Link-sharing and Resource Management Models for Packet Networks

Sally Floyd
Lawrence Berkeley Laboratory
floyd@ee.lbl.gov

February 7, 1995

(Joint work with Van Jacobson.
Credits to other members
of the Internet End-to-End Research Group)

Overview of talk:

- Overview of CBQ (class-based queueing)
- Statement of link-sharing goals
- Formal link-sharing guidelines
- Approximations to the Formal link-sharing guidelines
- Priority scheduling in a link-sharing framework
- Conclusions

And if there is extra time:

- Implementation issues
- Analysis of delay

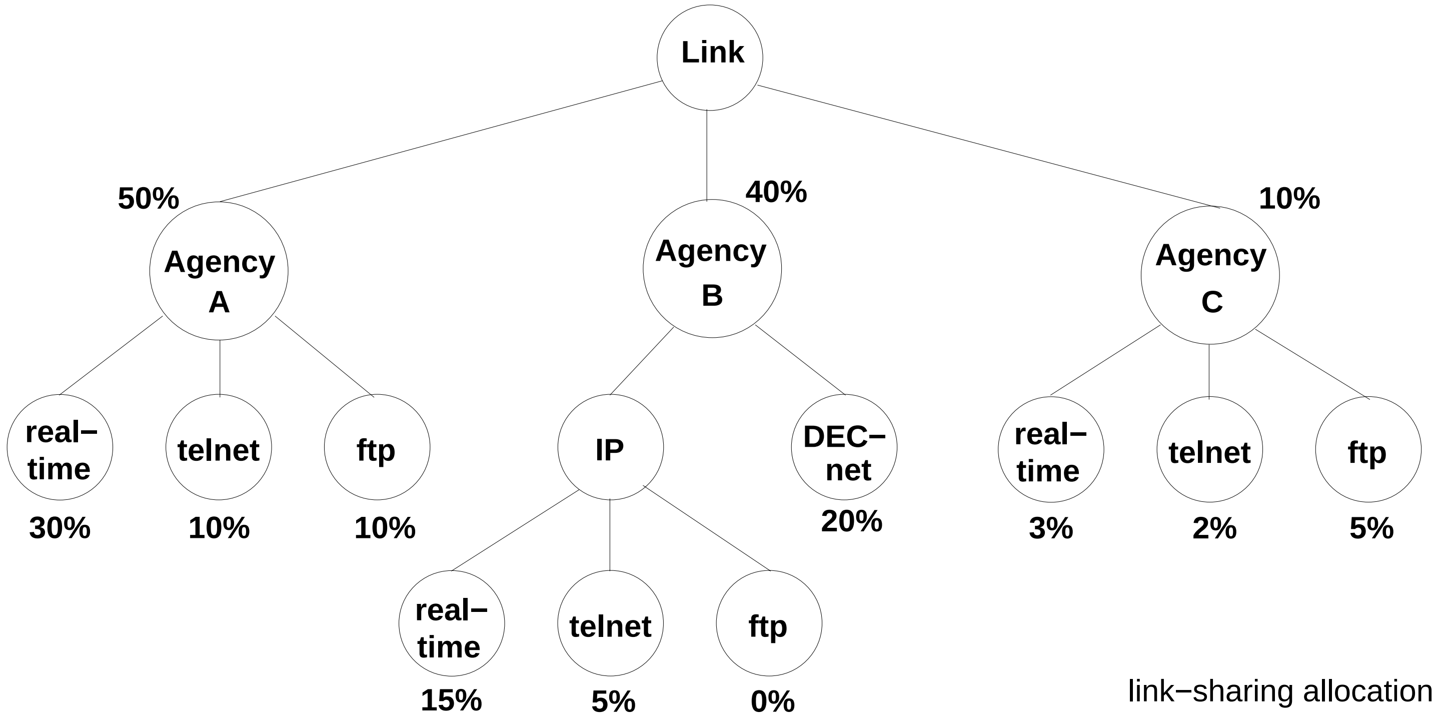
Policy goals for CBQ:

- Support connections that require bandwidth guarantees (e.g., packet voice and video).
- Support ‘quality of service’ (e.g., interactive telnet vs. bulk data ftp).
- Support flexible link sharing.

Methods:

- Separate low-level mechanisms from high-level policy, to allow evolution.
- Aggregate connections in classes. Each class has a priority and a throughput allocation.
- Construct a hierarchy of classes.
- Avoid extensive per-conversation parameterization.

Example class hierarchy:



- Link-sharing between organizations, protocol families, and/or traffic types

- Hierarchical link-sharing

- Different links in the network will have different link-sharing structures.

Mechanisms:

- Classifier: Map arriving packets to classes, using information in the packet header.
- Estimator: Compute a short-term estimate of the class's bandwidth.
- Selector: Find the class that is allowed to send the next packet. (In our proposal, look for the highest priority class, then use round-robin within classes of the same priority.)
- Delayer: For a class that is over its link-sharing allocation and contributing to congestion, compute the next time this class is allowed to send a packet. A delayed class is rate-limited to its allocated link-sharing bandwidth.

