Addressing the Challenges of Web Data Transport

Venkata N. Padmanabhan

Microsoft Research

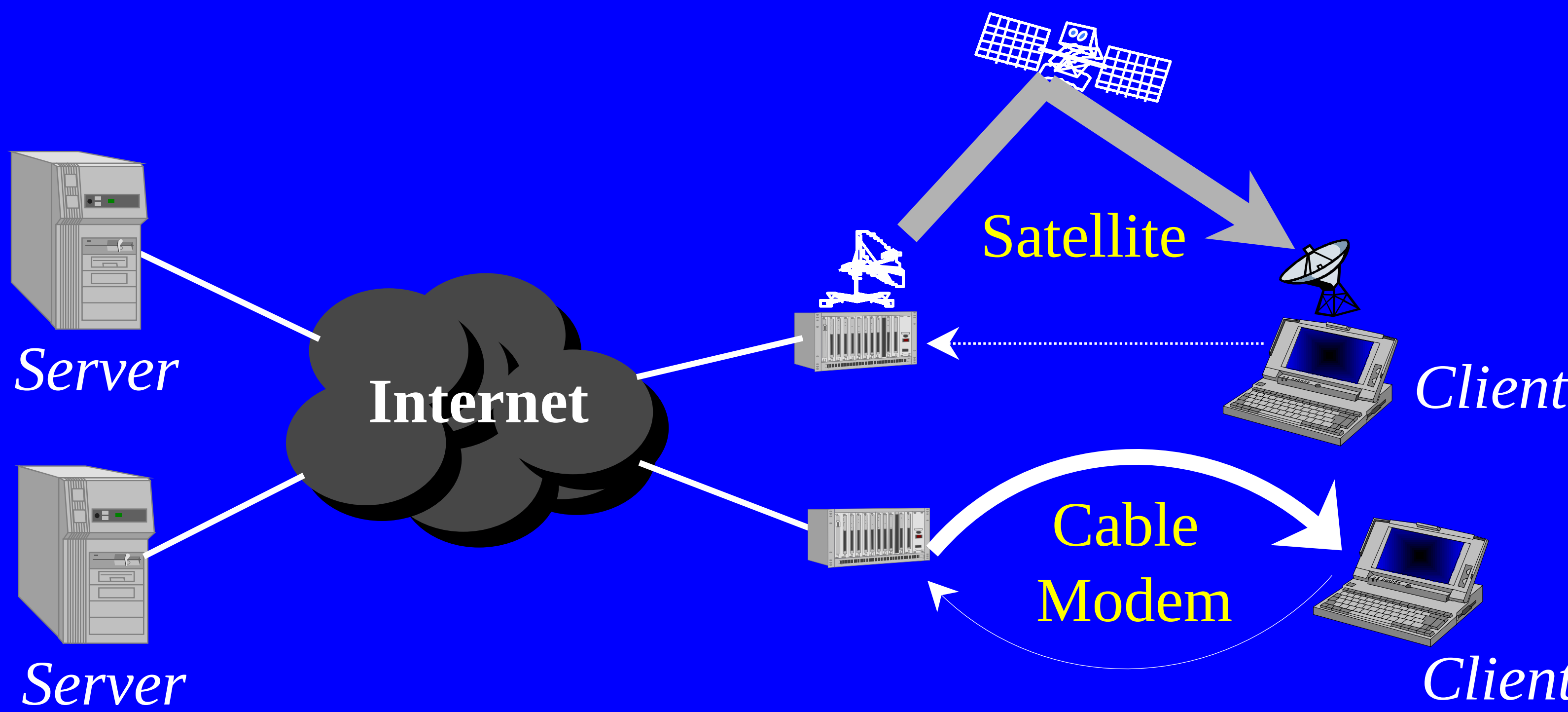
UW Whistler Retreat

December 1998

Outline

- Challenges
- Solutions
  - TCP Session
  - Fast Start
- Ongoing and Future Work

The Big Picture



Goal: Transfer data from servers to clients efficiently

Why is this hard?

