. Improving the Performance of Interactive TCP Applications using Service Differentiation

W. Noureddine and F. Tobagi

Department of Electrical Engineering
Stanford University
Stanford, CA, USA

Proceedings of IEEE Infocom
New York, NY
June 2002

Introduction (1)

- Everyone has experience with bad delays
  - Interactive apps (audioconference, telnet, games) need response time about 150ms
  - Web needs about 5 seconds, with some Web applications (ie- stock trading) less
- Delays can be from overloaded servers
  - But content providers can fix
  - Concentrate on delays from network

Introduction (2)

- Telnet delay from typed character until echo
  - Includes transmission, propagation, queue
  - If loss, then TCP retransmit
- Web delay the same, but also from connection establishment
  - HTTP 1.0 has one connection per object
  - HTTP 1.1 allows multiple objects per conect.

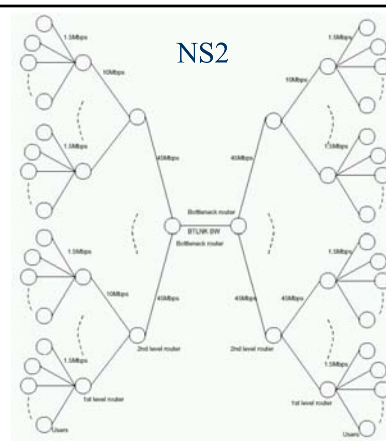
Introduction (3)

- Internet designed for throughput
  - TCP probes for maximum data rate even if causes loss
    - Periods of sending followed by idle (time-out)
  - Large queues because increases utilization
  - But not necessarily best for interactive applications!
- This paper
  - Classes of traffic (in DiffServ)
  - Telnet > Web > FTP
  - Window size for Web

Outline

- Introduction (done)
- Simulation Setup (next)
- The Effects of Congestion
- QoS Framework
- App Based Differentiation
- TCP-State Based Differentiation
- Conclusions

NS2



* 800 hosts
- 400 pairs
* Bwidth
-1.5 (T1)
-10 (LAN)
-155 (T3)
-Vary bttlnk RTTs
-20–200ms
* Buffers
-64 (T1)
-64 (LAN)
-250 (T3)
-500 (Bttl)
-(Smallish, so congstn)

Traffic Models (1)

- TCP
  - NewReno (most common on Internet)
  - Receiver unlimited window
- Telnet
  - Limited by Nagle's algorithm (*what is that*?)
  - Send 100 byte packet (includes MAC+TCP/IP hdr) and wait for echo
  - Random wait, but about 5 chars per second (rate of a fast typist)
  - Performance measure is *echo delay*
  - Aggregate telnet traffic less than 2 Mbps

Traffic Models (2)

- HTTP
  - 1.0 – Index page plus 4 parallel connections. Close each between object
  - 1.1 – Index page plus all objects requested and sent over one connection
  - Number and size from [16]
  - "Think time" is 2.5 seconds (gives heavy use)
  - 5 out of 400 are for measurement
    - 81 KB, 1 KB index with 8 10 KB images
    - Performance is *download time*
  - Aggregate traffic is 33 Mbps

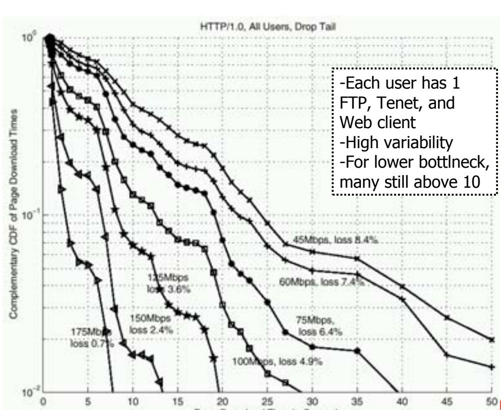
Traffic Models (3)

- FTP
  - Pareto (heavy tail) size, average 200 KB
  - Delay average 2 seconds between
  - 10 probe sessions (out of how many?) of 200 KB
    - Performance is *transfer time*
  - Bandwidth is elastic but cannot fully utilize more than 100 Mbps by itself
  - Also, FTP traffic along the reverse path to get competing traffic for acks

Outline

- Introduction (done)
- Simulation Setup (done)
- The Effects of Congestion (next)
- QoS Framework
- App Based Differentiation
- TCP-State Based Differentiation
- Conclusions