Link-sharing goals:

- Each interior or leaf class should receive roughly its allocated link-sharing bandwidth over appropriate time intervals, given sufficient demand.

- If all leaf and interior classes with sufficient demand have received at least their allocated link-sharing bandwidth, the distribution of any ‘excess’ bandwidth should not be arbitrary, but should follow some set of reasonable guidelines.

Non-goals:

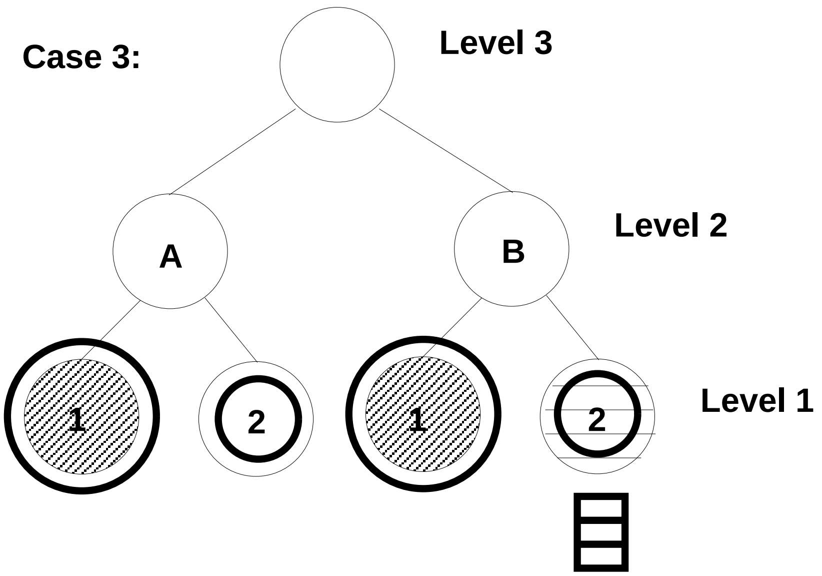
- Congestion-control/congestion avoidance

- Fair sharing (for some definition of “fair”)

“Non-goals” does not mean that these are non-problems. Other machinery solves these problems working within this framework.

Definition of terms:

- general and link-sharing scheduler
- regulated and unregulated classes
- overlimit, underlimit, and at-limit classes
- satisfied and unsatisfied classes
- levels in the class structure



Formal link-sharing guidelines:

A class can continue unregulated if one of the following conditions hold:

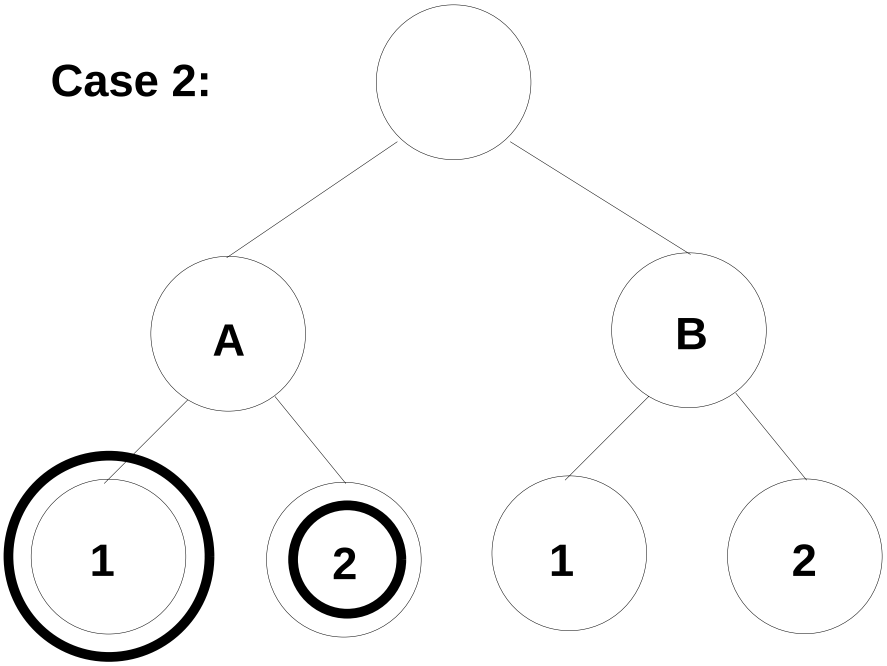
- **1:** The class is not overlimit, OR
- **2:** The class has a not-overlimit ancestor at level i, and there are no unsatisfied classes in the link-sharing structure at levels lower than i.

Otherwise, the class will be regulated by the link-sharing scheduler.

A regulated class will be rate-limited to its link-sharing bandwidth.

