithm is P_{off}
- If P_{off} is rejected by the online algorithm, at least *1/log* γ resource was sold by the online for at least ½ P_{off}
- Offline – Online < P_{off}
- It follows that Offline/Online < *1 + 2 * log* γ



Choosing The Resource

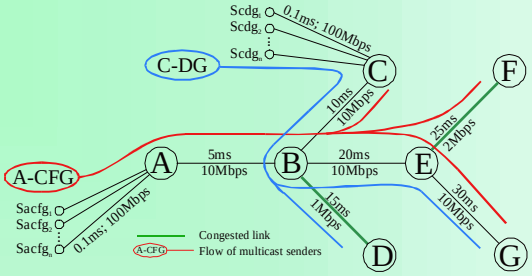
Potential resource: Link available bandwidth

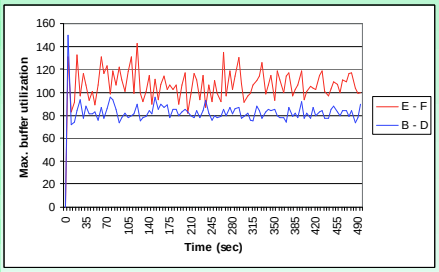
- Intuitively, this is the **resource** we try to control
- Dynamic, as a result of congestion control
- At the mercy of the external traffic
- Difficult to measure (usually requires some flooding)

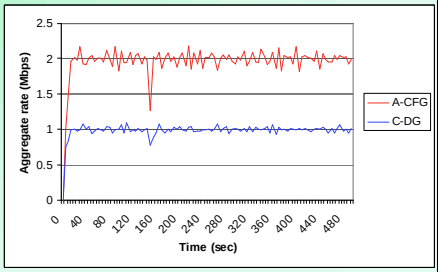
Potential resource: Link buffer

- Immediate indication of congestion
- Easy to control as a resource owned by the overlay network
- Easy to measure (locally available information)
- Extremely dynamic

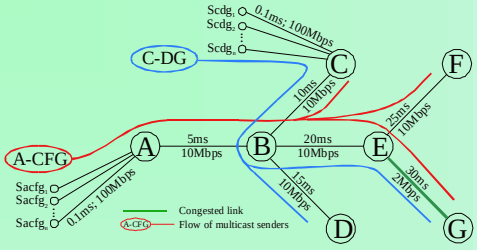
Simulation: Resource Utilization

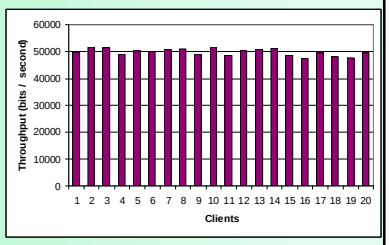


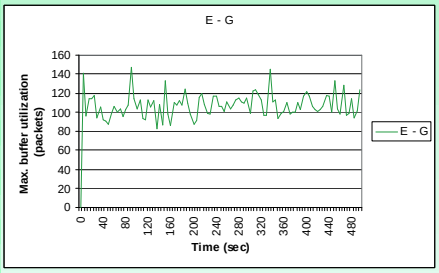


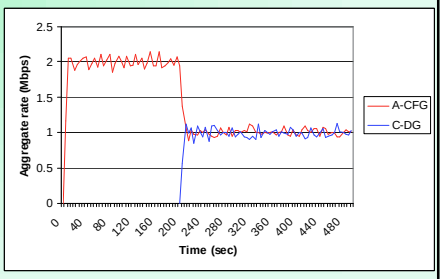


Simulation: Resource Sharing

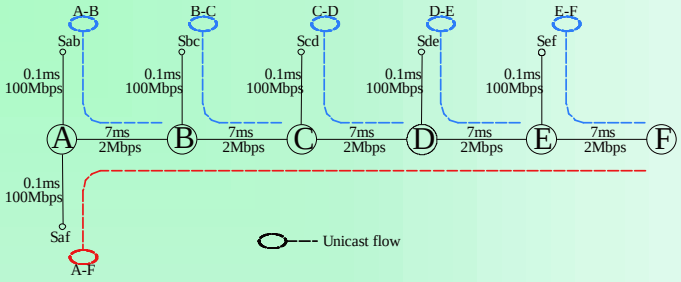


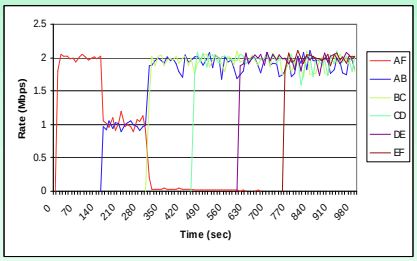


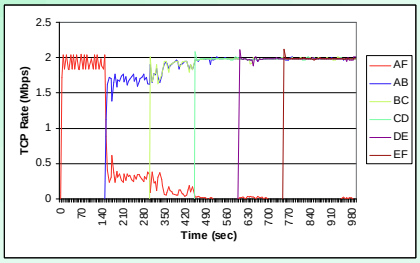




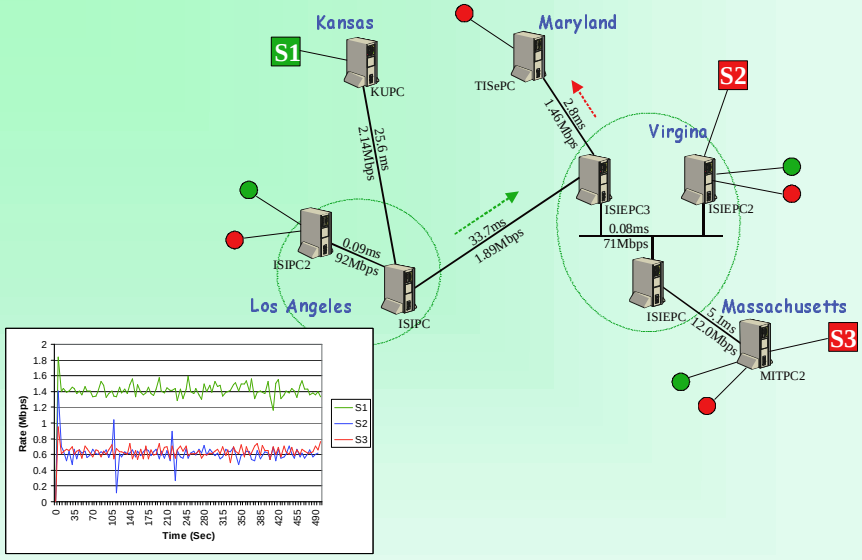
Simulation: Additive Cost In A Chain







Internet Results



Simulation: Resource Sharing

ns2 Demo