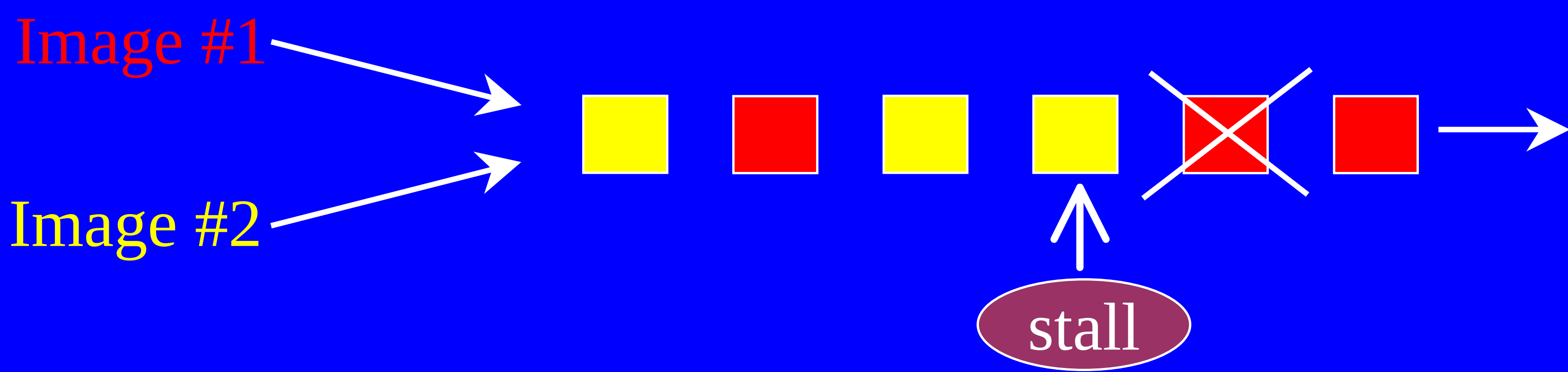
#1: Multiple independent components

#2: Bursty data transfers

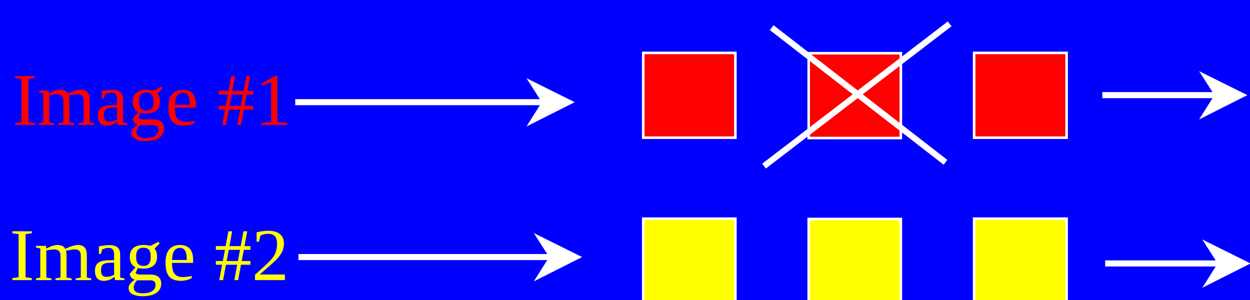
#3: Access network characteristics

#1: Multiple Independent Components

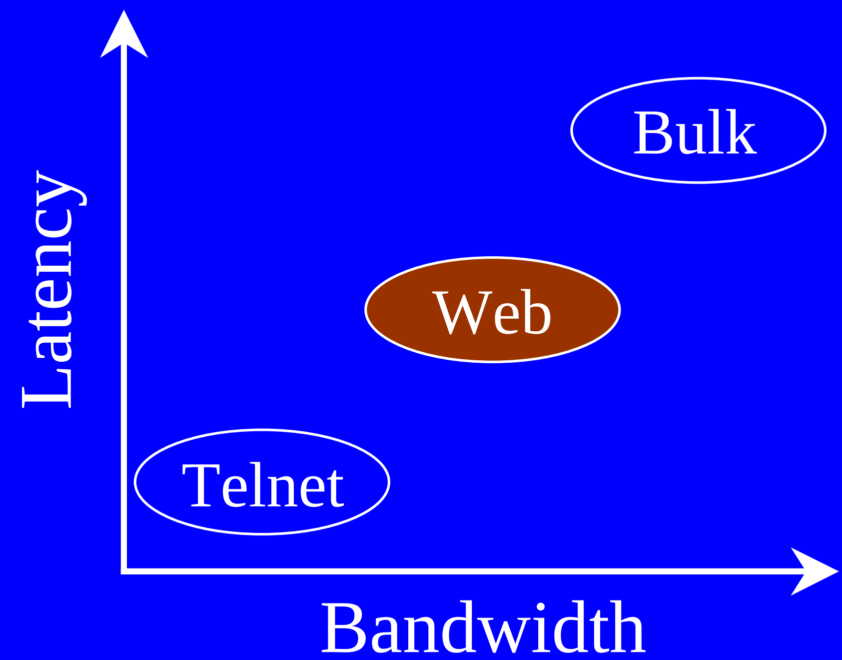
Interleaved data stream ⇒ undesirable coupling



Concurrent data streams ⇒ competition



#2: Bursty Data Transfers



- Download time sensitive to latency & bandwidth
- Shared network $\Rightarrow$ need to probe before use

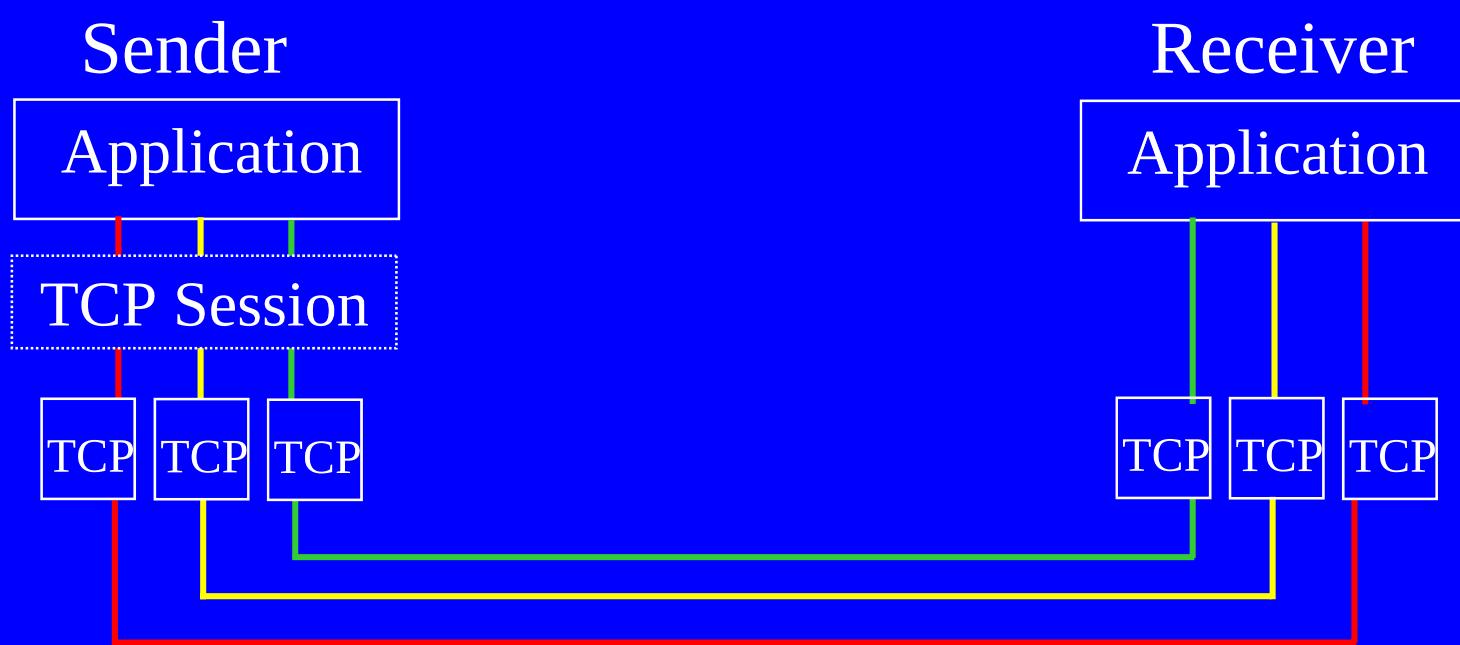
Probing for bandwidth requires time

#1: How to avoid competition and coupling?

