The Effects of Asymmetry on TCP Performance

Hari Balakrishnan   Venkata N. Padmanabhan

Randy H. Katz

University of California at Berkeley

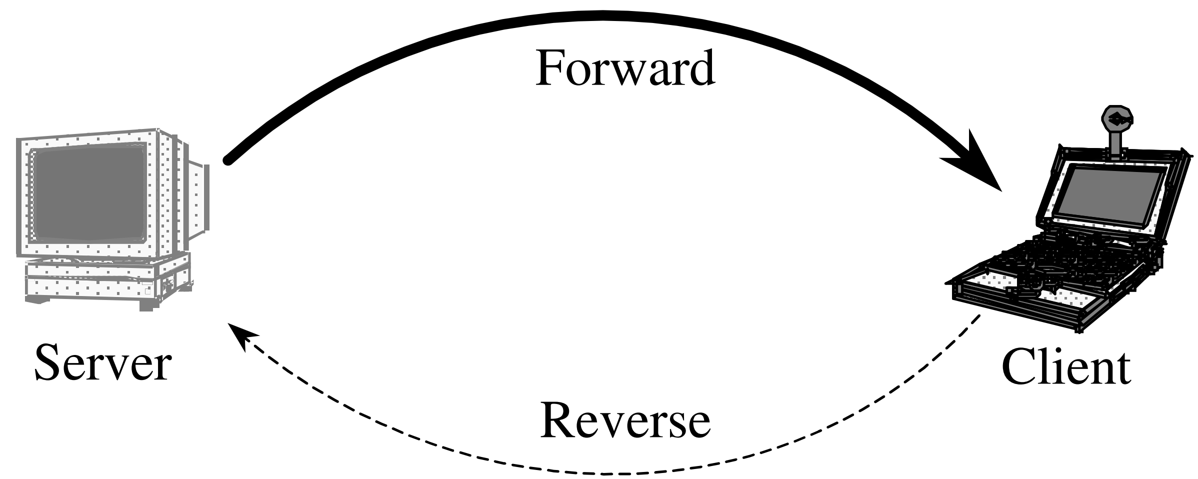
Daedalus/BARWAN Retreat

June 1997

Outline

- Overview
- Bandwidth asymmetry
  - experimental testbed
  - simulation results
- Media access/latency issues
  - experimental testbed
  - simulation results
- Summary
- Future work

