TCP Fast Start: A Technique For Speeding up Web Transfers

Venkata N. Padmanabhan

Microsoft Research

Randy H. Katz

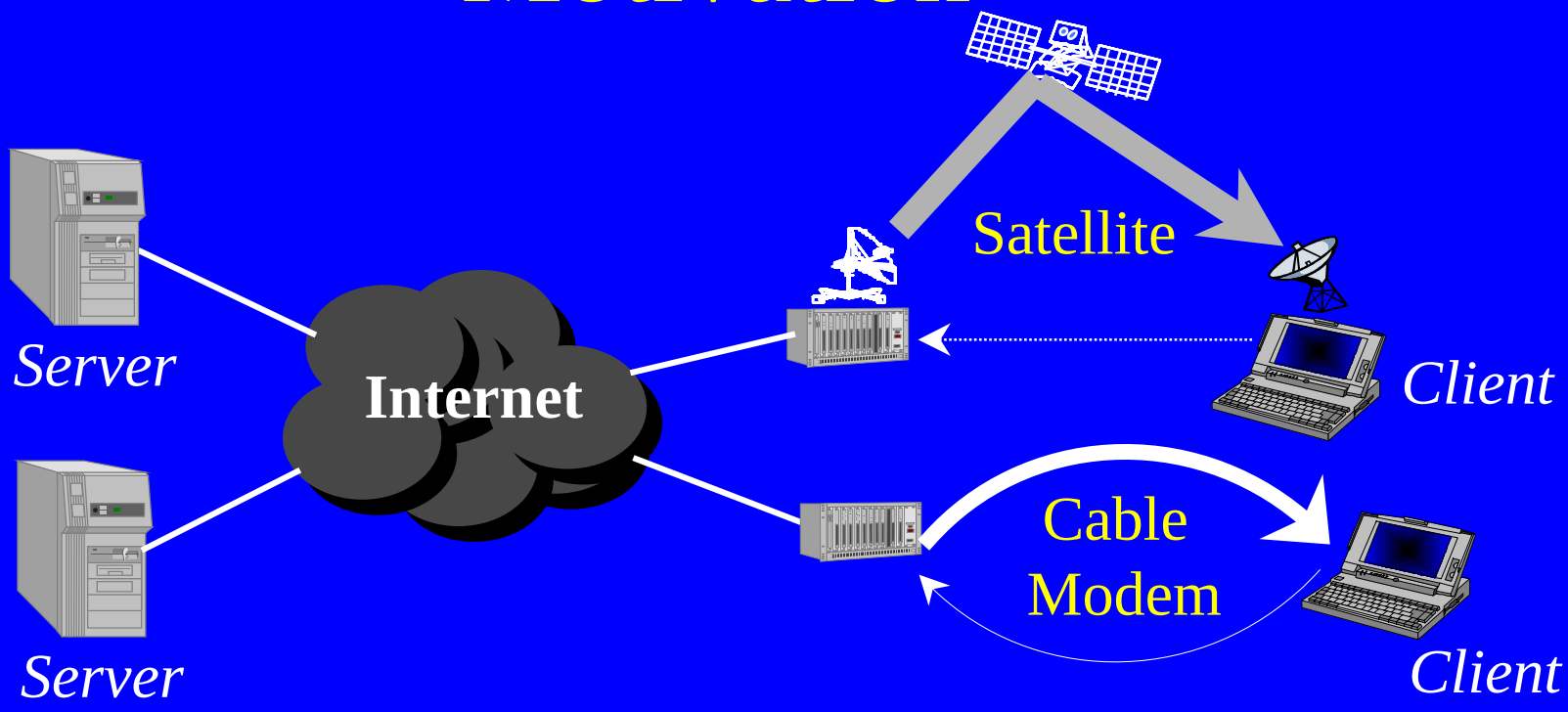
University of California at Berkeley

IEEE Global Internet Conference, 1998

Outline

- Motivation
- Related Work
- TCP Fast Start Basics
- Robustness
- Asymmetric Access Networks
- Conclusions and Future Work

