Coded TCP

(work with: Muriel Medard, Jason Cloud,
Ali ParandehGheibi, MinJi Kim)

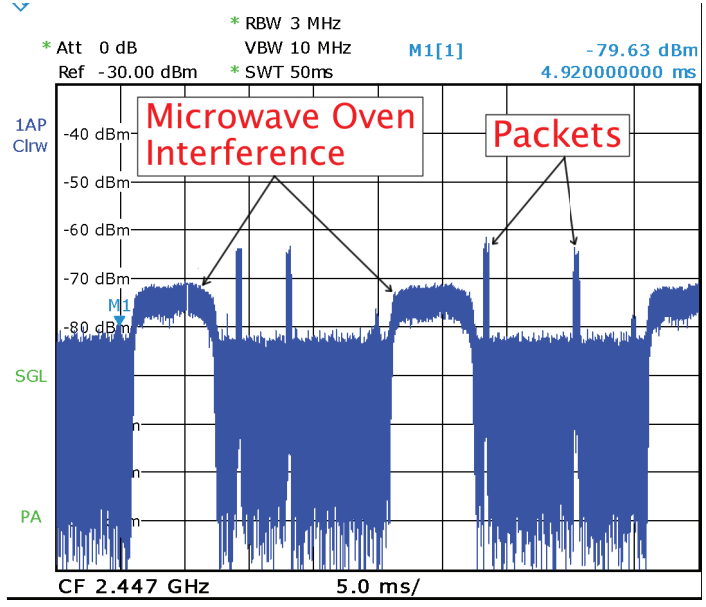
Doug Leith

Outline

- Why do error-correction coding at the transport layer ?
- Error-correction coding using delayed feedback
- Congestion control on lossy paths
- Implementation & performance measurements

Why error-correction coding at the transport layer ?

- Interference-related losses are challenging, yet increasingly common in dense 802.11 deployments
- Hidden terminals
- New dynamic devices e.g. channel bonding, will only make this worse
- Microwave interference in unlicensed band

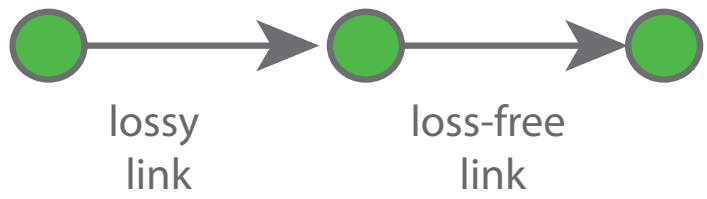


Why error-correction coding at the transport layer ?

Why not enhance error-correction at the link layer ? Link layer offers many advantages:

- Link layer has access to low-level information e.g. whether a packet loss is due to queue overflow or channel error.
- Usually quick feedback, so ARQ efficient
- Hop by hop encoding is generally more efficient than end-to-end encoding

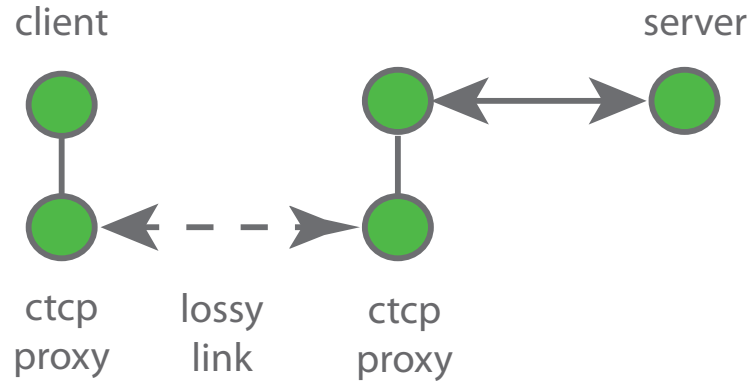
If link layer improvements are possible, make them !



Why error-correction coding at the transport layer ?