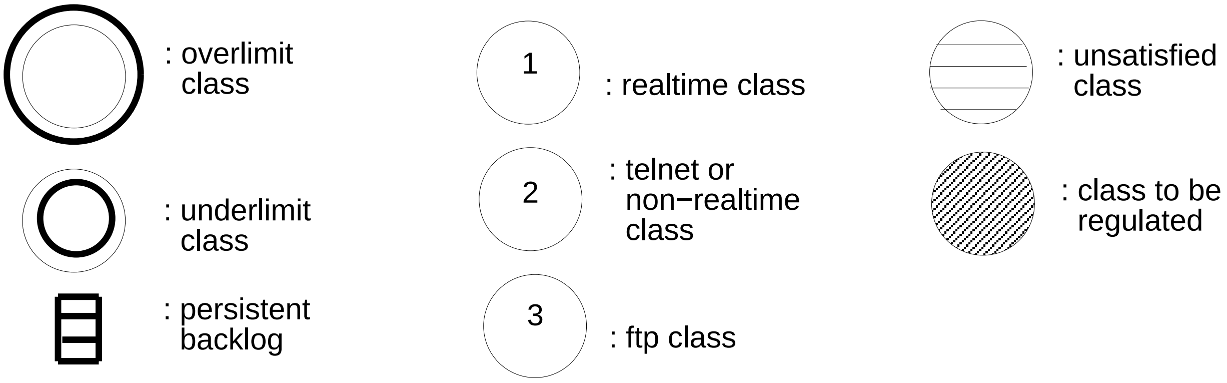
Examples of the link-sharing guidelines:

Case 2:



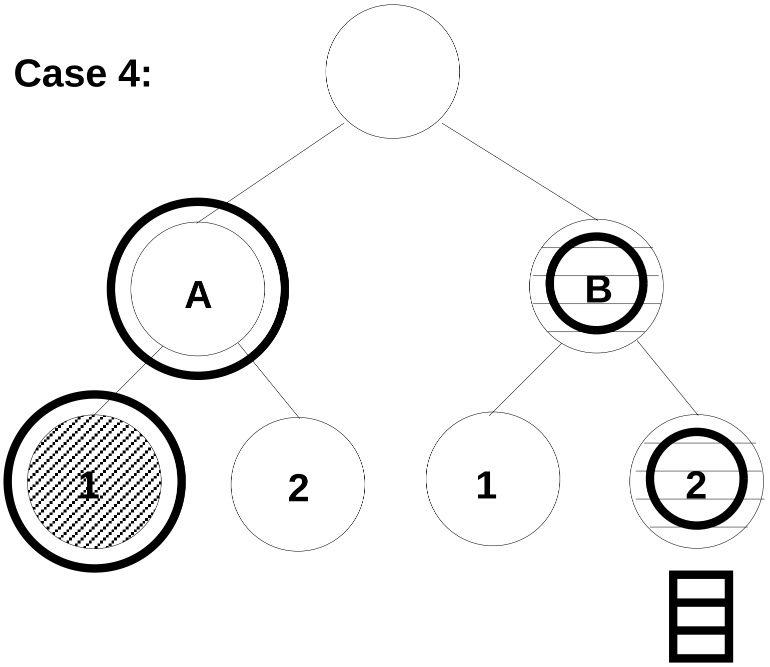
- No classes have to be regulated.

Legend:



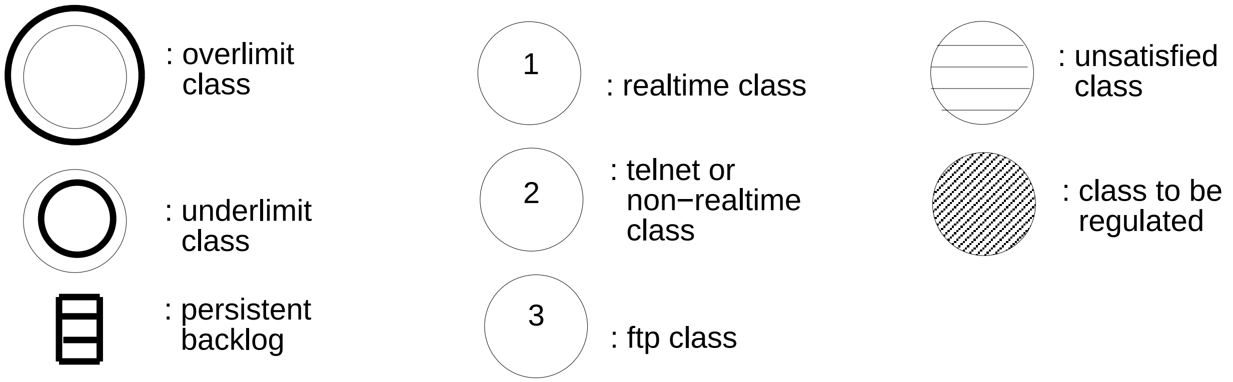
More examples of the link-sharing guidelines:

Case 4:



- Both the Agency B class and Agency B non-realtime class are unsatisfied. The Agency A realtime class needs to be regulated.