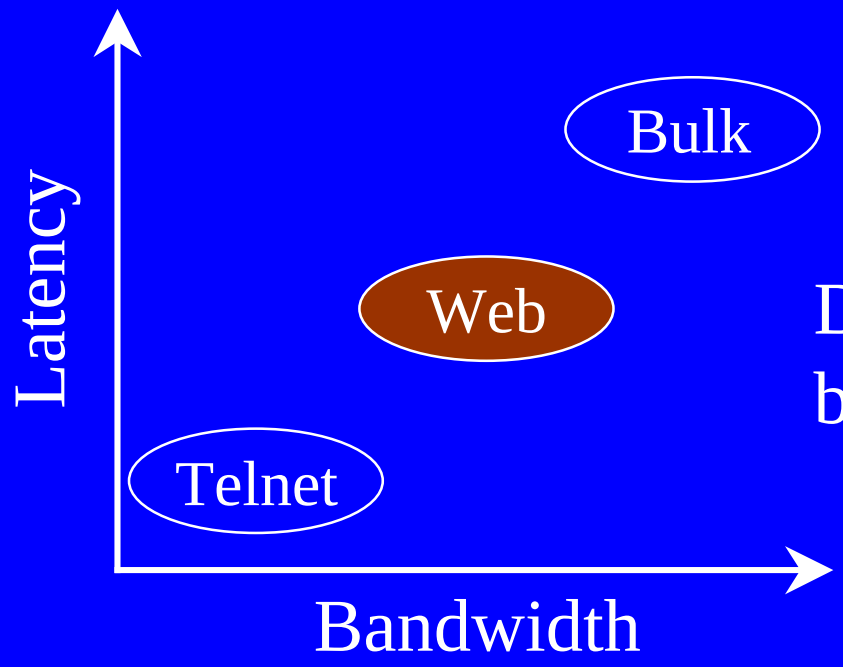
Motivation



- Dominance of Web browsing
- New access network technologies

A small download time is important.

Problem: Bursty Transfers



Download time is sensitive to both latency and bandwidth.

Internet is a shared, reservation-less network ⇒ need to probe for bandwidth before using it

Probing for bandwidth requires time

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 ⇒ at least *log n* RTTs

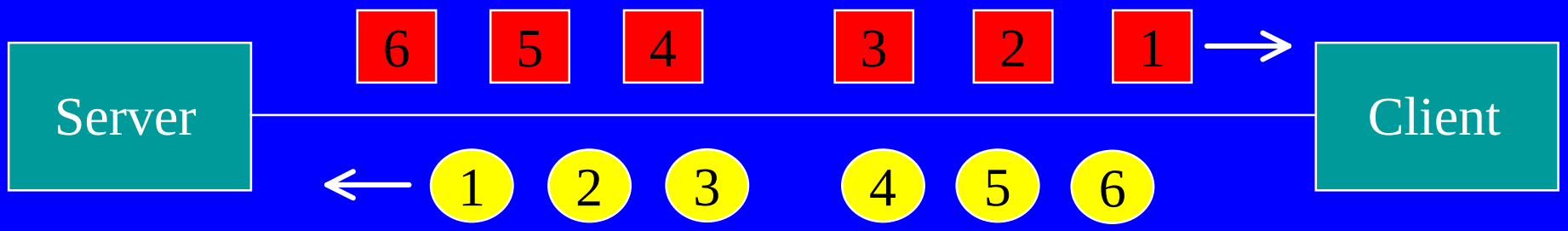
How to reduce cost of probing?

- P-HTTP [PM94]
  - avoid repeated probing for components of a single Web page but not across pages
- T/TCP [B94]
  - cache connection count, RTT
- TCP Control Block Interdependence [T97]
  - cache *cwnd*, but large burst could cause losses
- Rate-based Pacing [VH97]
  - smooth out using *rate*, but that could itself be stale
- 4K slow-start [AFP98]

