Congestion Control

CSC 343·643

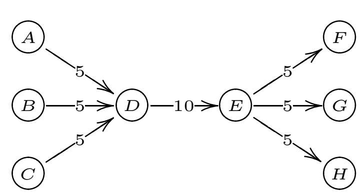
Wake Forest University
Department of Computer Science

Fall 2013

Congestion Control

Too many packets in a subnet degrades performance ⇒ **congestion**

- *What happens to performance under congestion?*
  - Queues at routers increase
  - Delays increase and packets are possibly dropped
- *How does congestion occur?*
  - Network resources are finite *Example resources?*



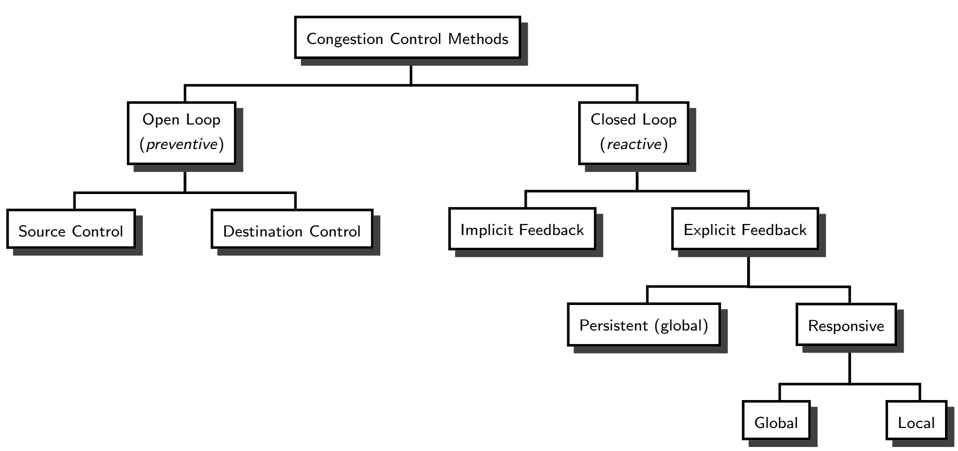
  - Total input rate greater than capacity $\sum r_i > c$ for $D - E$ link

- How do we control congestion?
  - Throttle traffic
  - Increase resource supply (typically not possible)
- **Flow control** relates to point-to-point traffic (sender and receiver)
  - Assure sender does not transmit faster than receiver can accept

    What is the difference between congestion and flow control?

    If flow control is working properly, will congestion occur?
- We are interested in methods that control congestion

Congestion Control Categories



Control Theory

Control theory point of view: **open loop** or **closed loop**

- Open Loop
  - Preventive method $\Rightarrow$ avoid congestion
  - No **feedback** (current system status) is used
  - Decide if/when a new session can start
- Closed Loop
  - Reactive method
    1. Monitor system
    2. Pass info to control points (feedback)
    3. Adjust system based on feedback

Is congestion control in the Internet open-loop or closed-loop?

Open Loop Congestion Control

- Control decisions do **not** depend on any feedback
  - Do **not** monitor network state
  - Requires knowledge and accounting of network resources
- Before session is allowed to start
  1. Determine session route and resource requirements
  2. Compare against resource availability
     - Form of Connection Admission Control (CAC)
  3. If allowed, must **police** session (use no more than contracted)

What are some problems associated with this approach?

Give an example, actual, network that uses open loop?

Closed Loop Congestion Control

- Control decisions based on network state
  - Monitor congestion
  - Control traffic based on measurements (feedback)
  - Feedback can be **explicit** or **implicit**
- Explicit feedback is sent as separate messages
  - Router may detect congestion and send control messages

    Any problems associated with explicit feedback?
- Messages not sent in implicit feedback

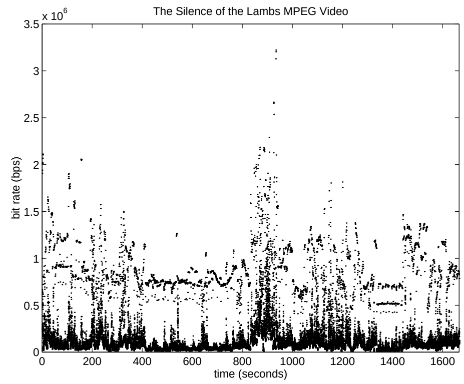
  If no messages, how is feedback collected?