- HTTP/1.0
  - avoids coupling but not competition
- P-HTTP [PM94]
  - avoids competition but not coupling
- TCP Control Block Interdependence [T97]
  - avoids coupling
  - avoids competition at the time of initialization but not beyond

TCP Session

Decouple service model from transport algorithms



Sender-side changes ⇒ easy to deploy incrementally

TCP Session

TCP session components

- Integrated congestion control
- Connection scheduling
- Integrated loss recovery

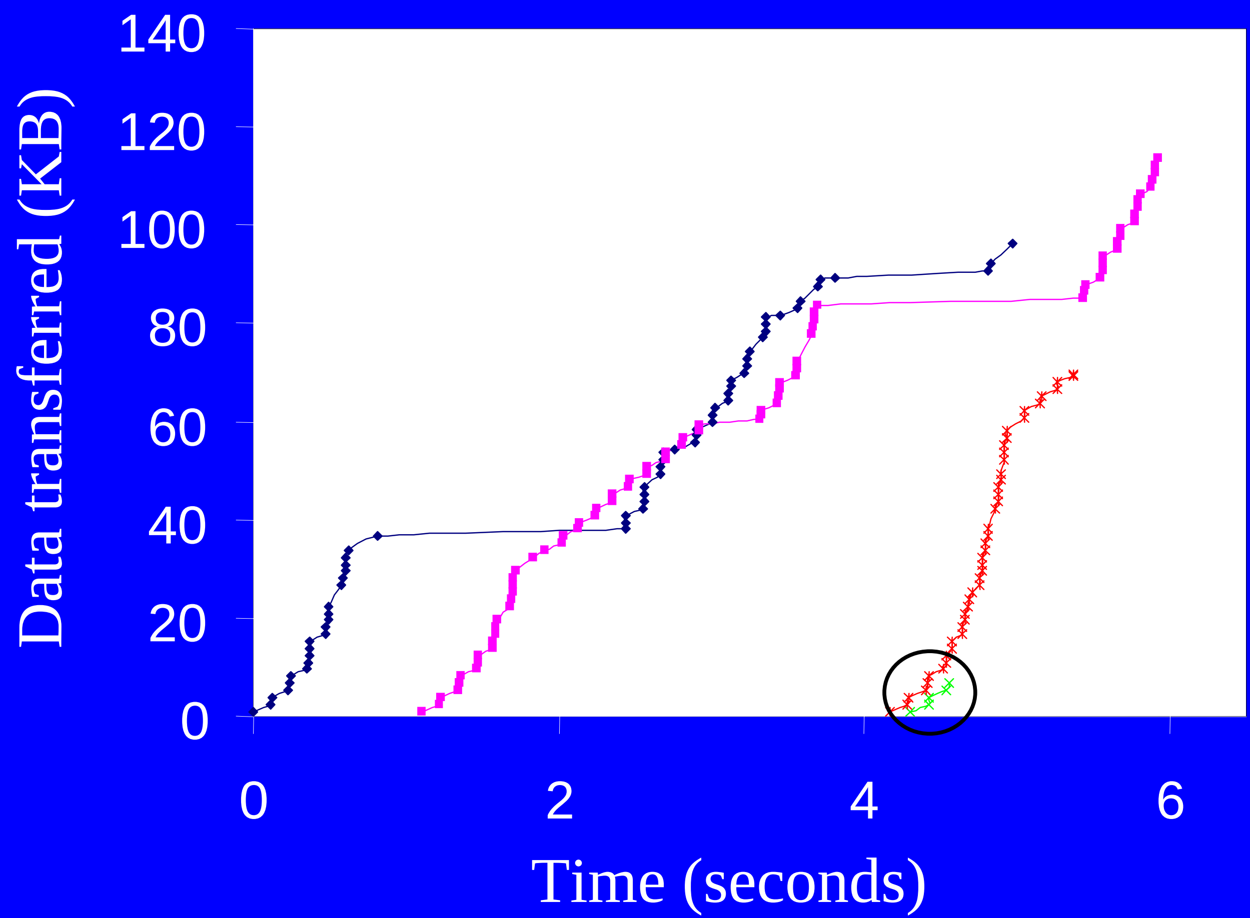
Flexible granularity of integration
(default: host-pair)

Congestion Control and Scheduling

Key idea: *how much* data, not what data

- Unified congestion window controls amount of session-wide outstanding data
- Window growth and shrinkage not tied to the number of connections
- Decouple connection scheduling from congestion control