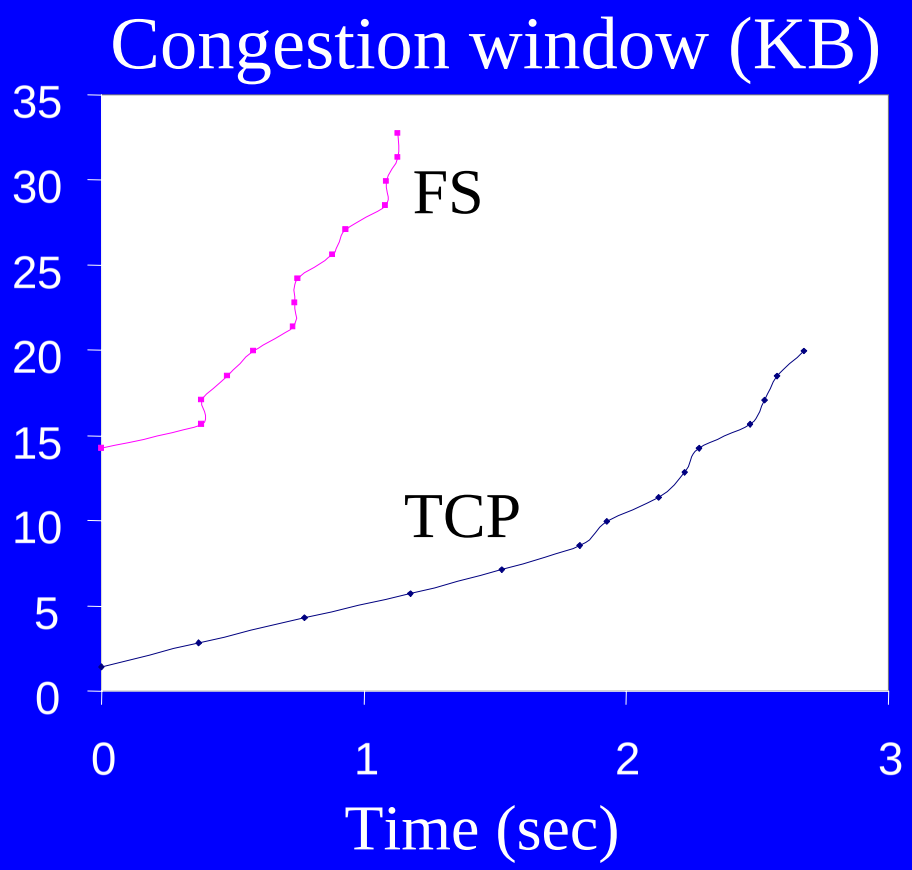
TCP Fast Start

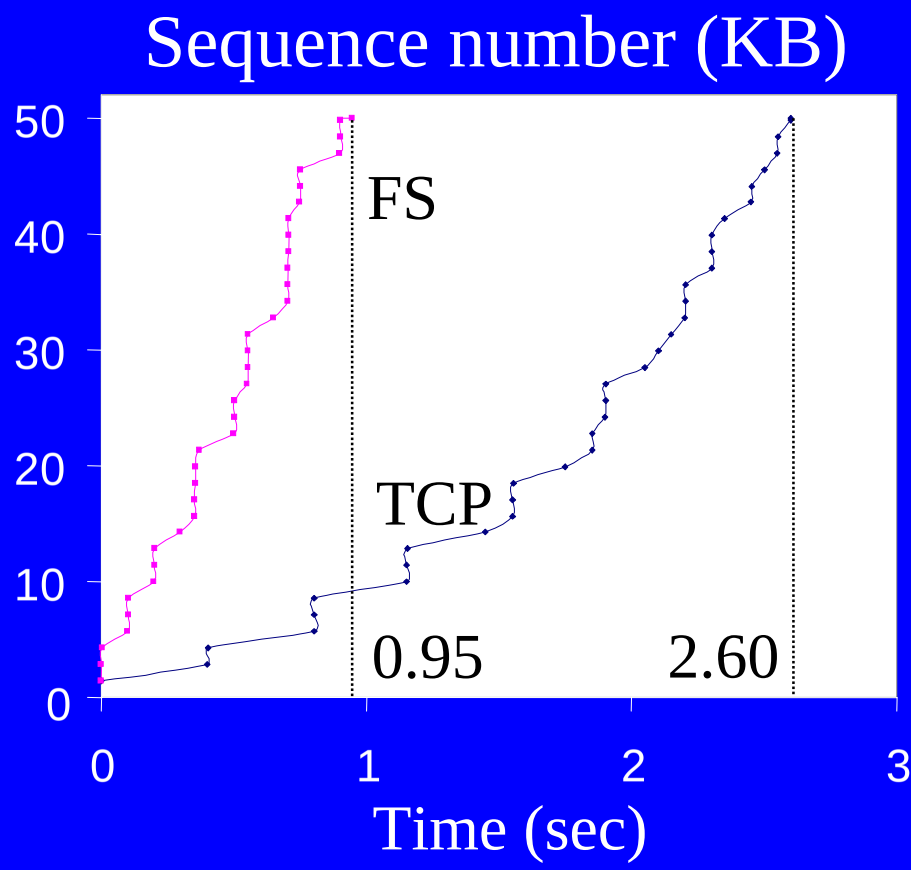
Basic idea: use cached network parameters to reduce the cost of probing