- In addition, explicit feedback can be **persistent** or **responsive**
  - Persistent - Constant feedback provided (even if no congestion)
  - Responsive - Feedback provided only if congestion occurs
- Classic control problem, *feedback delay*
  - Measurements may be too *old* for accurate decision
  - Acting on old feedback may cause oscillations

  What are some problems associated with this approach?

Burstiness

- One main cause of congestion is that traffic is **bursty**
  - Multple packets/cells sent back-to-back
  - Prefer a predictable and **smooth** arrival pattern
- Consider MPEG compressed data



  - A new frame (picture) sent every 1/30 second

    We know when to expect a new frame, what is unpredictable?

  - So MPEG compression has ______ bit rate and ______ quality
- Transmit 5 cells (10 bits each) during a one second interval
  - Suppose the source can send in two fashions
    1. Send a cell every 1/5 second
    2. Send a cell every 1/10 second
  - Average rate for both is $\frac{5 \times 10 \text{ bits}}{1 \text{ sec}} = 50$ bps
  - However, second method sends all the packets in the first half of the second interval (**burst**)

Why are predictable sources more desirable?

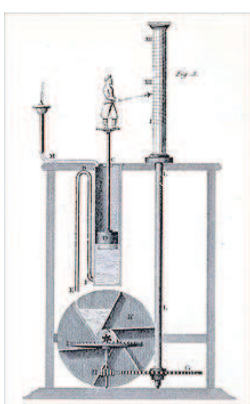
Traffic Shaping

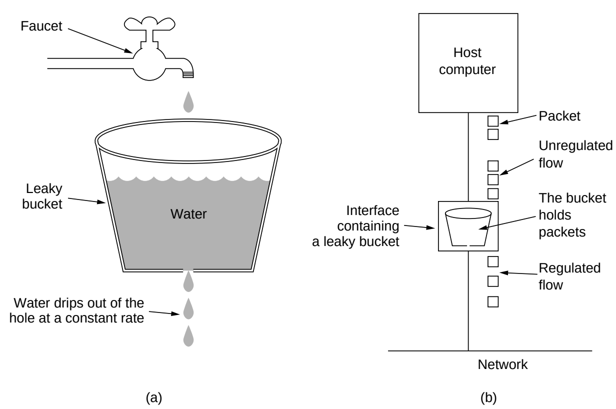
- Regulating average rate and burstiness of data transmission
- *Can we use the sliding window protocol?*
  - Limits amount transmitted at once (*helps manage the buffer space at the receiver*)
  - Does not limit the rate at which it is sent
- We will consider two traffic shaping mechanisms,
  - **Leaky bucket** and **token bucket**
- Shaping traffic is also referred to as **smoothing**

  What type of congestion control would use traffic shaping? Open loop (preventive) or closed loop (reactive)?

Leaky Bucket Algorithm

- Model - bucket with a hole near the bottom





  - Regardless of the rate water enters the bucket
    1. Outflow is constant ρ if any water present
    2. Zero if no water
- Same *bucket* idea can be applied to cells

- Conceptually, each source is connected to the network via an interface containing a leaky bucket (finite queue)
  - If a cell arrives and the queue is full, the cell is discarded
  - Otherwise it joins queue and awaits transmission
  - Server transmits one cell per *clock-tick*

    What does this smell like?