Legend:



: overlimit class

: underlimit class

: persistent backlog

: realtime class

: telnet or non−realtime class

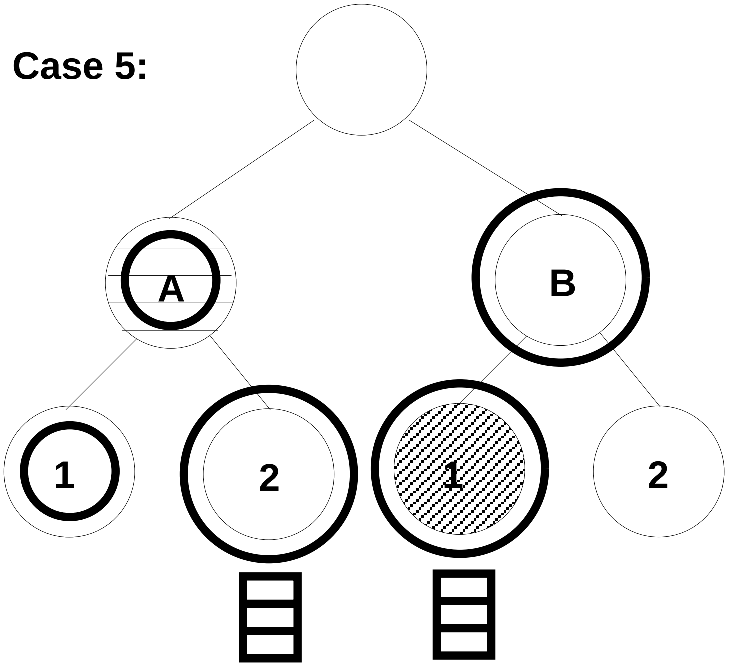
: ftp class

: unsatisfied class

: class to be regulated

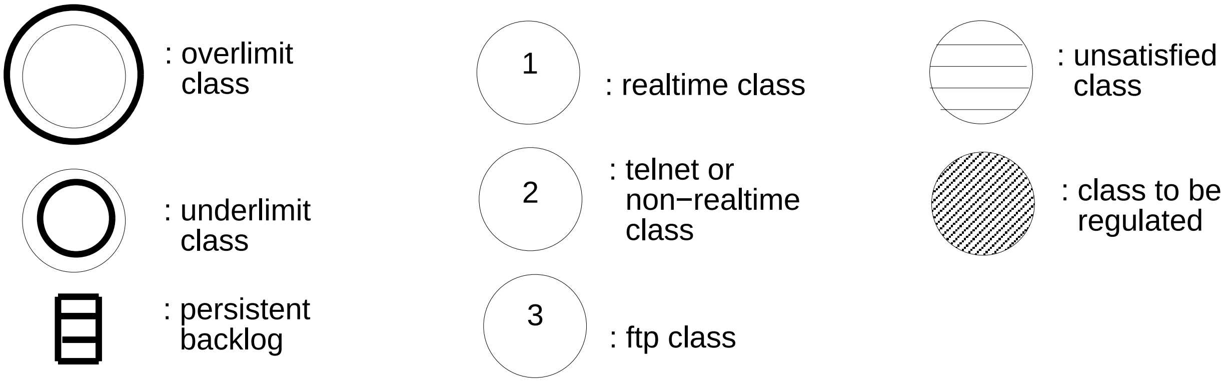
More examples of the link-sharing guidelines:

Case 5:

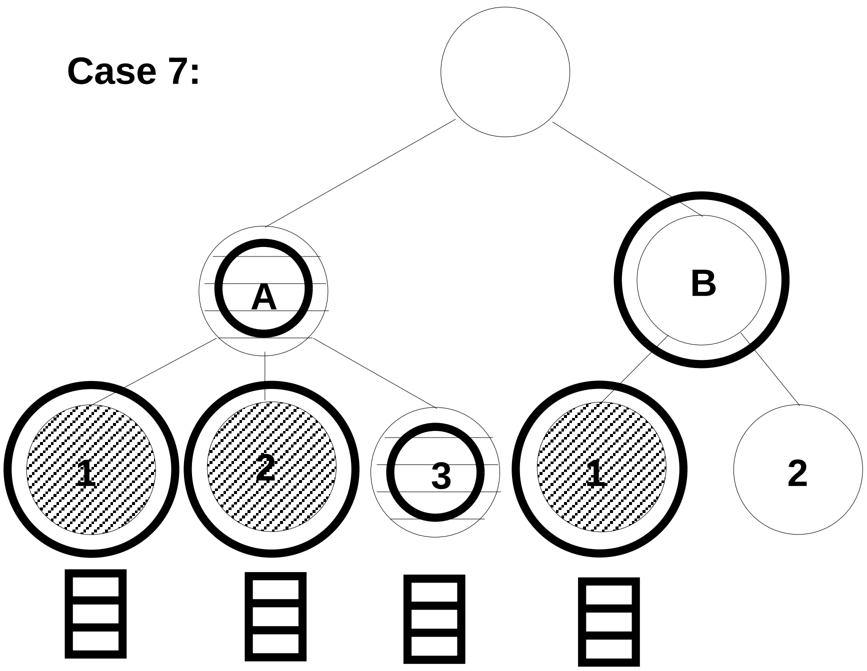


- The Agency A class itself is unsatisfied. The Agency B realtime class needs to be regulated.