- 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



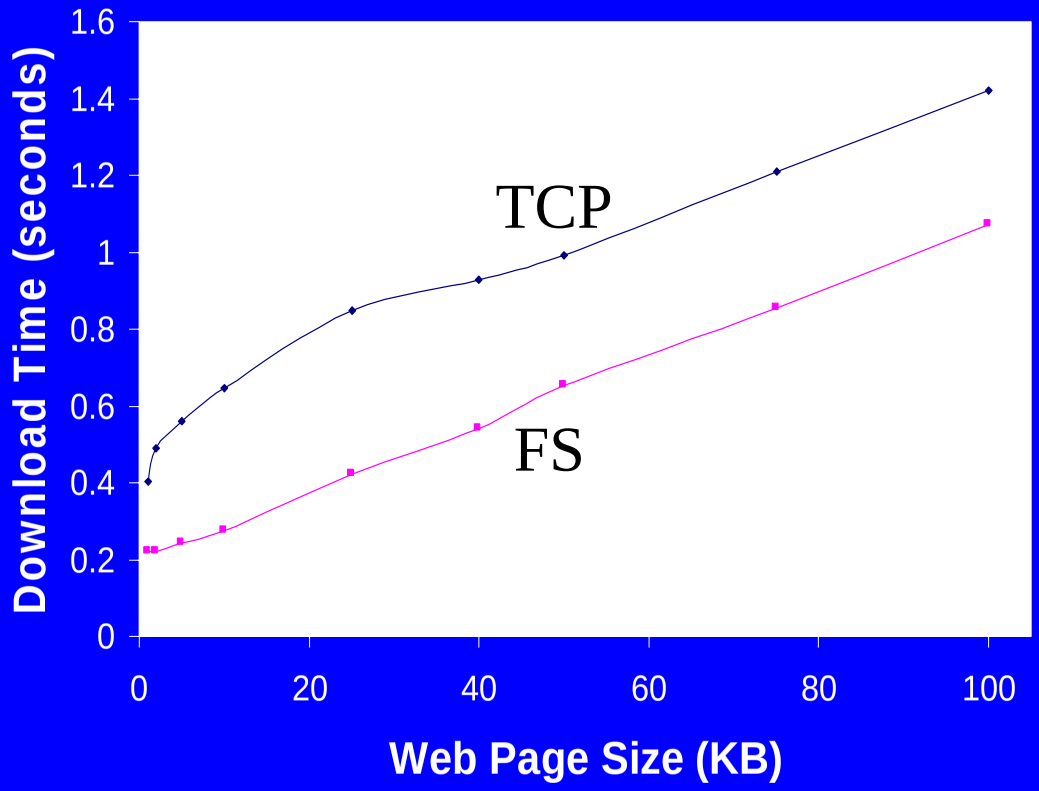
Dynamics of TCP Fast Start





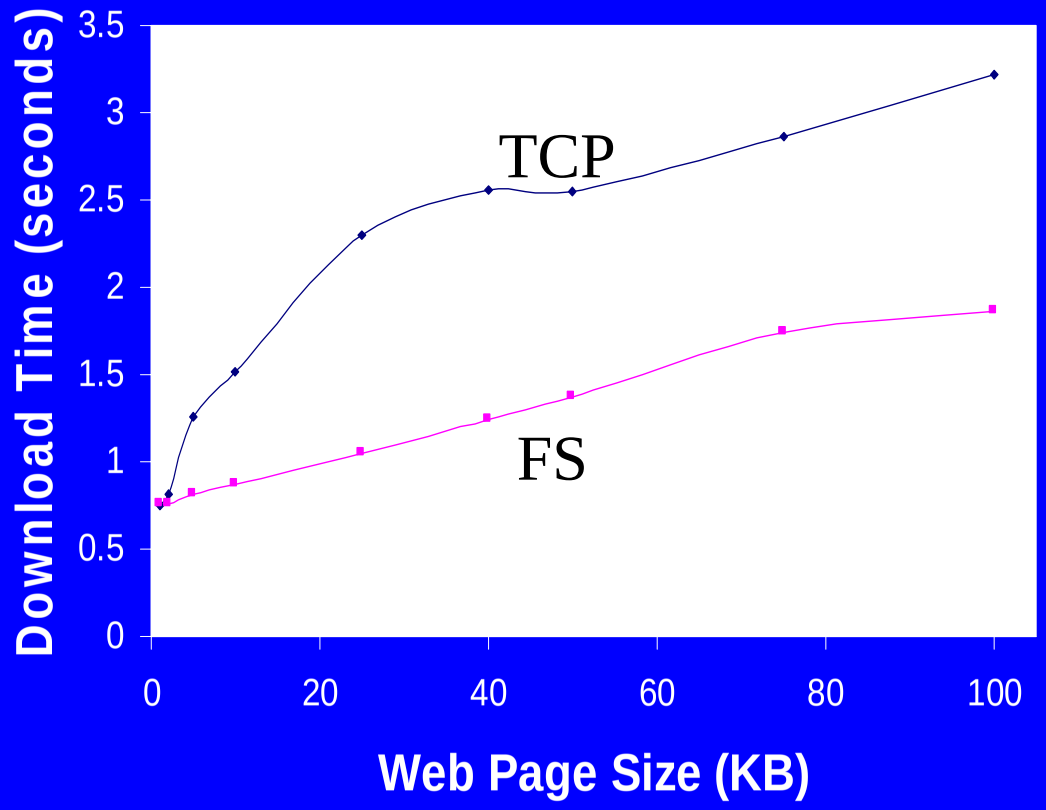
BSD/OS implementation operating over DirecPC satellite network

Terrestrial WAN



- 70 ms RTT