Effects of Congestion

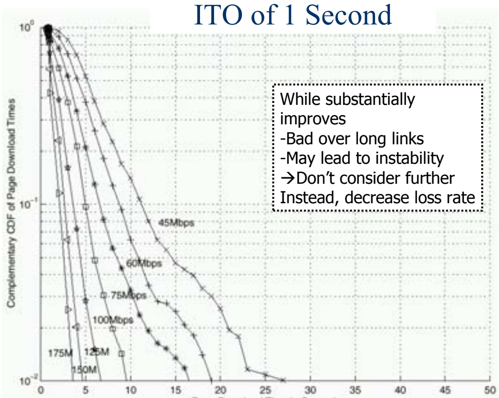


-Each user has 1 FTP, Tenet, and Web client
-High variability
-For lower bottlneck, many still above 10

TCP with Small Windows

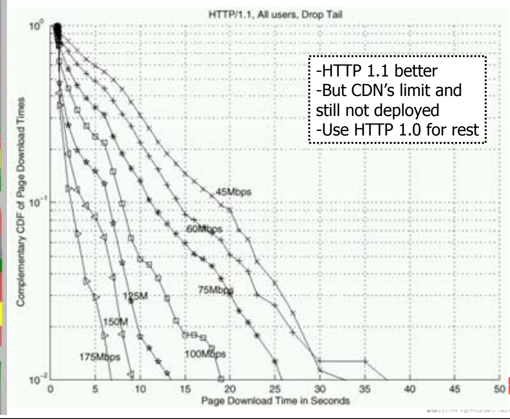
- HTTP 1.0 has high number of connection establishments
- SYN packet lost is costly to recover
  - Initial Time Out (ITO) typically 3 or 6 seconds
- Small window doesn't allow 3 duplicate acks
  - Retransmission Time Out (RTO) min 1 sec
- Work has proposed better clock granularity and timer min [3]

ITO of 1 Second



While substantially improves
-Bad over long links
-May lead to instability
→Don't consider further
Instead, decrease loss rate

Effects of Congestion



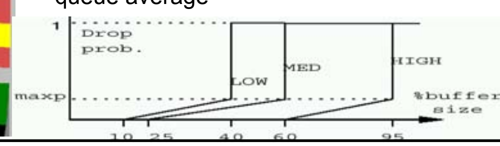
-HTTP 1.1 better
-But CDN's limit and still not deployed
-Use HTTP 1.0 for rest

Outline

- Introduction (done)
- Simulation Setup (done)
- The Effects of Congestion (done)
- QoS Framework (next)
- App Based Differentiation
- TCP-State Based Differentiation
- Conclusions

Prioritized Dropping

- Use DiffServ's Assured Forwarding (AF) [8]
  - Four classes defined
  - Each with 3 drop precedence levels
- Only consider TCP traffic, but (unresponsive) UDP traffic (marked and policed) would be another class
- RED with 3 priorities [18], each has EWMA queue average



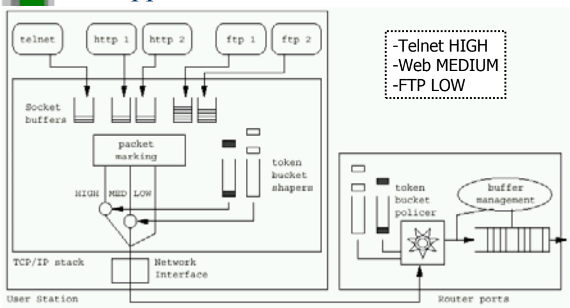
SLAs and Pricing

- Users and network providers work on Service Level Agreements (SLAs)
  - Limit aggregate rates of HIGH and MED
  - Specify per-user limits and allowable burst sizes
- Users pre-mark own traffic
- Network provider polices marks
  - Can be done at edge of network

Outline

- Introduction (done)
- Simulation Setup (done)
- The Effects of Congestion (done)
- QoS Framework (done)
- App Based Differentiation (next)
- TCP-State Based Differentiation
- Conclusions