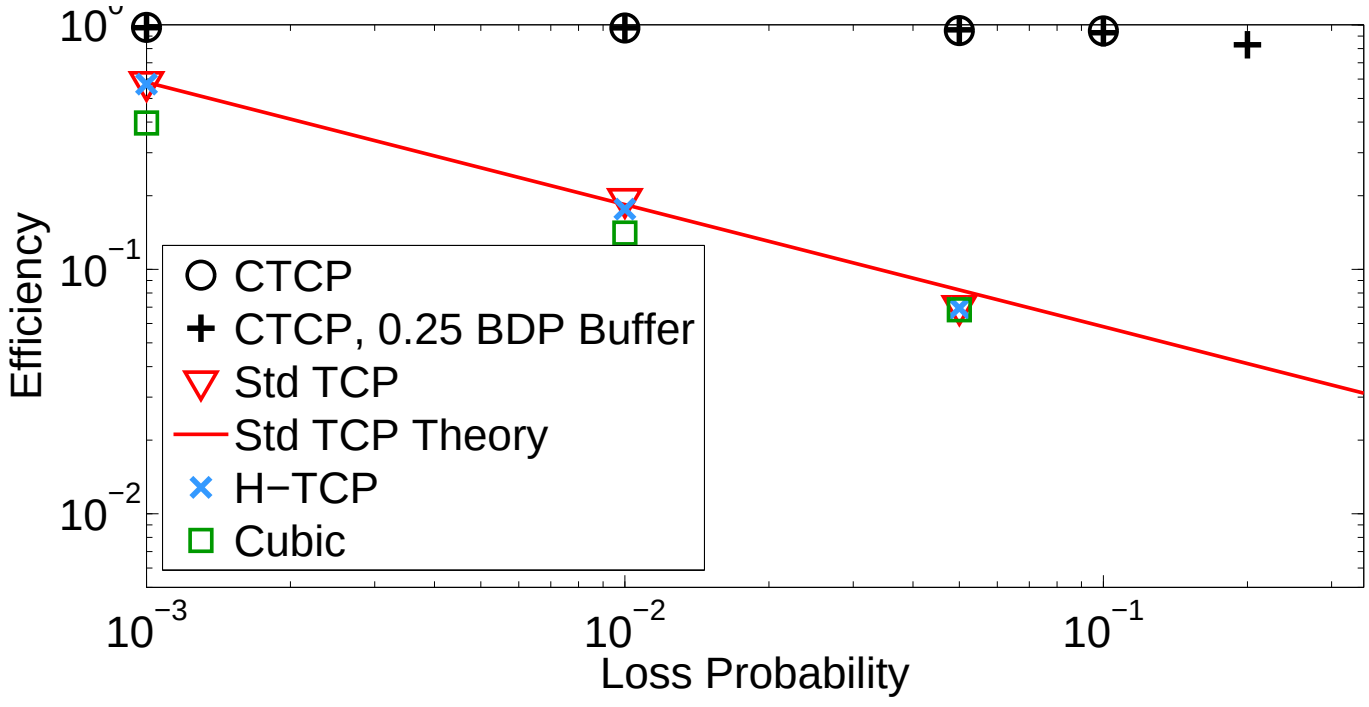
Transport layer has some compelling practical advantages:

- No need for changes to installed network equipment
- No need for root-privilege changes to user equipment, just a user-space app
- No need for changes to installed servers



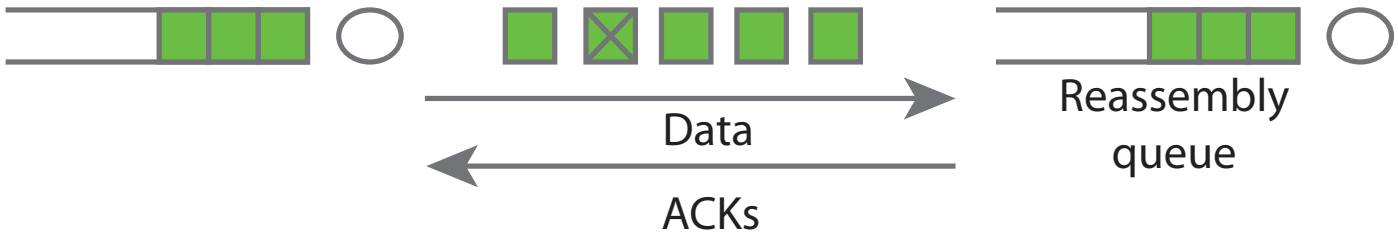
Why error-correction coding at the transport layer ?

Plus potential exists for considerable performance gains.