- 40% (1.67X) improvement for 40 KB Web page

DirecPC Satellite Network



- 335 ms RTT
- 50% (2X) improvement for 40 KB Web page

Robustness

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]

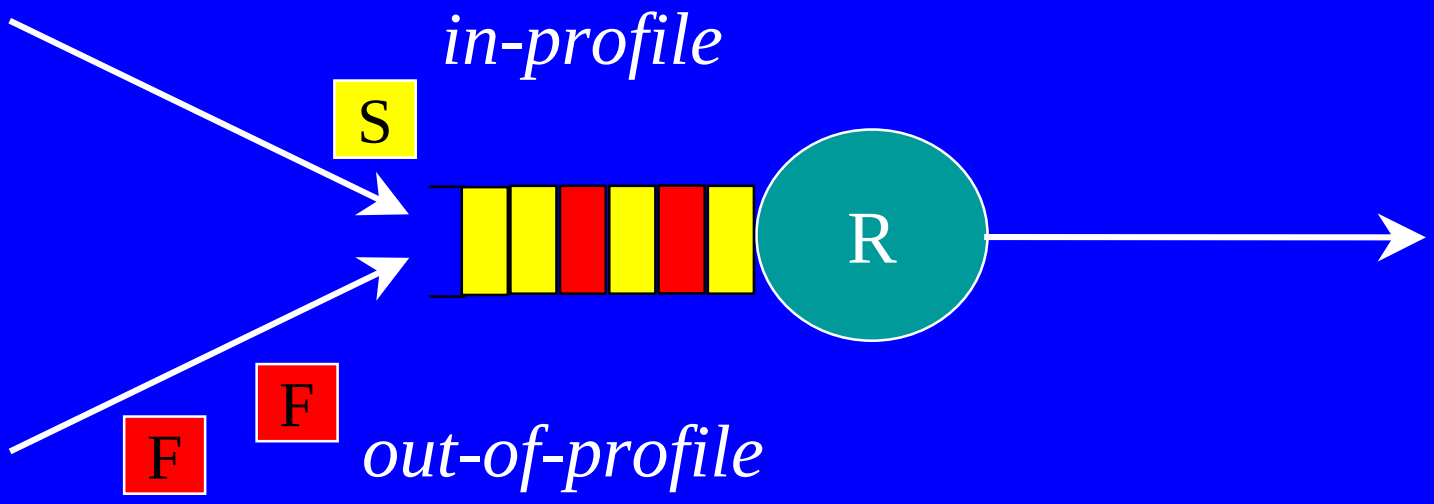
But we need to guard against *staleness*