Legend:

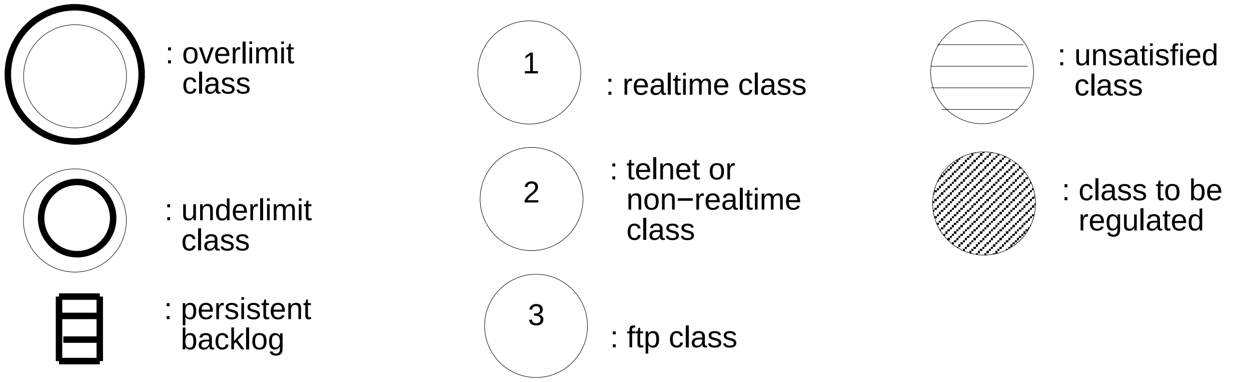


More examples of the link-sharing guidelines:



- Both the Agency A ftp class and the Agency A class itself are unsatisfied. All three overlimit leaf classes have to be regulated.

Legend:



Approximations to the Formal link-sharing guidelines:

Ancestors-Only link-sharing guidelines.
A class can continue unregulated if one of the following conditions hold:

- **1:** The class is not overlimit, OR
- **2:** The class has an underlimit ancestor.

Otherwise, the class will be regulated by the link-sharing scheduler.

- Advantages: ease and efficiency of implementation
- Disadvantages: performance is less robust

Approximations to the Formal link-sharing guidelines:

Top-Level link-sharing guidelines.
A class can continue unregulated if one of the following conditions hold:

- **1:** The class is not overlimit, OR
- **2:** The class has an underlimit ancestor whose level is at most *Top-Level*.

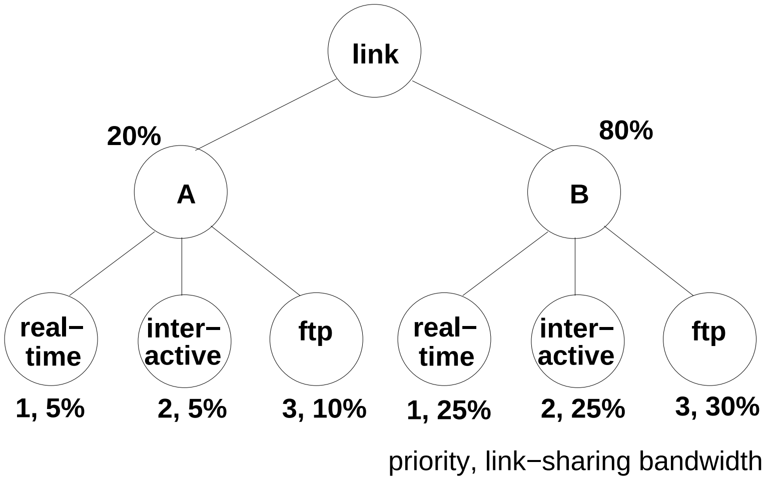
Otherwise, the class will be regulated by the link-sharing scheduler.

Heuristics for setting the Top-Level variable:

- *Top-Level* is set to the lowest level known to have an unsatisfied class.

Priority scheduling in a link-sharing framework:

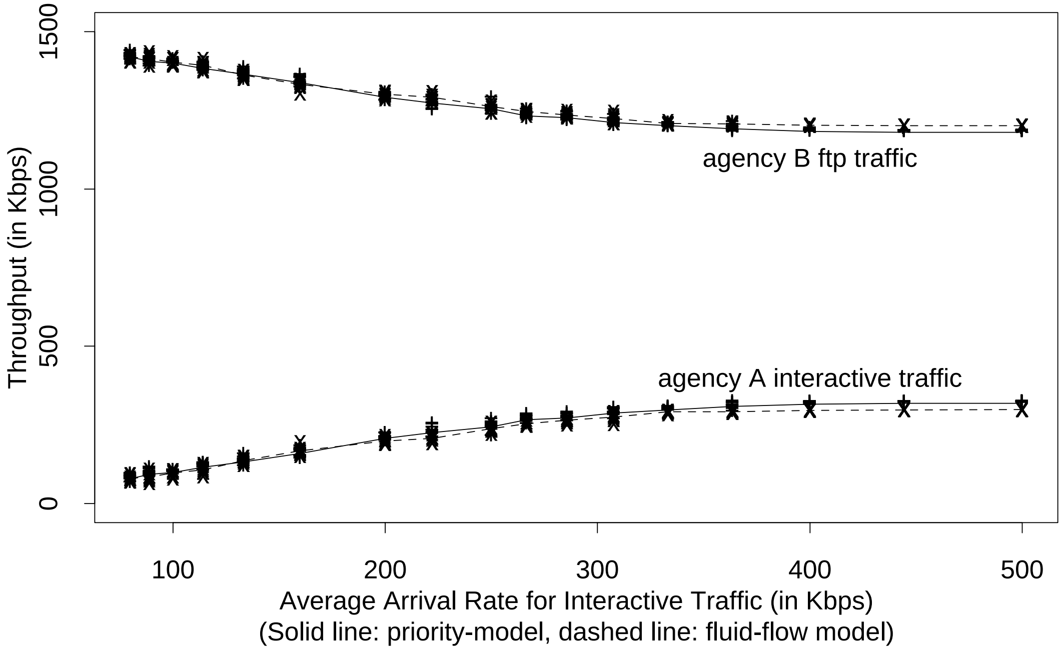
- Delay-sensitive and throughput-sensitive traffic
- Investigate the advantage (or lack of advantage) of incorporating priorities.