Competing TCP Connections

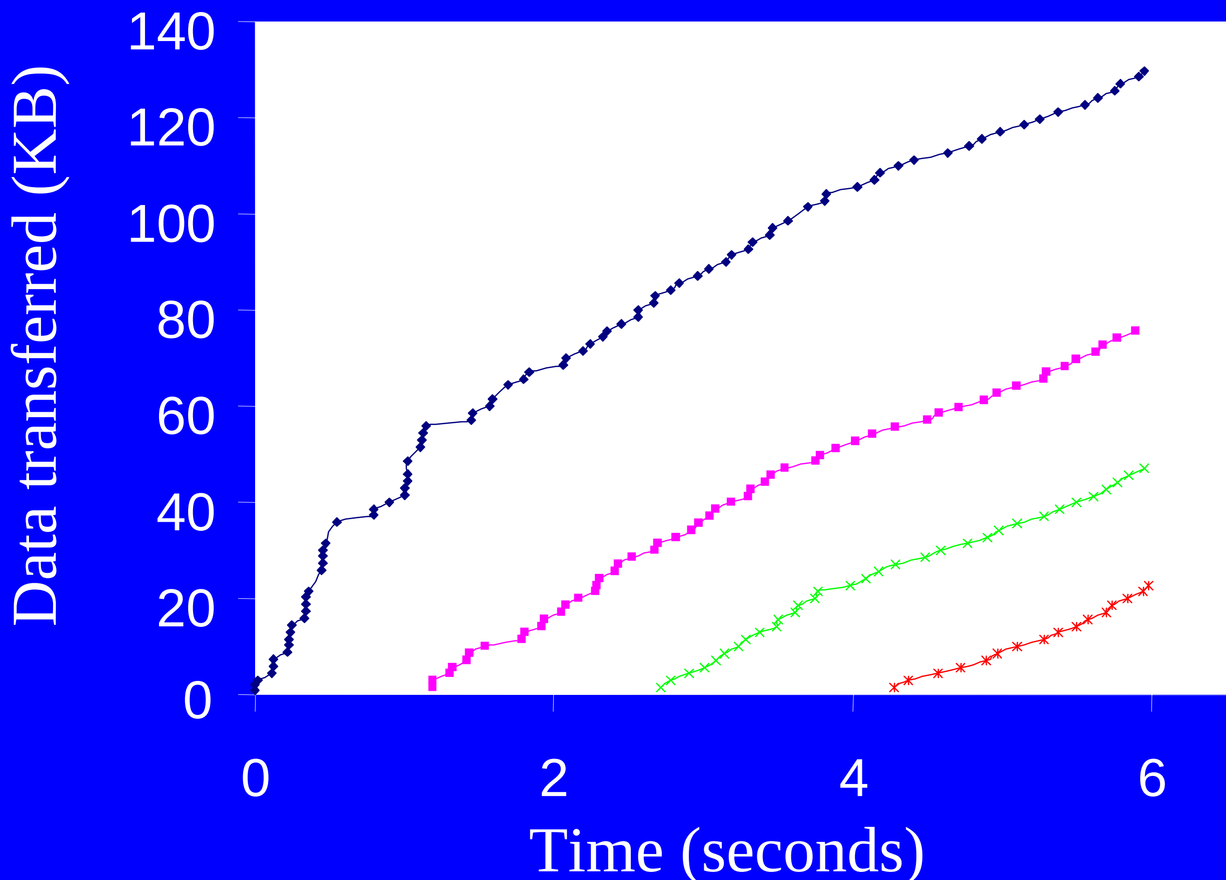


4 concurrent connections

1.5 Mbps/50 ms emulated link

Competition leads to inconsistent performance

Sharing with TCP Session



BSD/OS implementation

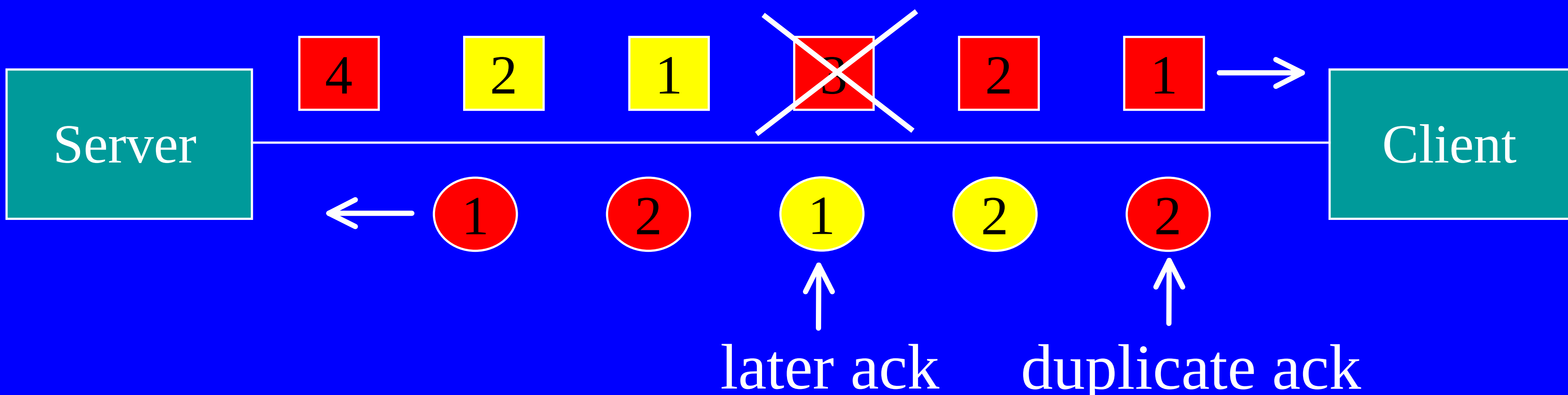
4 concurrent connections

1.5 Mbps/50 ms emulated link

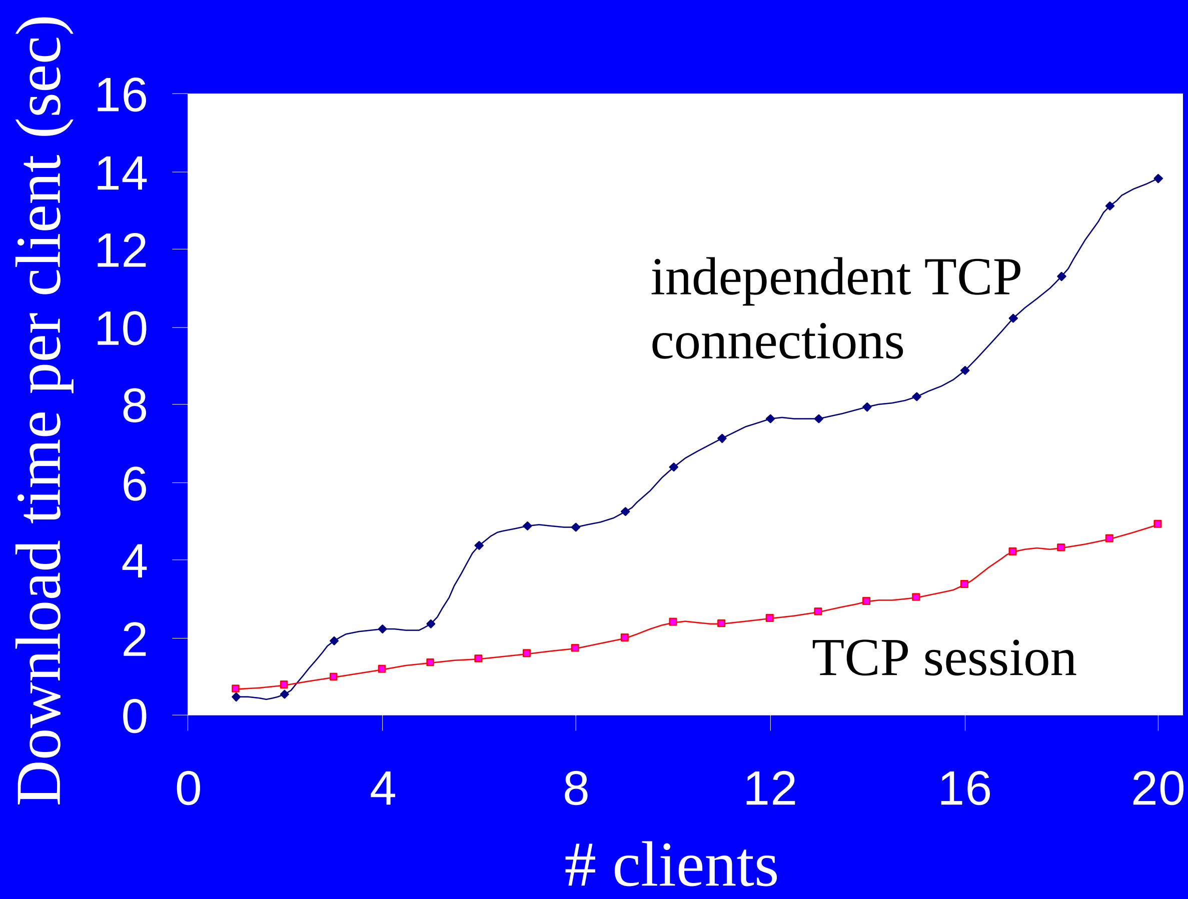
Sharing leads to more consistent performance

Integrated Loss Recovery

Key idea: use packet ordering information *across* connections to improve data-driven loss recovery