- 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

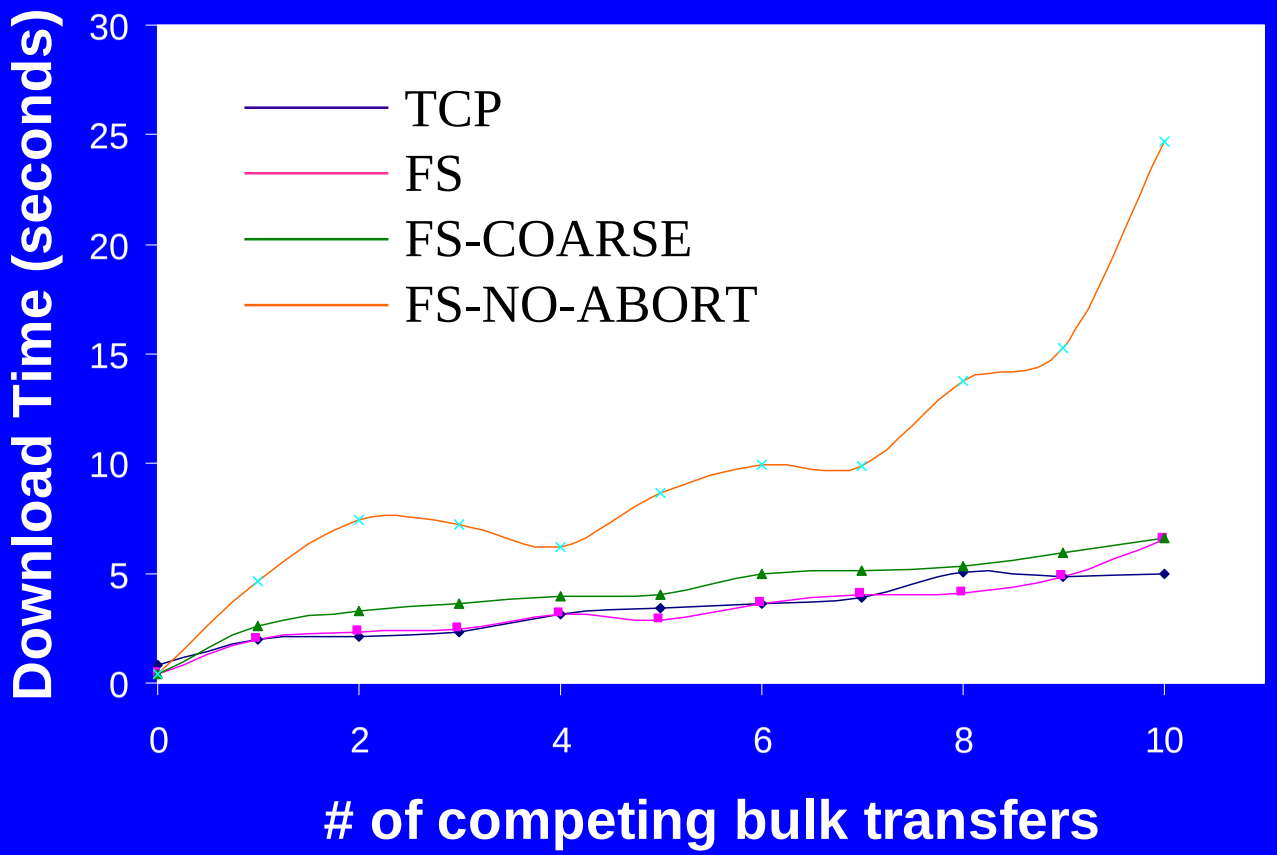
Protecting Oneself

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

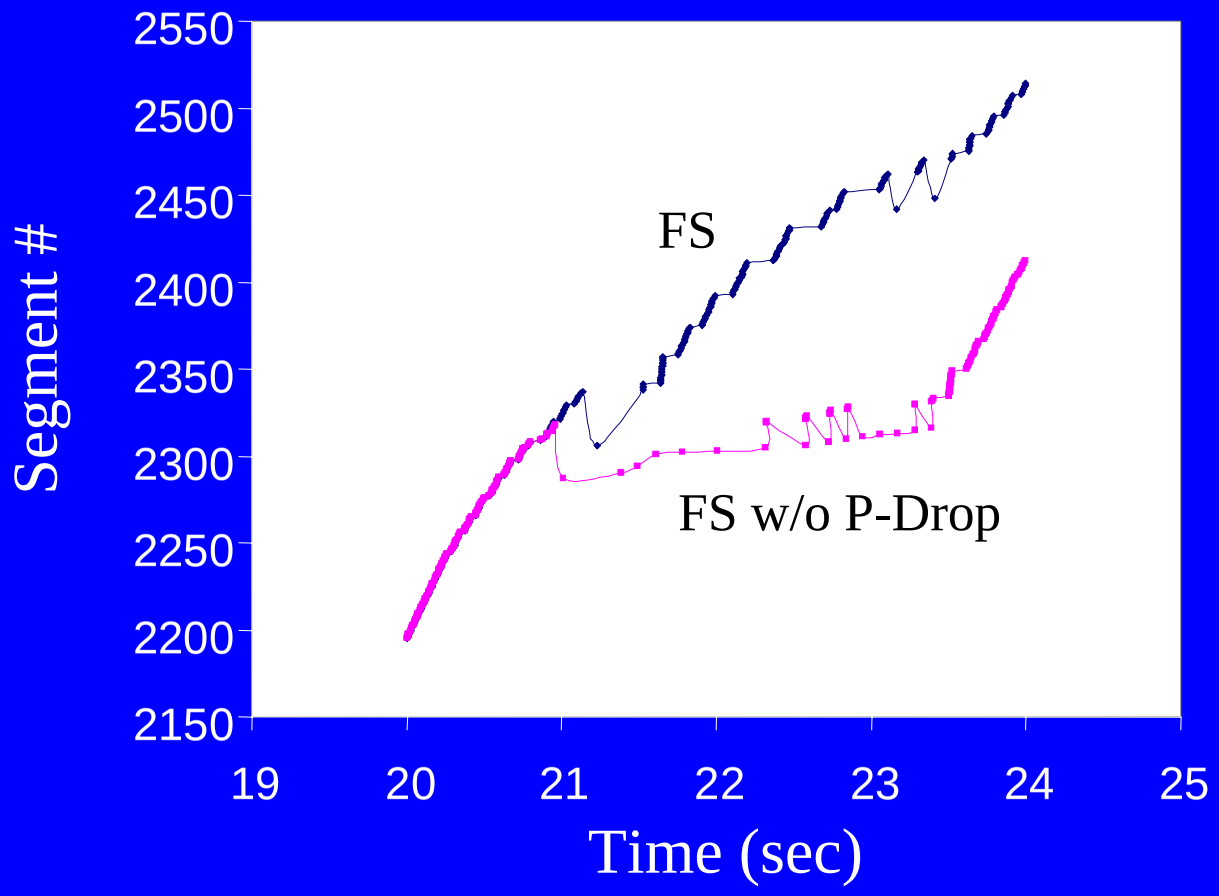
Impact of Staleness on Oneself