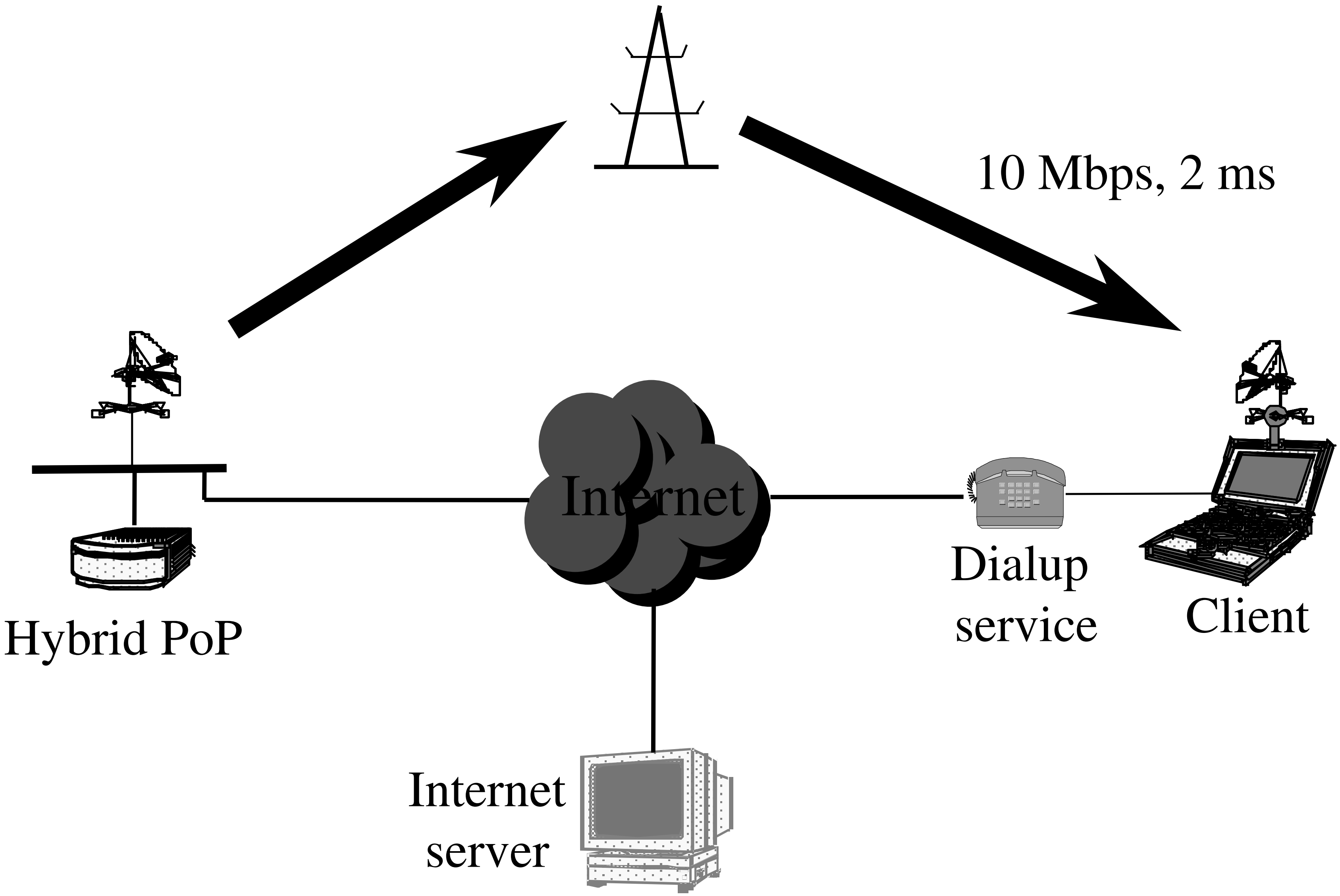
Aspects of Asymmetry



- *Bandwidth:* 10-1000 times more in forward direction
- *Latency:* asymmetric channel access and interfering traffic
- *Packet loss:* more losses in one direction

Goal: Analyze and evaluate how the network and traffic in one direction affect performance in the other

Hybrid Wireless Cable Network



Bandwidth Asymmetry

- Bandwidth-constrained reverse channel could limit data throughput in forward direction
  - contention for buffer space
  - packet scheduling issues
- Several factors determine performance
  - normalized bandwidth ratio
  - reverse channel buffer size
  - whether unidirectional or bidirectional traffic
- Solutions:
  - end-to-end and/or router-based

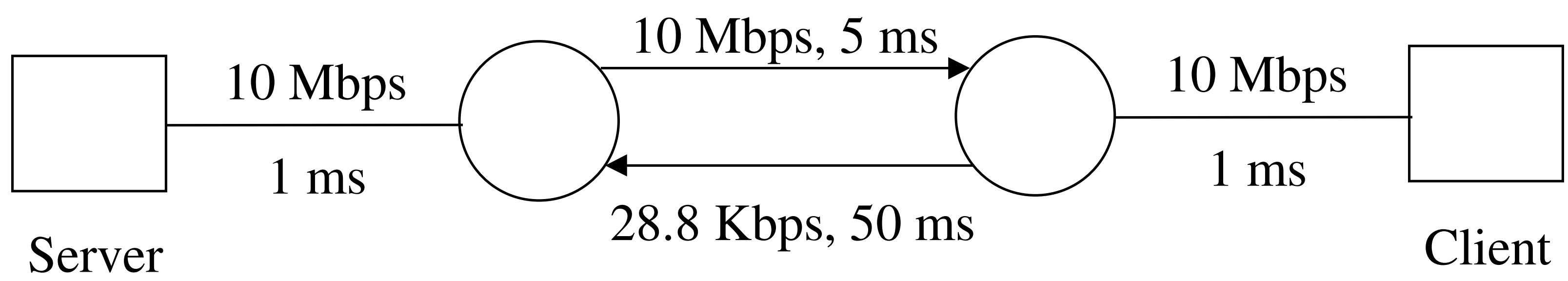
End-to-End Solutions

- At receiver: *ack congestion control*
  - extension of TCP delayed acks
  - frequency of acks is varied adaptively depending on level of congestion in the reverse channel
  - congestion feedback
    - from router (e.g., RED)
    - from sender
- At sender:
  - window growth tied to amount of data acked rather than the number of ack packets received
  - potentially large bursts broken up into smaller ones