  - As a result, source can only send one cell per *clock-tick*

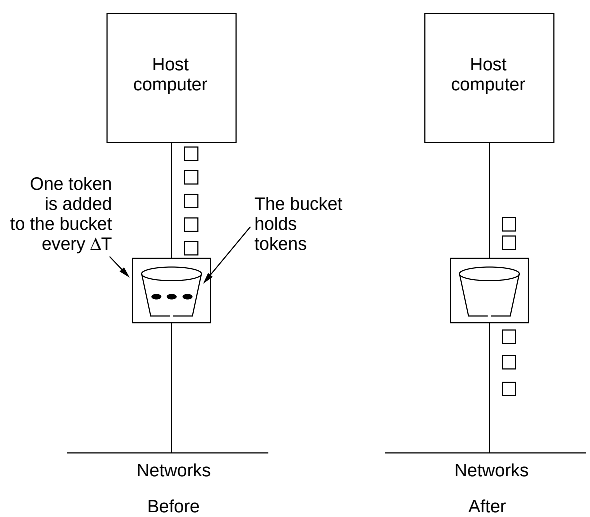
  If a source is bounded by when it can send a packet, how could a higher data rate be achieved?

- If packets are the same size the algorithm works as described
  - If packets have variable size, use number of bytes per *clock-tick*
- For example, assume the rule is 1024 bytes per *clock-tick*
  - For every *clock-tick* the source can send: a single 1024 byte packet, or two 512 bytes packets, ...
  - Also called **byte-counting leaky bucket**

Token Bucket Algorithm

- Leaky bucket enforces a rigid output rate ρ
  - However, some applications need a limited *speed-up* in transmission when a burst arrives
  - **Token bucket algorithm** allows limited burst
- Token bucket algorithm design
  - Leaky bucket holds **tokens**
  - Tokens generated by a clock, one token every ΔT seconds
  - Bucket has a maximum size (capacity) of c tokens
  - For a cell to be transmitted, it must remove one token from the bucket

Bursts are allowed, what is the maximum size?



- Token bucket provides a different type of traffic shaping
  - Allows idle source to *save tokens*
  - Up to c cells can be sent at once

- What if packets (variable length) are transmitted, not cells?
  - Let each token represent k bytes
  - Packet transmitted only if enough tokens are present

  What is a simple implementation of a token bucket?

- **Composite shapers** $\Rightarrow$ Leaky bucket + token bucket
  - Token bucket regulates the maximum burst size, would like to also bound the *peak rate*
  - Send the output of the token bucket into a leaky bucket (which governs the output rate)

Token Bucket Performance

- Let s = burst length (seconds), c = bucket capacity (bytes), ρ = token arrival rate (bytes/second), and m = maximum source rate (bytes/second)
- *What is the duration of a maximum-rate burst through a token bucket?*
  - Maximum bytes sent from the token bucket during a burst is