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

Other Results

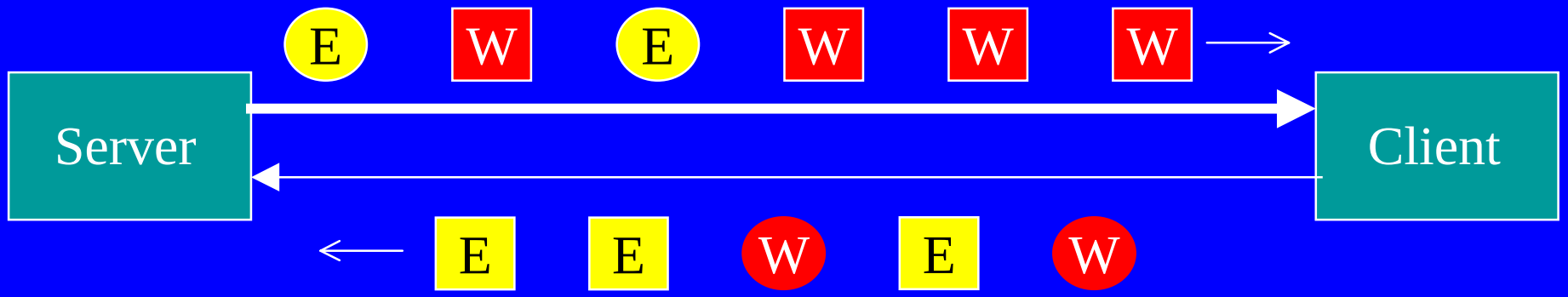
- RED helps because router is better able to absorb bursts
- Fast start makes little difference in dialup networks
- Fast start can help significantly in asymmetric access networks