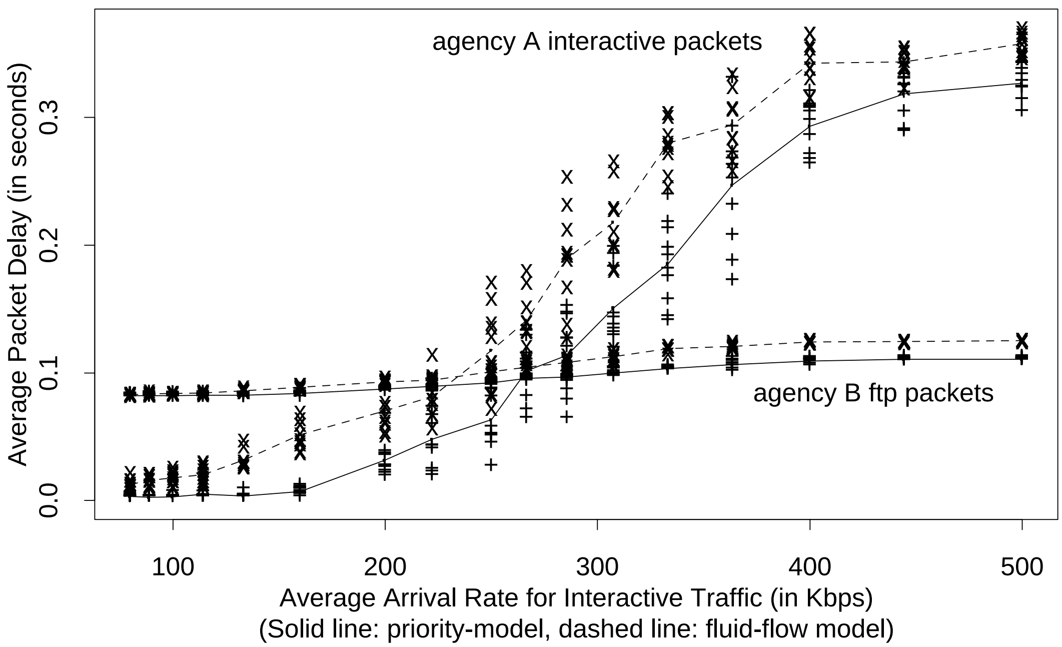


- Agency A interactive class: UDP, bursts of four 1000-byte packets are sent at exponential time intervals.
- Agency B interactive class: UDP, single 50-byte packets at exponential time intervals.
- FTP class: three TCP connections.
  Each class has its own queue with a 20-packet buffer.

Priority scheduling: simulation results

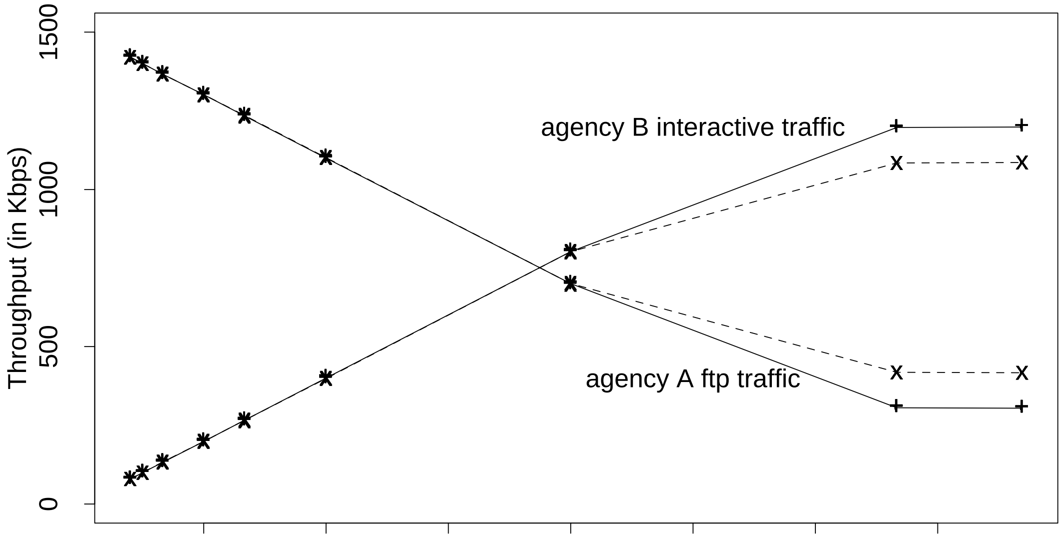
A higher-priority class with bursty arrivals, receiving a small fraction of the link bandwidth.