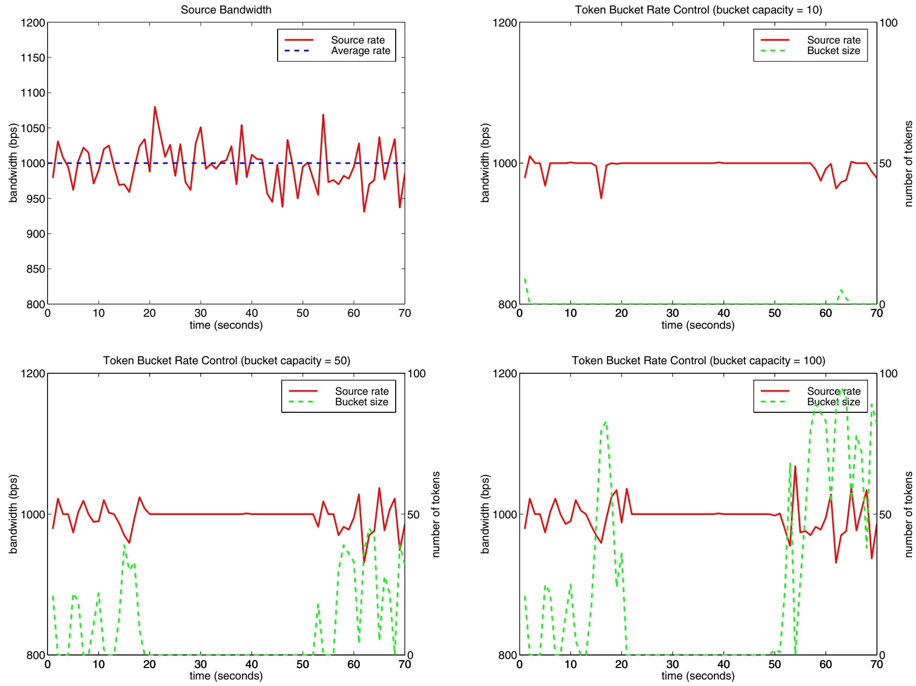
$$c + \rho \cdot s$$

  - Maximum bytes the source can send during a burst is

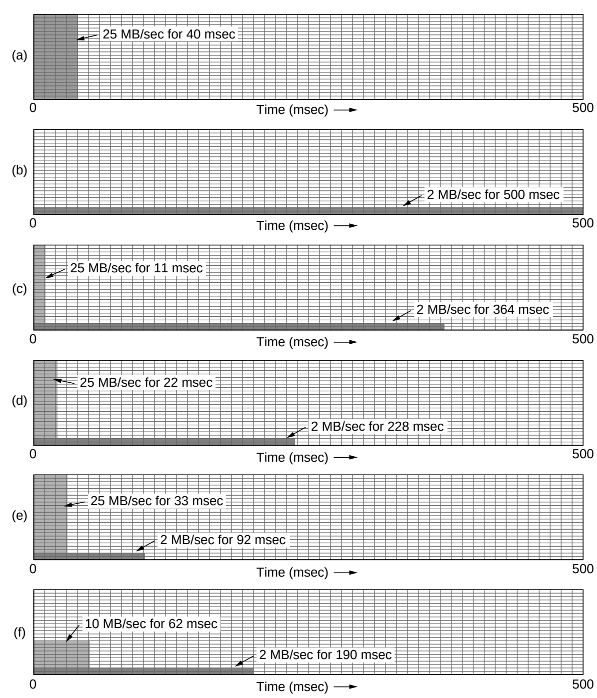
$$m \cdot s$$

  - Setting the two equal and solving for s

$$s = \frac{c}{m - \rho}$$



(*the larger the bucket, the burstier the output*)



(a) Input. (b) Output from a leaky bucket, draining at 2 MB/sec. (c - e) Output from a token bucket with token rate at 2 MB/sec and c =250, 500, 750 KB. (f) Output from 500 KB token bucket feeding a 10 MB/sec leaky bucket.

Current Traffic Shaper Ideas

- Name changes...
  - Leaky buckets are also referred to as *peak rate limiters*
  - Unfortunately, many call a *token bucket* a *leaky bucket* (token bucket is considered a leaky bucket variation).
- Possible traffic shaper modification
  - If a packet arrives and no token is available, it is *marked* then sent (instead of dropped)
  - The *mark* indicates the packet is *out-of-bounds* traffic
  - Routers are allowed to drop *marked* packets if congested

    What is the advantage of this modification?

Closed Loop Congestion Control

- Assume a router can monitor the utilization of an output link
- Let s be the sampled utilization and

$$u_n = \alpha \cdot u_{n-1} + (1 - \alpha) \cdot s$$

  where u is an estimate of the utilization (based on the current and previous measurements, as given by $0 \leq \alpha \leq 1$)

  Why add a history component?