Asymmetric Access Networks

- High-bandwidth downlink, low-bandwidth uplink
- Cost-effective provision of bandwidth in the direction of interest
- Example: unidirectional cable with dialup return

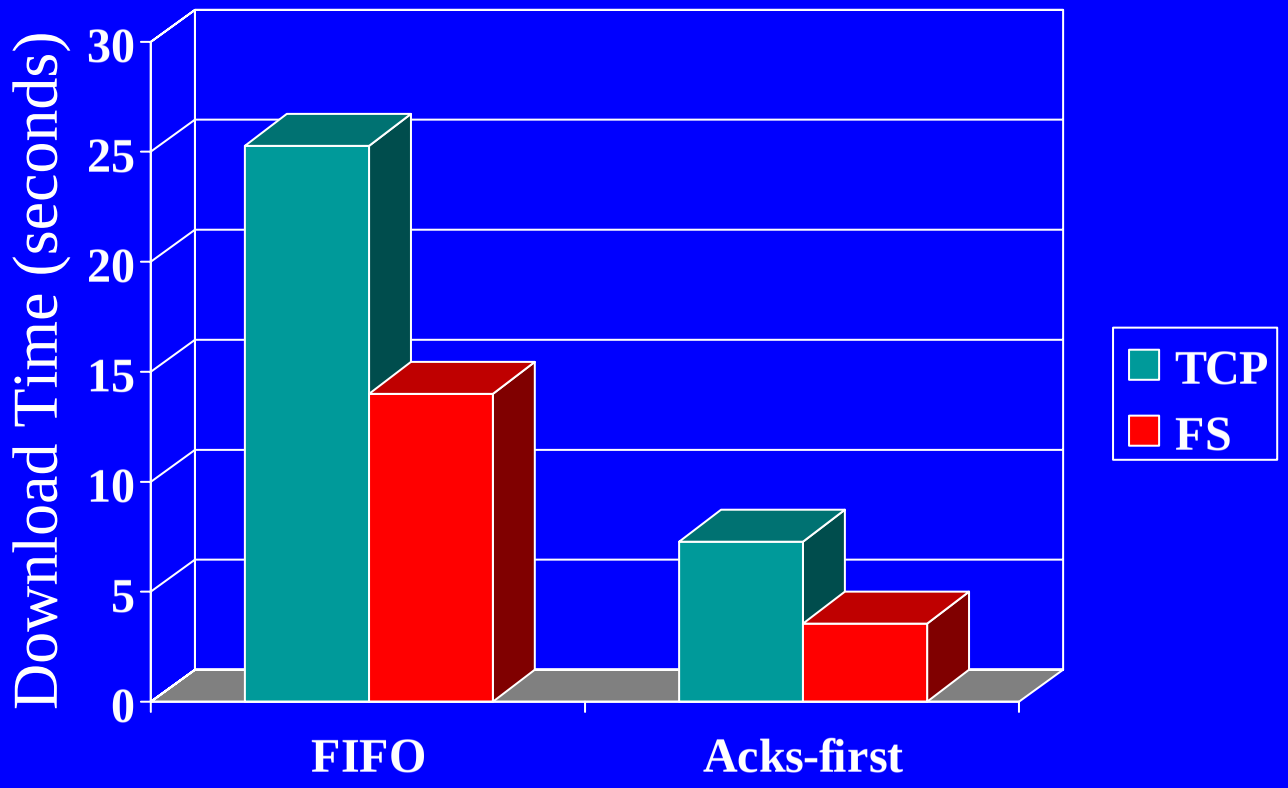
Bidirectional traffic can disrupt ack clocking and so impact downstream throughput adversely

Impact of Bidirectional Traffic



- 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

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