Performance

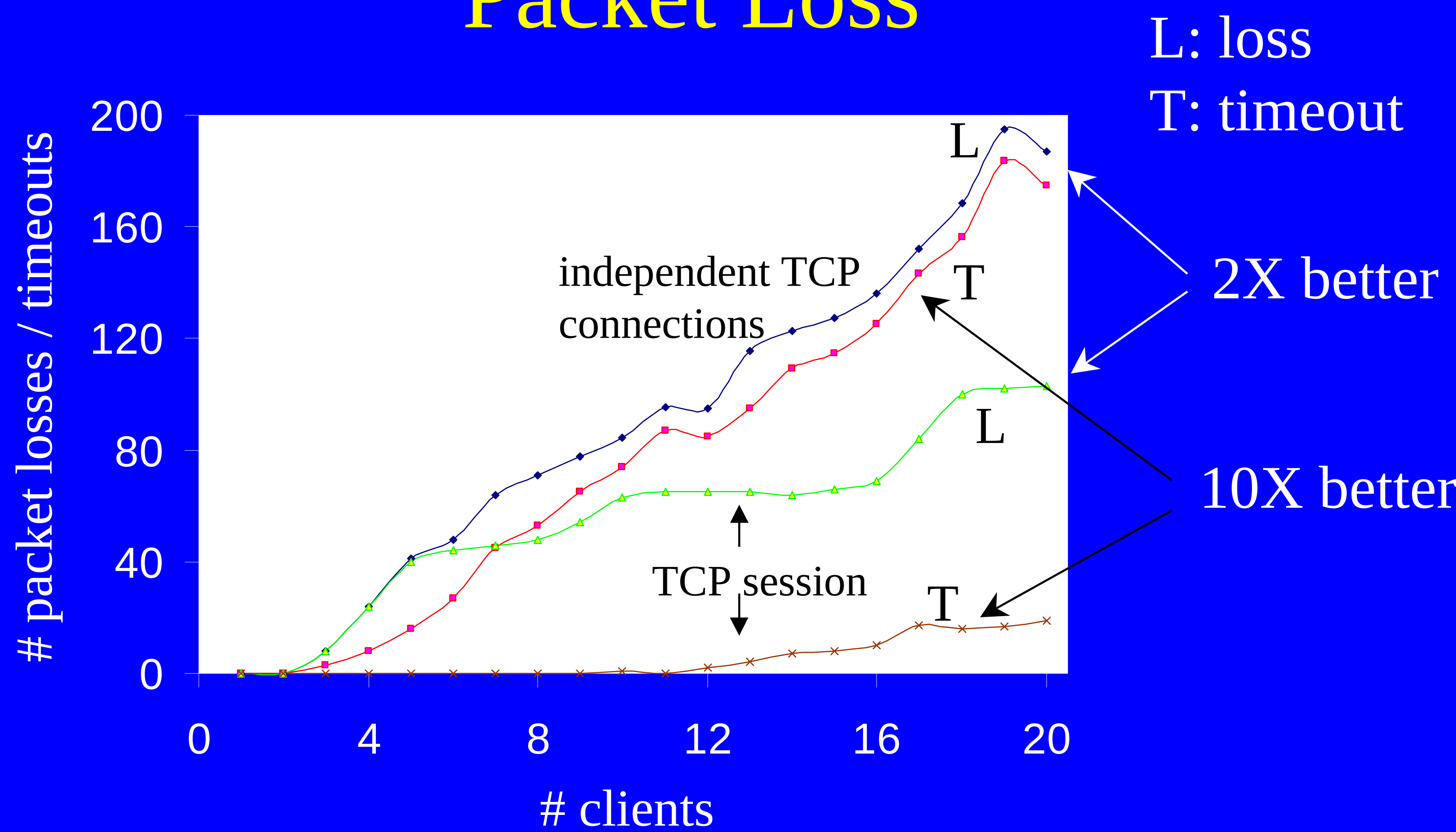


Server and clients connected via 1.5 Mbps/50ms link

4 concurrent 10 KB transfers between server and each client

2-3X reduction in download time

Packet Loss



2X reduction in packet losses due to integrated congestion ctrl.
10X reduction in timeouts due to integrated loss recovery.

Summary of TCP Session

Key idea: *separation of TCP functionality*

Advantages over independent TCP connections

- Fewer packet losses
- Better loss recovery
- More control over scheduling of data streams

Advantages over P-HTTP

- No coupling between concurrent data streams
- Not tied to specific application
- Changes confined to sender side

Bandwidth Probing in TCP

- *Slow-start* probing
  - exponential growth in congestion window starting with a size of one segment
  - *ack clocking* avoids burstiness
- *Linear* probing
- When is slow-start probing initiated?
  - upon connection start up
  - upon restart after an idle period
- How does it impact latency?
  - n-segment transfer $\Rightarrow$ at least $\log n$ RTTs

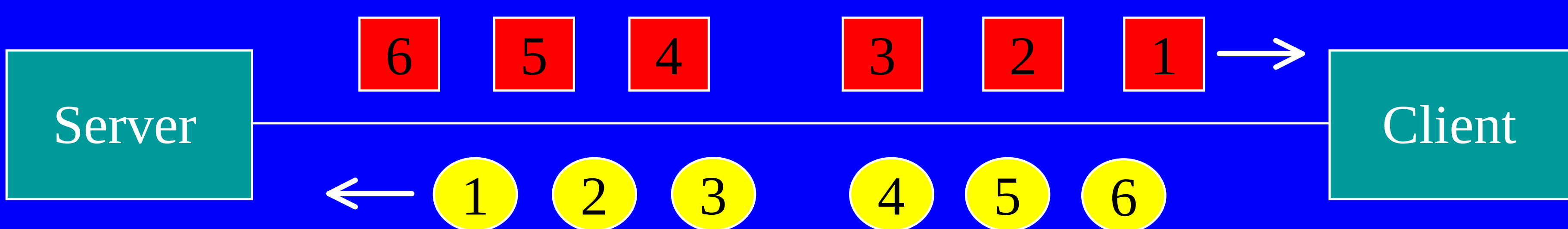
#2: How to reduce cost of probing?

- P-HTTP [PM94]
  - avoid repeated probing for components of a single Web page but not across pages
- 4K slow-start [AFP98]
- Rate-based Pacing [VH97]
  - smooth out using estimate of connection *rate*
  - but the estimate could itself be stale