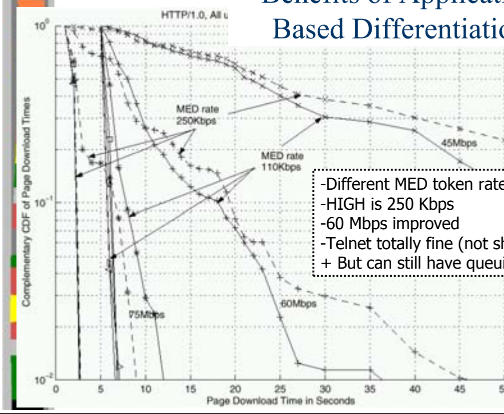
Application Based Differentiation



-Telnet HIGH
-Web MEDIUM
-FTP LOW

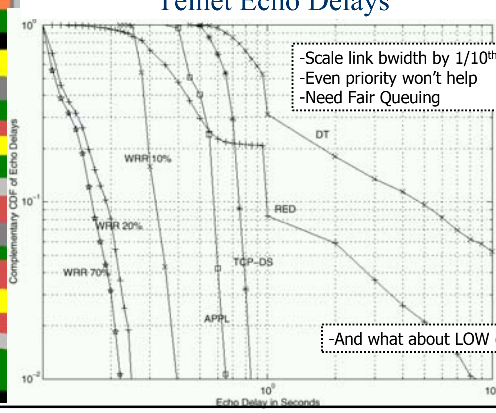
-Token bucket shapers to get flow rate
-End host does it since edge network will, too

Benefits of Application Based Differentiation



-Different MED token rates
-HIGH is 250 Kbps
-60 Mbps improved
-Telnet totally fine (not shown)
+ But can still have queuing delay

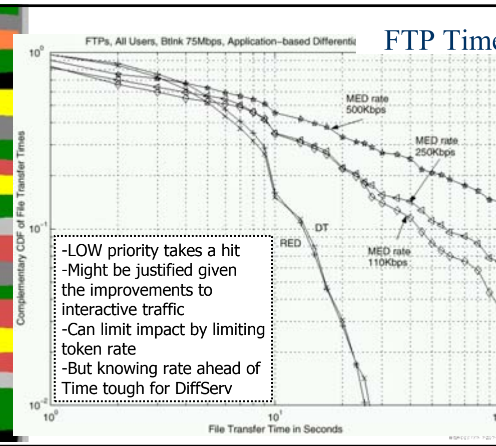
Telnet Echo Delays



-Scale link bwidth by 1/10th
-Even priority won't help
-Need Fair Queuing

-And what about LOW (FTP)?

FTP Times



-LOW priority takes a hit
-Might be justified given the improvements to interactive traffic
-Can limit impact by limiting token rate
-But knowing rate ahead of Time tough for DiffServ

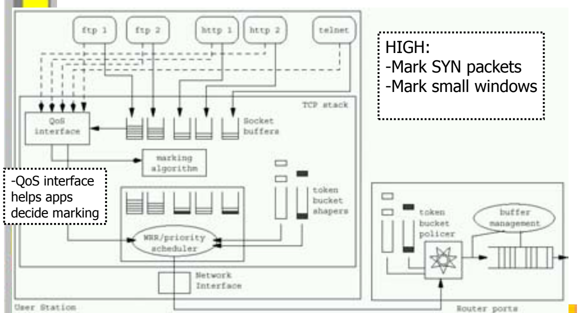
Difficulties in Marking Web Traffic

- Web traffic not all the same size
  - Often use Web for large file transfers
  - Even stream video over HTTP (long)
- Large transfers may interfere with interactivity of small transfers
- And don't always know size ahead of time (increasingly dynamically generated)

→ Solution, is to mark individual packets

Outline

- Introduction (done)
- Simulation Setup (done)
- The Effects of Congestion (done)
- QoS Framework (done)
- App Based Differentiation (done)
- TCP-State Based Differentiation (next)
- Conclusions



HIGH:
- Mark SYN packets
- Mark small windows

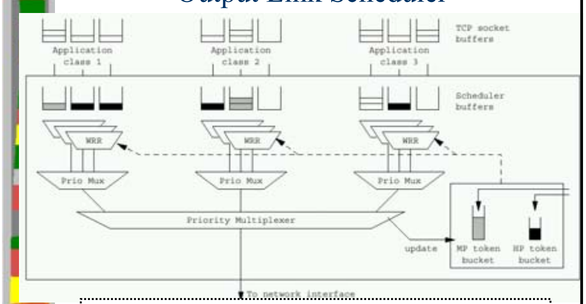
- QoS interface helps apps decide marking