Router-based Solutions

- *Ack filtering*
  - older acks removed in favor of more recent ones
    - in extreme case, all except most recent one removed
  - where to place the acks that remain?
- *Acks-first* scheduling
  - acks given higher priority than data packets

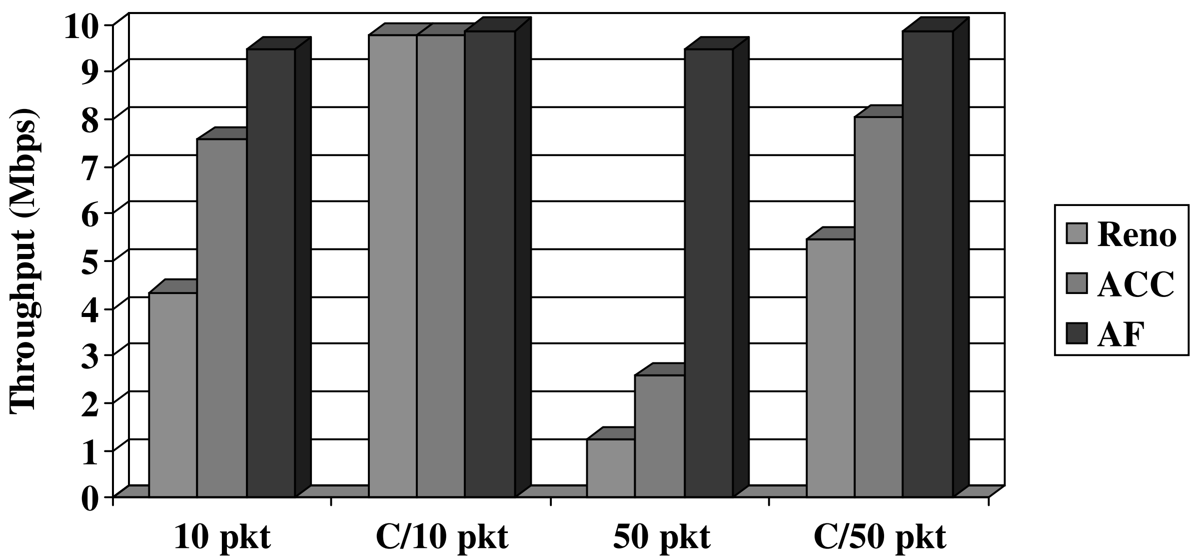
Simulation Model



– Used ns simulator with Daedalus enhancements
– Parameters chosen to model Hybrid system
– Metrics:
    - aggregate throughput in each direction
    - fairness index

Single One-Way Transfer

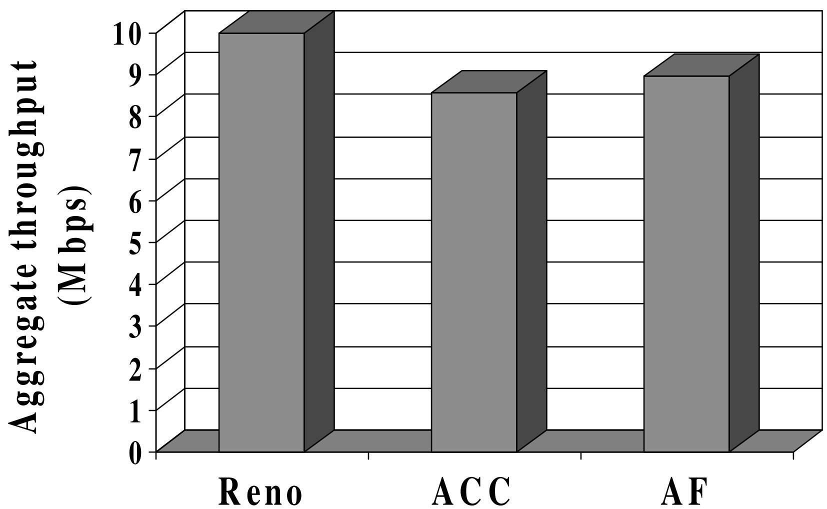
– Single TCP transfer in the forward direction
– Maximum window size set to 100 KB