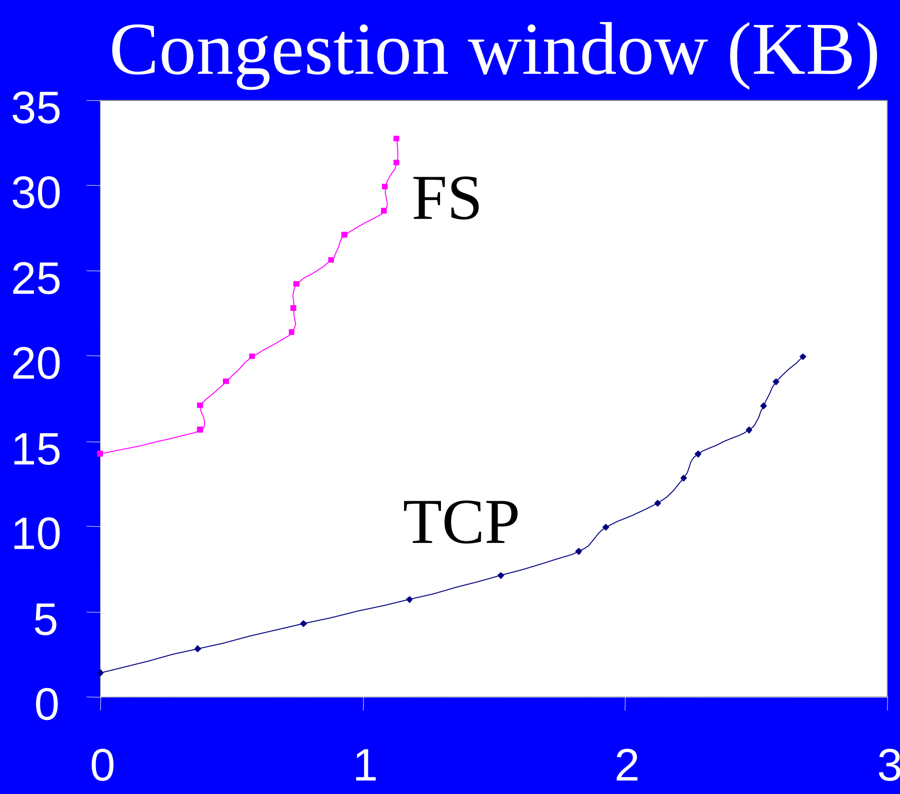
TCP Fast Start

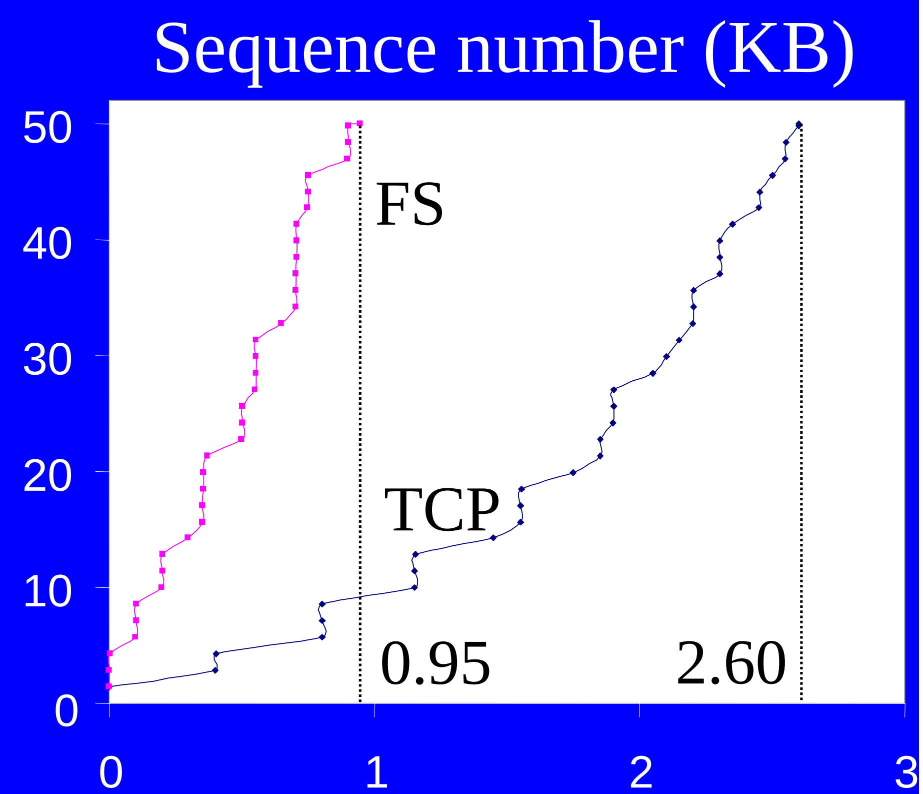
Basic idea: use cached network parameters to reduce the cost of probing

- Reuse most recent successful window size
  - slow-start ⇒ *oldcwnd/2*, linear phase ⇒ *oldcwnd-1*
- Estimate connection's rate as *cwnd/srtt*
- Break up large burst into *maxburst*-sized bursts



Dynamics of Fast Start





Data transfer over DirecPC satellite network

Robustness of Fast Start

Goal: Fast start should help when cached info is valid but *not* hurt when it is stale

Studies indicate that available bandwidth is often stable for several minutes [P97,BSSK97]

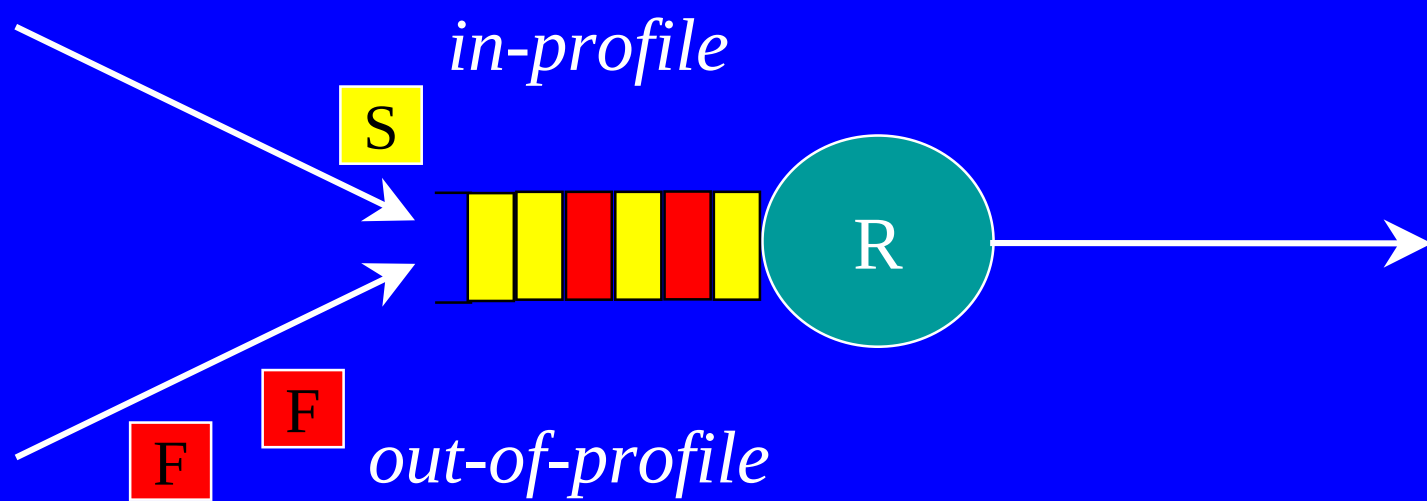
But we need to guard against *staleness*

- Protecting others
- Protecting oneself

Protecting Others

Protect others from over-aggressive fast start

Preferentially drop fast start packets (except first one)