- Once u moves above a threshold (watermark) **choke packets** are transmitted back to the senders
- Sender reduces rate when choke packet received

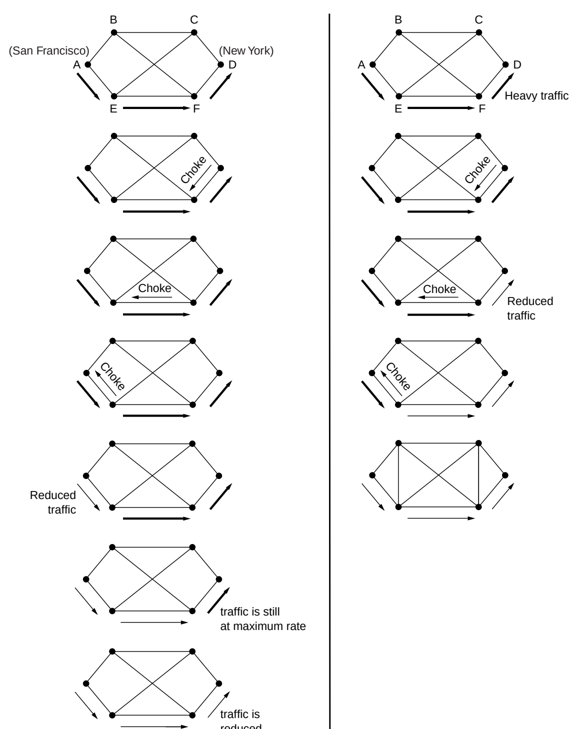
  What are some disadvantages to this approach?

Hop-by-Hop Choke Packets

155 Mbps sent from San Francisco (router A) to New York (router D)

- Assume router D becomes congested and send a choke packet to A
  - It would take 30 msec for the choke packet to reach A
  - During this time 4.6×10^6 bits are sent towards D
  - This data (in the pipeline) is possibly lost
  - *Upstream routers do not react to congestion*
- An alternative is to have the *upstream* routers respond
  - Choke packet reaches F which must immediately reduce its rate
  - F sends a choke packet to E which immediate reduce its rate
  - This repeats until a choke packet reaches the source

    So what must be done at each upstream router?



Choke packet effects only the sender

Choke packet effects each router

Price-Based Congestion Control

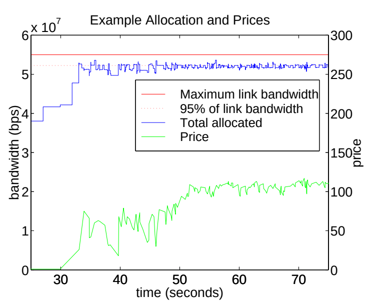
- Choke packets cause senders to reduce at the same rate
  - What if some applications need more link capacity than others?

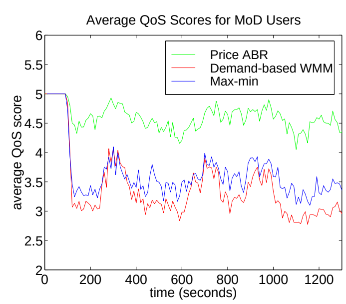
    Why not customize the choke packets?

- Congestion pricing is a *state-less* method to allocate bandwidth
  - If you want more, then pay for it! (competitive market)
  - *A view not shared by the Canadians (AKA French)...*
- Assume we wish to control link usage
  - Iteratively price capacity based on supply and demand
  - Send prices back to users
  - If the price increases demand will fall, and vice versa

- Price calculated as

$$p_{n+1}^i = p_n^i \cdot \frac{d_n^i}{\alpha \cdot s^i}$$





- Advantages
  - No per-connection state information (one price sent)
  - All types of fairness achieved
  - Provides security

What are the disadvantages?

Implicit Closed Loop Congestion Control

- The previous congestion control examples were explicit
  - Control messages sent from the router to the source

    While this provides congestion control, there are several implementation problems such as?

- Routers do **not** send messages with implicit feedback
  - Source and destination monitor data sent and received
  - React when congestion is detected

    How exactly can congestion be detected?

    Any problems?