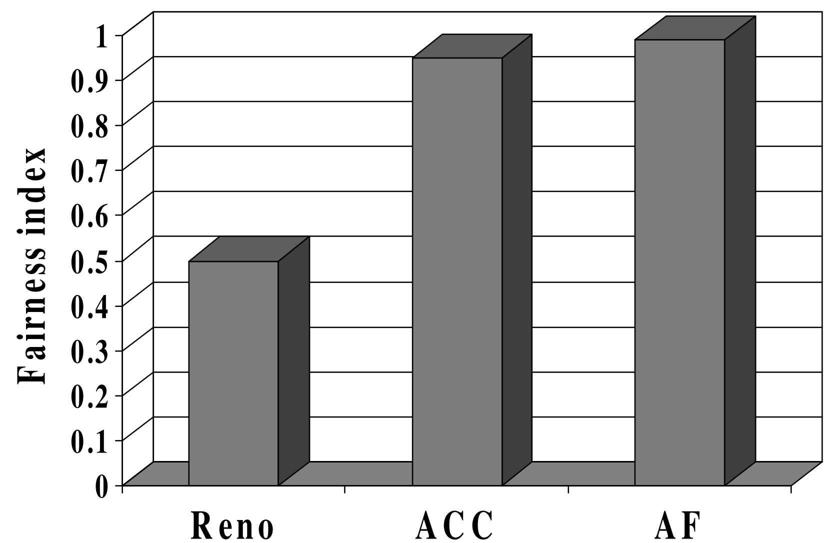


– Header compression helps
– Large reverse channel buffer hurts

Competing One-Way Transfers

– Two forward-direction transfers with 28.8 Kbps reverse channel with header compression





– ACC and AF help maintain free space in reverse channel buffer

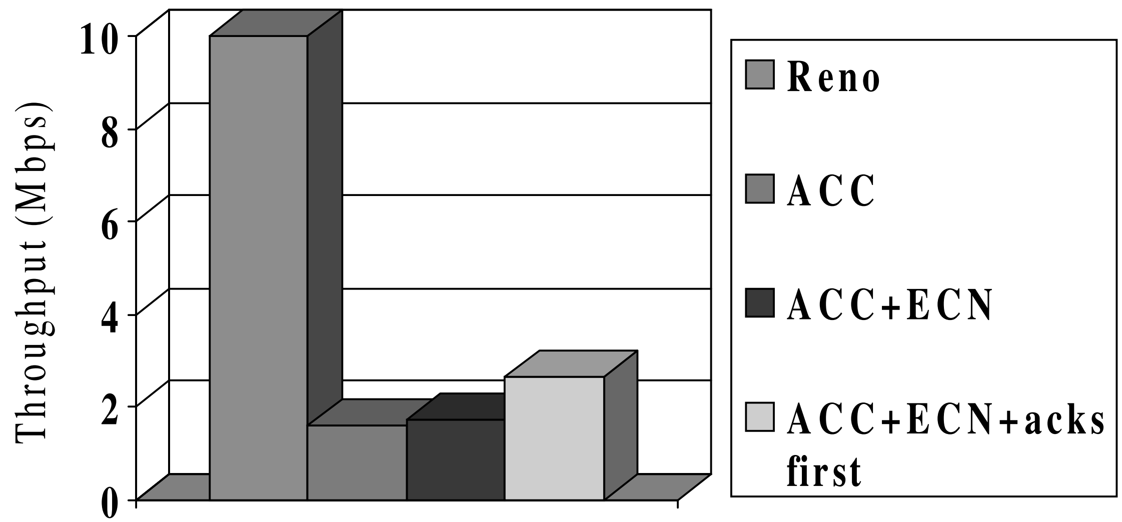
- fairness improves without degradation in throughput