- Enables control on time scale finer than RTT
- Avoids potential congestion collapse

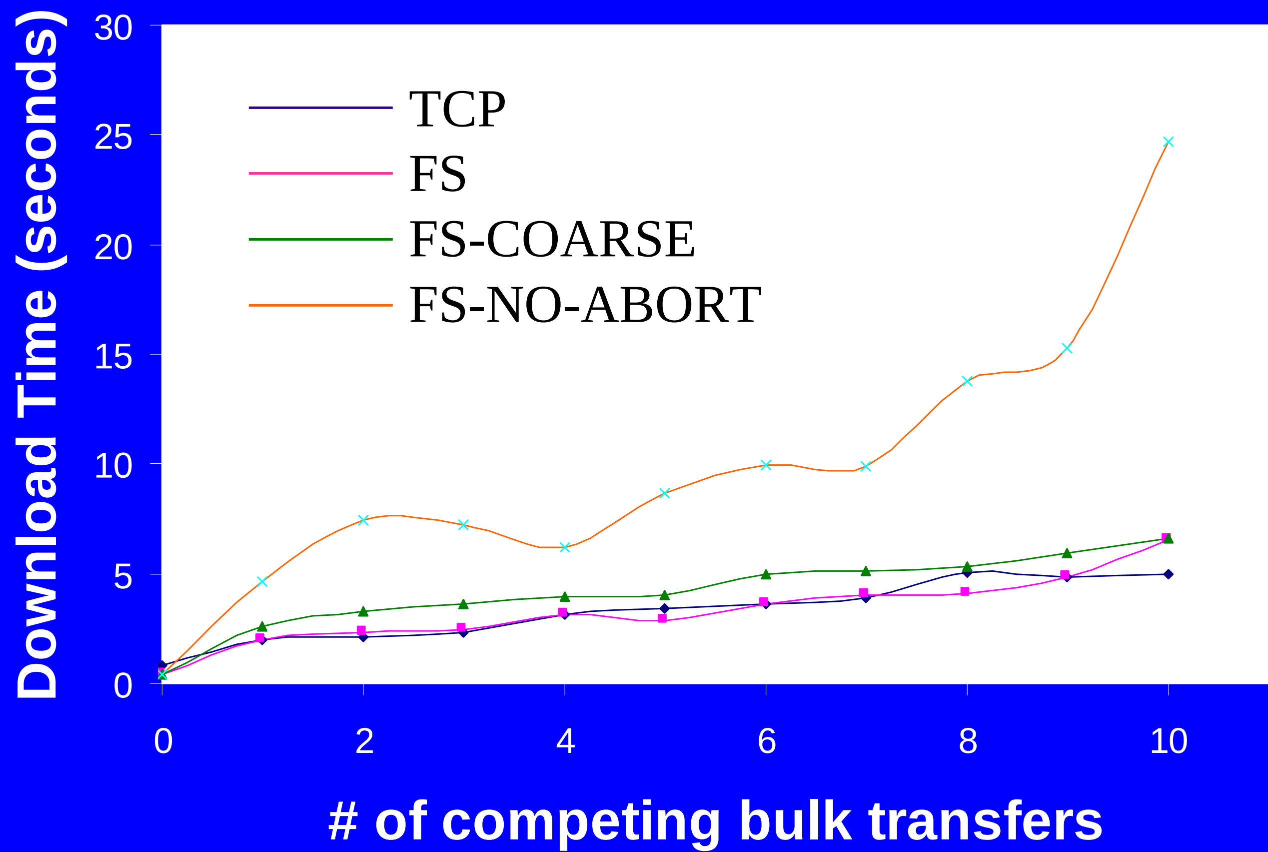
Protecting Oneself

Protect oneself from consequences of burst loss

Quickly detect and abort failed fast start attempt

- Fine-grained *reset* timer during fast start phase
  - tied to the fast TCP timer (200 ms)
- If reset timer expires, abort fast start
  - reset *cwnd to* one segment, initiate slow start
  - no other congestion control penalties
    - *ssthresh* not halved, RTO not backed off
- Abort also when multiple losses within RTT