Marking Algorithm

if sendwnd $\leq HIGH_{thresh}$
mark packet as HIGH
else if SYN or fast retransmit or fast recovery
mark packet as HIGH
else if sendwnd $\leq MED_{thresh}$
with probability $\frac{HIGH_{thresh}}{sendwnd}$
mark packet as HIGH
mark packet as MED
else
with probability $\frac{HIGH_{thresh}}{sendwnd}$
mark packet as HIGH
with probability $\frac{MED_{thresh}}{sendwnd}$
mark packet as MED
mark packet as LOW

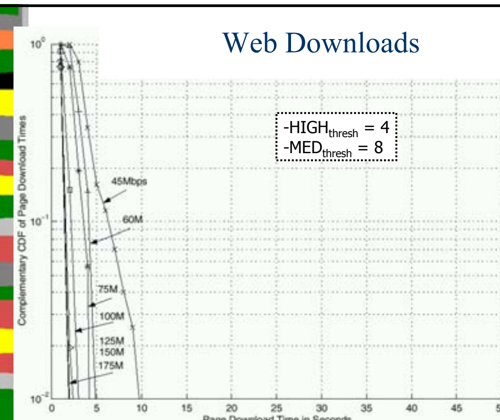
- Italics optional to smooth out abrupt changes
- $HIGH_{thresh}$ for Telnet large
- $HIGH_{tresh}$ for Web can be size of small object
- Users could purchase more HIGH

Output Link Scheduler



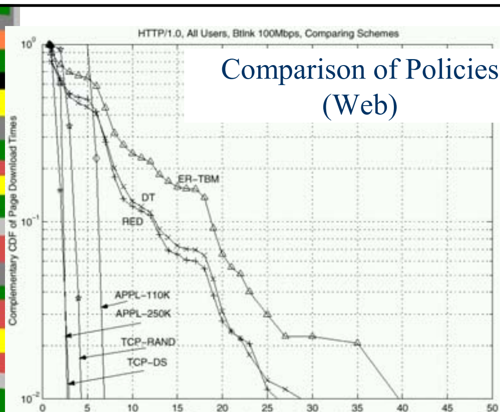
- Queue per application to prevent out-of order
- Send HIGH, MED, LOW from one class
- Go to next class. If upper class blocked, then cannot send (prevents small packets from starving higher priority)

Web Downloads

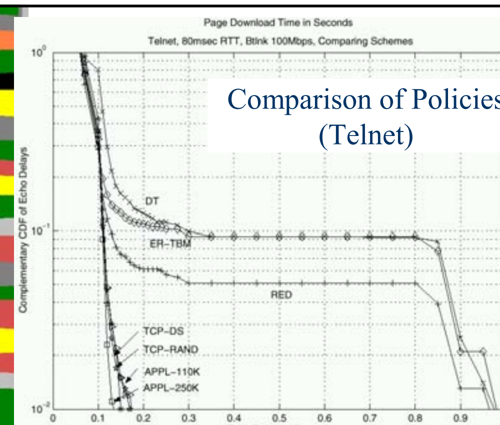


- $HIGH_{thresh}$ = 4
- MED_{thresh} = 8

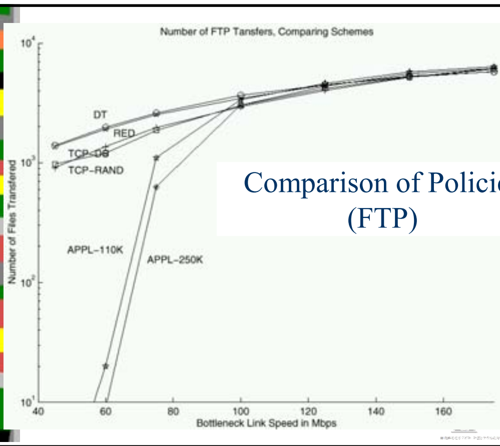
Comparison of Policies (Web)



Comparison of Policies (Telnet)



Comparison of Policies (FTP)



Conclusions

- Focus on congestion-induced delays
- Show how to reduce using multiple network service levels
  - Preference given to interactive applications
- Study affect of TCP state differentiation
- Good user-perceived performance can be achieved, without degrading other applications

Future Work?

Future Work (me)

- Other topologies
- Worry about complication of marking schemes
- Build application that uses QoS
- Streaming?