Two-way Transfers

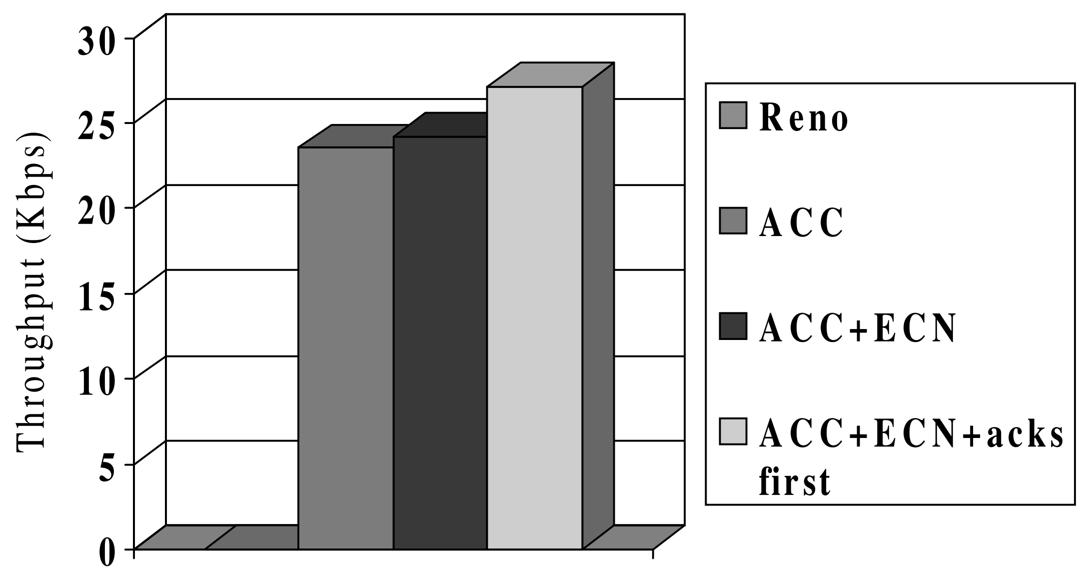
- Reverse transfer is initiated some time after forward transfer
- Maximum window size set to 100 KB