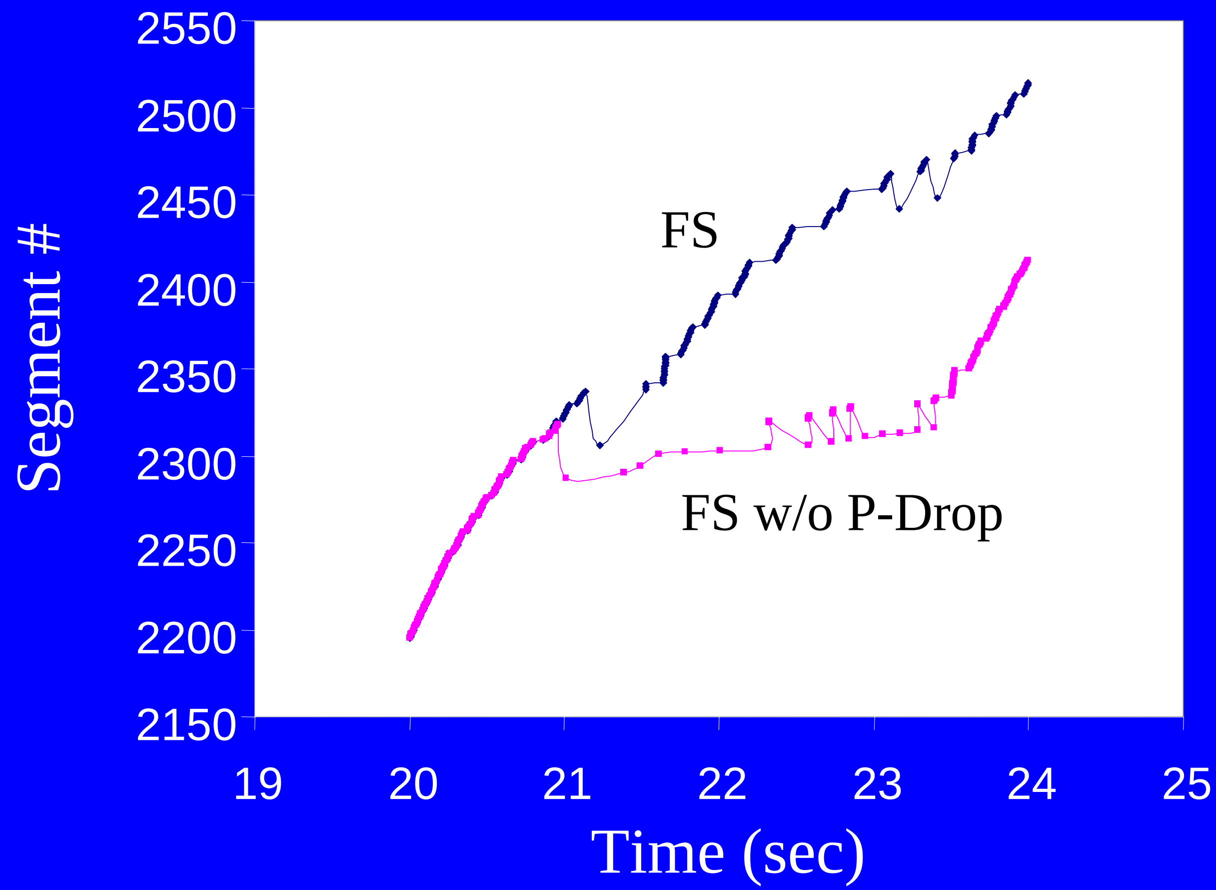
Impact of Staleness on Oneself



1.5 Mbps
50 ms

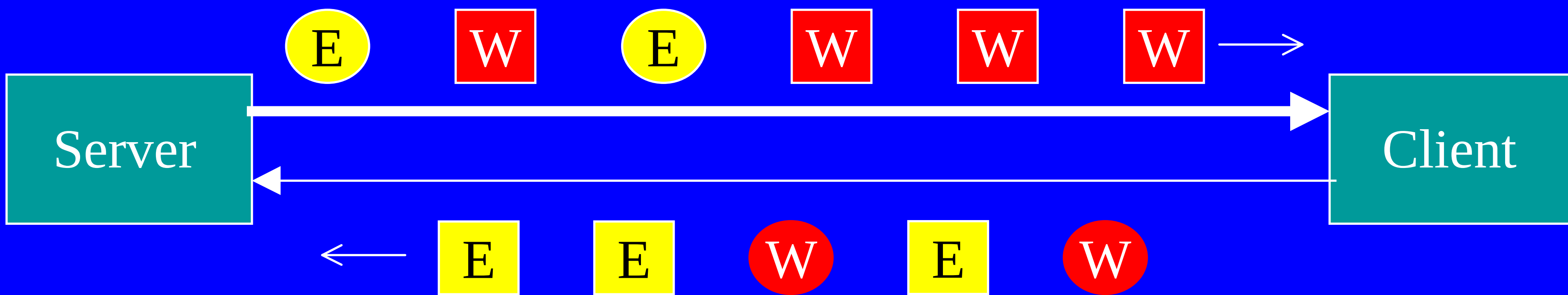
Aborting fast start in case of failure prevents significant performance degradation

Impact of Staleness on Others



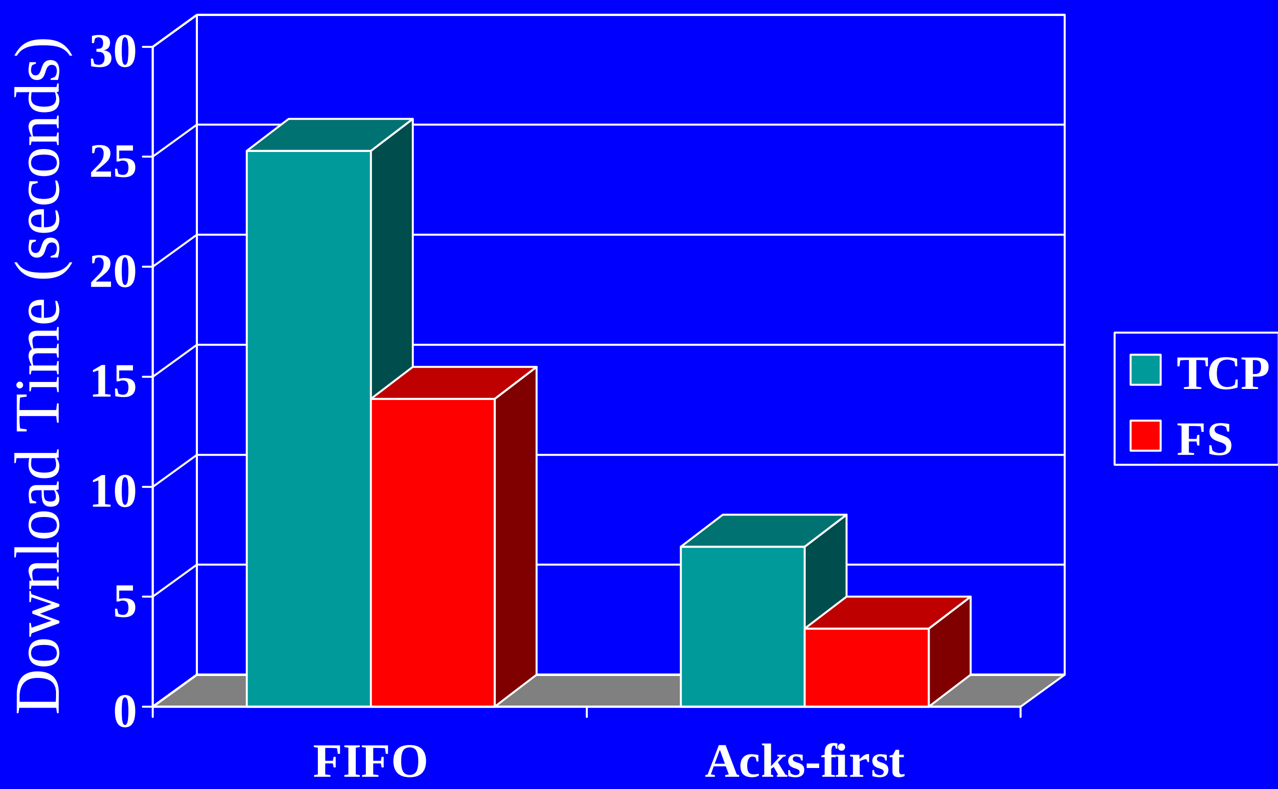
Priority dropping significantly decreases adverse impact on competing traffic

Asymmetric Access Network



- Problem: upstream data packets block acks
  - RTT can become very large
- Possible solution: *acks-first* scheduling [BPK97]
  - but RTT can still be large due to the packet in transmission

Impact of Bidirectional Traffic



175 KB page download over 10 Mbps/28.8 Kbps network

Fast start helps even though the inherent RTT is not large

Summary and Conclusions

- TCP Session
  - decouples service model from transport algorithms
  - enables concurrency without competition
- Fast Start
  - exploits differentiated services to complement end-to-end control with faster time-scale control
  - improves bandwidth utilization in the common case
  - avoids risk of performance degradation in the worst case

Conclusions

- Fast start is robust
  - significant benefit (2X) in favorable conditions
  - little performance degradation in adverse conditions
    - priority dropping, quick detection of failed fast start
- Reduced latency helps both clients and servers
  - client: faster downloads
  - server: resources freed up more quickly
- Significant benefit with new access networks
  - satellite, cable modem
  - provides path for incremental deployment