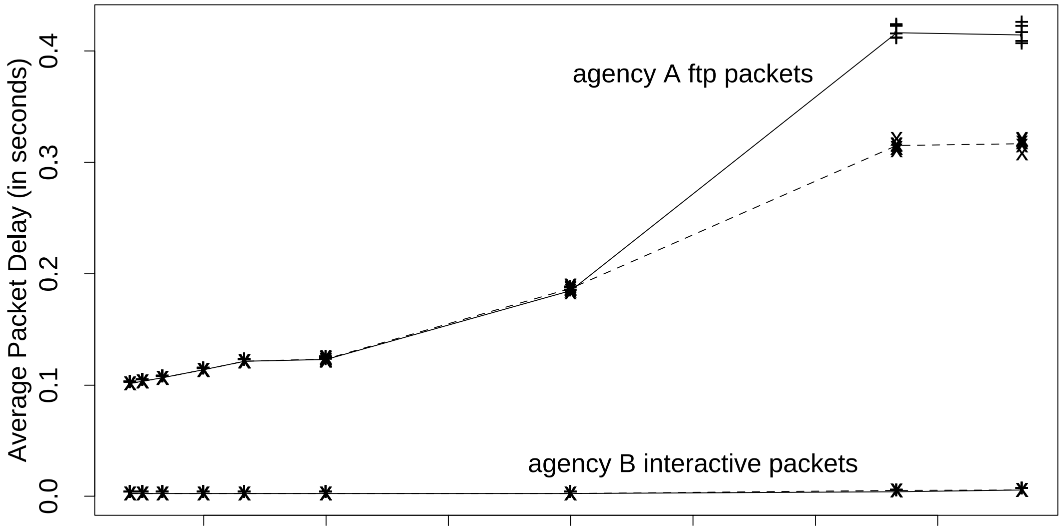


Priority scheduling: simulation results

A higher-priority class with smooth arrivals, receiving a large fraction of the link bandwidth.



Average Arrival Rate for Interactive Traffic (in Kbps)
(Solid line: priority-model, dashed line: fluid-flow model)



Average Arrival Rate for Interactive Traffic (in Kbps)
(Solid line: priority-model, dashed line: fluid-flow model)

Priority scheduling: Conclusions

- A higher-priority class with bursty arrivals receiving a moderate fraction of the link bandwidth. Priority scheduling reduces delay for the higher-priority class, without reducing throughput for the lower-priority class.

- A higher-priority class with fairly smooth arrivals, receiving a large fraction of the link bandwidth. Priority scheduling is of little benefit in reducing delay.

Link-sharing and realtime traffic:

- For a class of predictive service traffic, if the admission control procedure’s prediction of future traffic is incorrect, and the predictive service class becomes oversubscribed, the choice at the gateway is to limit the bandwidth of the predictive service class (e.g., using CBQ), or to allow starvation of lower-priority classes.

- For emerging realtime applications such as source- or receiver-based rate-adaptive video classes, one possibility would be for this traffic to be aggregated in a CBQ class with high-priority and an allocated bandwidth.

- CBQ machinery can be used to implement RSVP.

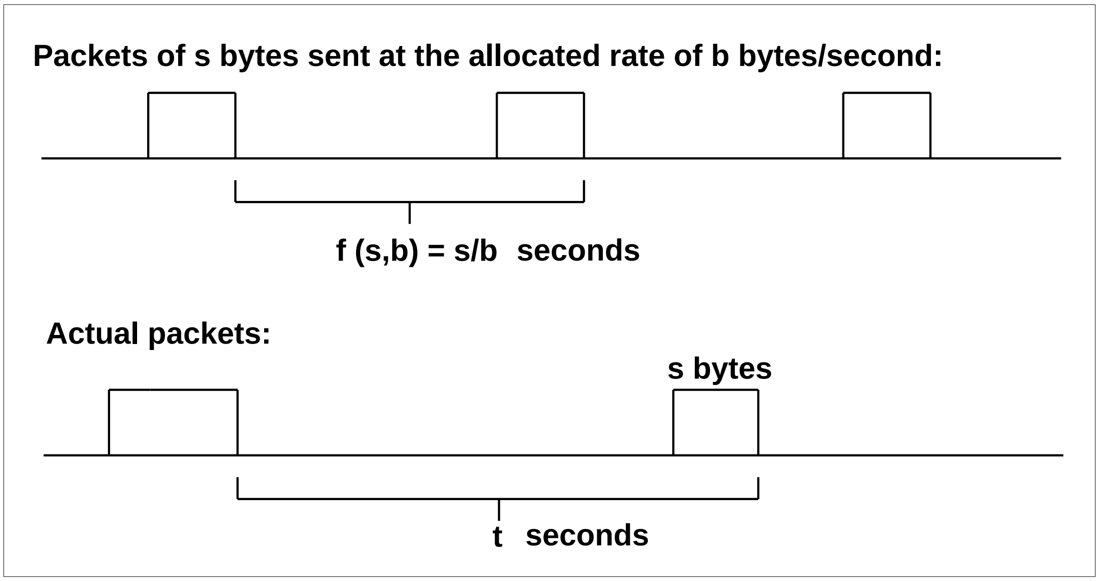
Related work:

- “Implementing Real Time Packet Forwarding Policies using Streams”, Wakeman et al., Usenix, January 1995.

- Hoffman, Implementation report on the LBL/UCL/Sun CBQ kernel, presentation to the RSVP Working Group of the IETF, Toronto, July 1994.

Implementation issues: the estimator.

- AllocTime[PacketSize]: the time between packets when the class transmits packets at the allocated rate.


Packets of s bytes sent at the allocated rate of b bytes/second:
f (s,b) = s/b seconds
Actual packets:
s bytes
t seconds

- InterPktTime: the actual time between packets.

- IDLE is InterPktTime - AllocTime[PacketSize].

- AvgIDLE is positive when the class is over its allocated rate, and negative otherwise.

Implementation issues: the classifier.

- Uses information in the packet to locate the appropriate class structure.

- A classifier implemented by Wakeman et al. is now being used with CBQ to control link-sharing in the FAT pipe.

Implementation issues: the selector.

- Inspects classes in priority order.

- Inspects classes at the same priority using weighted round-robin.
  (Each class at each round gets to send its weighted share in bytes, including finishing sending the current packet. That class's weighted share for the next round is decremented by the appropriate number of bytes.)

- Invokes the delayer for an overlimit class that is unable to borrow.

Implementation issues: the delayer.

Rate-limits overlimit classes to their allocated bandwidth.

Analysis: priority one classes.

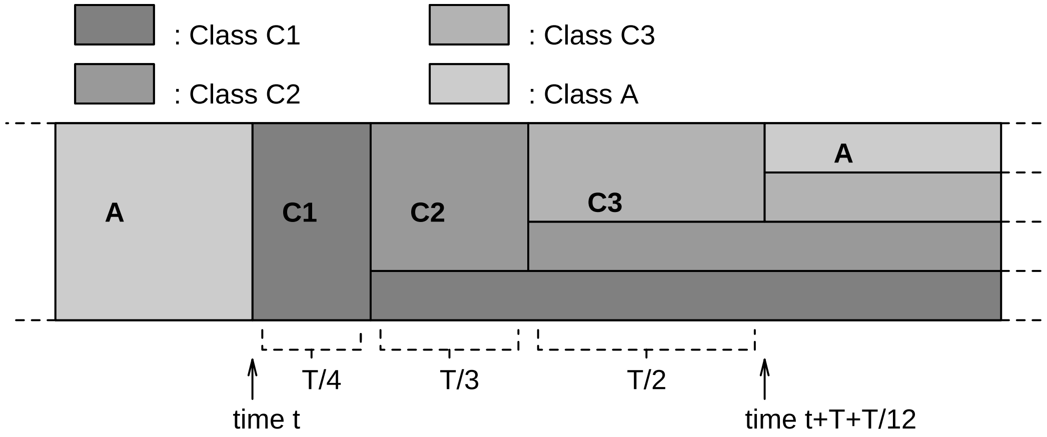
- A class that is not overlimit will not be regulated.

- When all classes are satisfied, no classes will be regulated.

- Assume a priority-based scheduler that uses weighted round-robin among classes of the same priority. Further assume that at most half of the bandwidth is allocated to priority-one classes.

  Then each priority-one class is guaranteed to receive its allocated share of the bandwidth in each round-robin round, unless some of this allocation was used ‘in advance’ in the most recently-sent packet from that connection.

Analysis: Limits on starvation for lower priority classes.



- Four classes with equal link-sharing allocations, where class A has the lowest priority. Class A can be denied bandwidth for $T + T/12$ seconds.

- Given n classes in the link-sharing structure, with link-sharing allocations p_1 - p_n, the lowest-priority class, with allocation p_n, could be denied bandwidth for at most

$$\sum_{i=1}^{n-1} \frac{p_i}{1-\sum_{j=1}^{i-1} p_j} T$$

seconds, where T seconds is the interval over which bandwidth usage is measured.

Conclusions:

- We believe that needs met by link-sharing are fundamental, given the presence of congestion. We do not believe that the need for controlled link-sharing is a transient stage that will disappear with the full commercialization of the Internet.

- It is not easy to fully anticipate the service requirements of emerging real-time applications on the Internet. We believe that mechanisms for controlled link-sharing add flexibility for satisfying requirements of emerging real-time applications.