Two-way Transfers

Forward



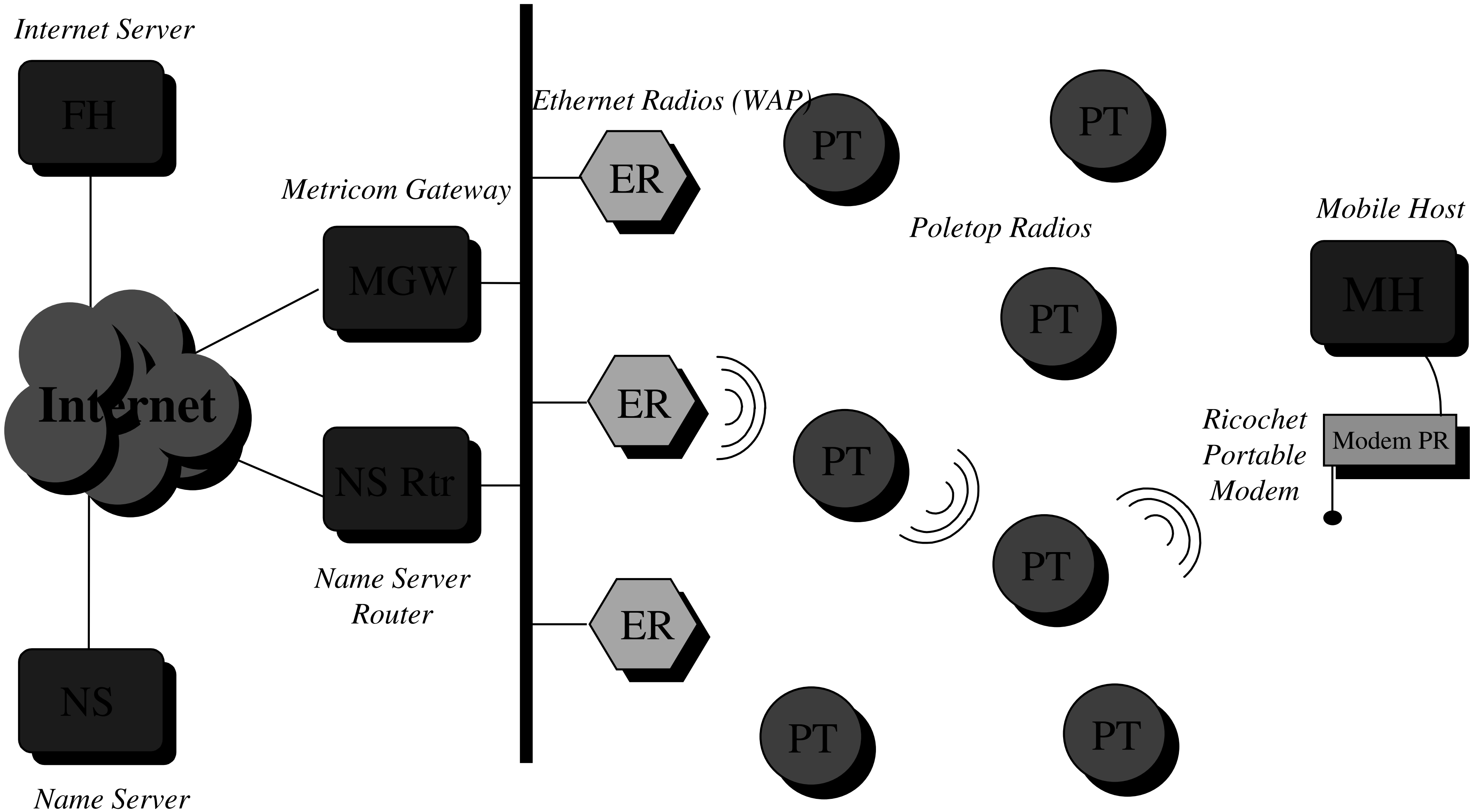
Reverse



Two-way Transfers

- Interaction between ack and data packets
- High degree of unfairness with TCP Reno
- ACC helps reverse transfer by not congesting reverse channel buffer
- Acks-first scheduling minimizes impact of (large) data packets on acks
  - 1 KB data packet takes 280 ms for transmission
  - max. possible forward throughput is 2.9 Mbps
  - throughput achieved is 2.67 Mbps

Ricochet Network Topology



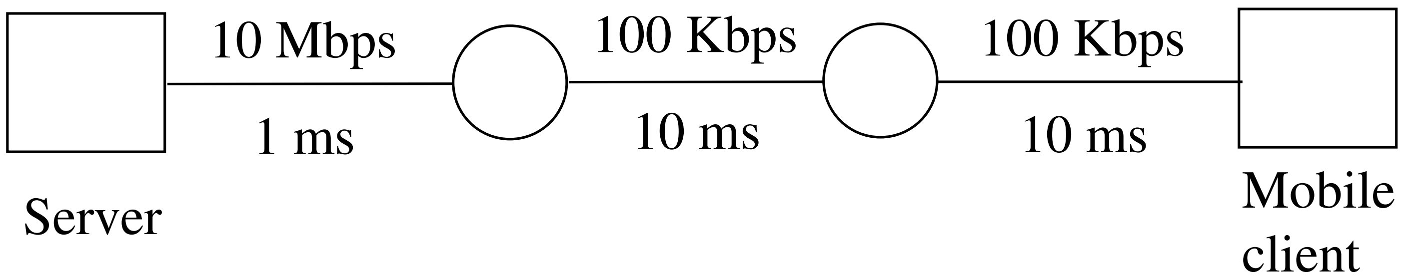
Media Access Issues

- Nodes in packet-radio network need to synchronize before they can communicate
  - poll/pollack procedure
  - radio turnaround time
  - exponential backoff if peer is busy
  - simple ACK/NACK based ARQ protocol
- Per-packet overhead is large and variable
  - increased packet count results in large and variable latency
  - in particular, the flow of acks adversely affects latency for data packets

Solutions

- Decrease the number of acks entering the packet radio network
  - ack congestion control
  - ack filtering
- Sender changes
  - window increase is tied to amount of data acknowledged
  - potential bursts broken up

Simulation Model



- 2 or 3 wireless hops
- radio turnaround time of 12 ms
- radio queue size of 10 packets
- exponential backoff in multiples of 20 ms slots

Results

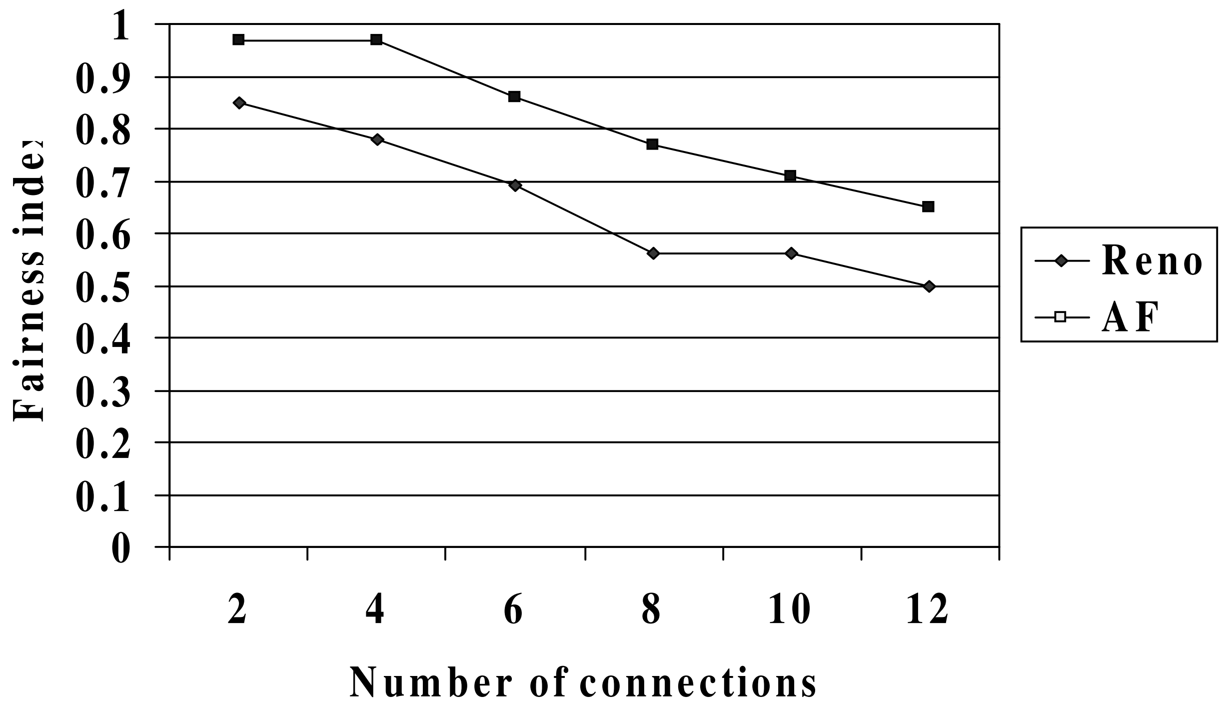
Effect on RTT and throughput

- Ack filtering decreases the chances that the peer radio is busy, so backoffs are less frequent
- 2 wireless hops
  - Reno: mean RTT = 2.67 s, std dev = 1 s
  - AF: mean RTT = 1.85 s, std dev = 0.6 s
  - 25% higher throughput with AF (24 Kbps versus 19 Kbps)
- 3 wireless hops
  - 34% higher throughput with AF (17.1 Kbps versus 12.7 Kbps)

Results

Effect on fairness

- simultaneous connections over 2-hop network
- ack filtering makes performance of each connection more predictable



Summary

- Flow of acks has a significant impact on TCP performance
- A good solution has several components
  - decreasing the frequency of acks when there is congestion in the reverse direction (ACC or AF)
  - priority scheduling of acks (acks-first)
  - sender adaptation to combat infrequent acks

Future Work

- Performance with short transfers
- Receiver feedback to aid fast window growth
  - receiver tells sender the rate at which it is receiving data packets
- Sender-based detection of ack congestion
- *Ack reconstructor* to shield sender from effects of infrequent acks
  - inserts acks to bridge large gaps in sequence
  - spaces apart bursts of acks
- Implementation and validation on testbed