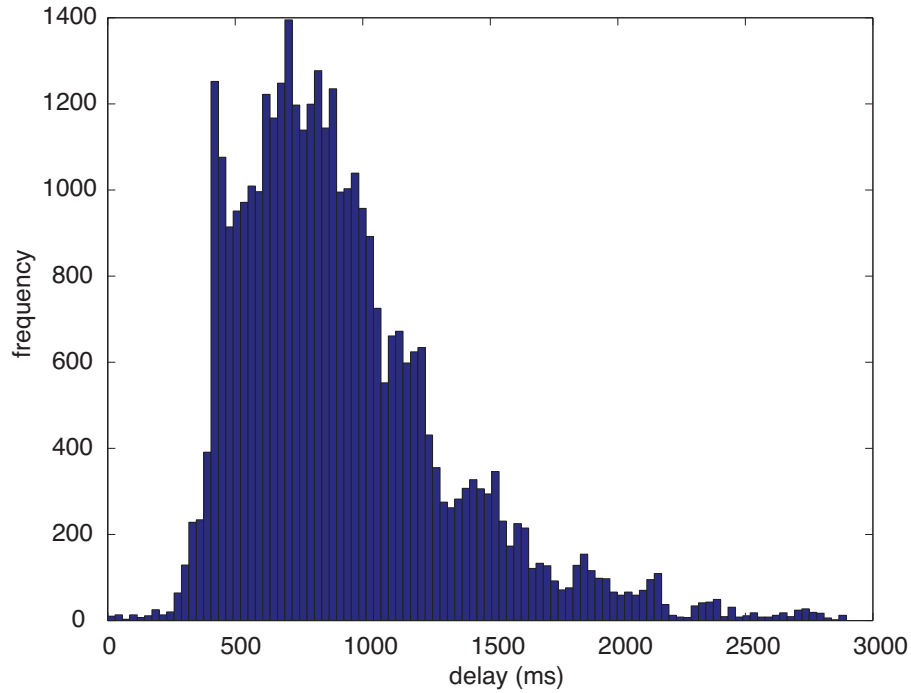
Link 25Mbps, RTT 20ms

Error-correction coding using delayed feedback



- Packets may be erased in transit
- In-order delivery at receiver via reassembly queue
- Feedback to sender via ACKs
- May be large path delay/RTT e.g. 20-50ms
  $\Longrightarrow$ 100s of data packets in flight, feedback to sender is delayed.

ARQ



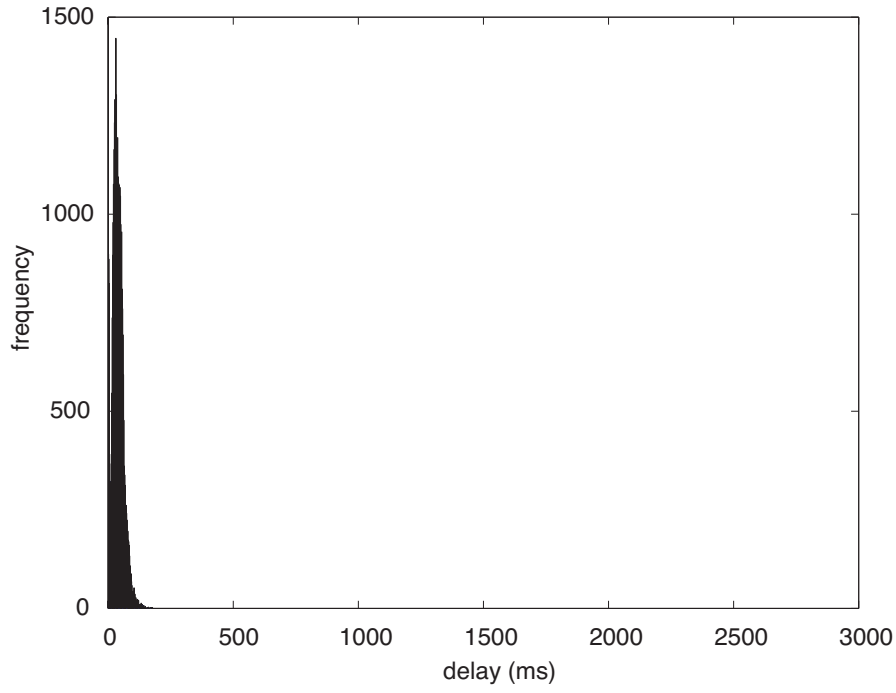
10Mbps/25ms link, 20% loss rate, 50MB file transfer

Measured packet delay, less RTT, using ARQ only.

Hybrid coding approach with delayed feedback

- FEC based on currently estimated loss rate (from ACKs) i.e. send extra packets in an attempt to preempt loss
- Use feedback to deal with cases when this fails, send additional coded packets
- A throughput-delay trade-off here
- Use RLC in GF(256) as FEC code, but could use something else
- Block size of 32 packets

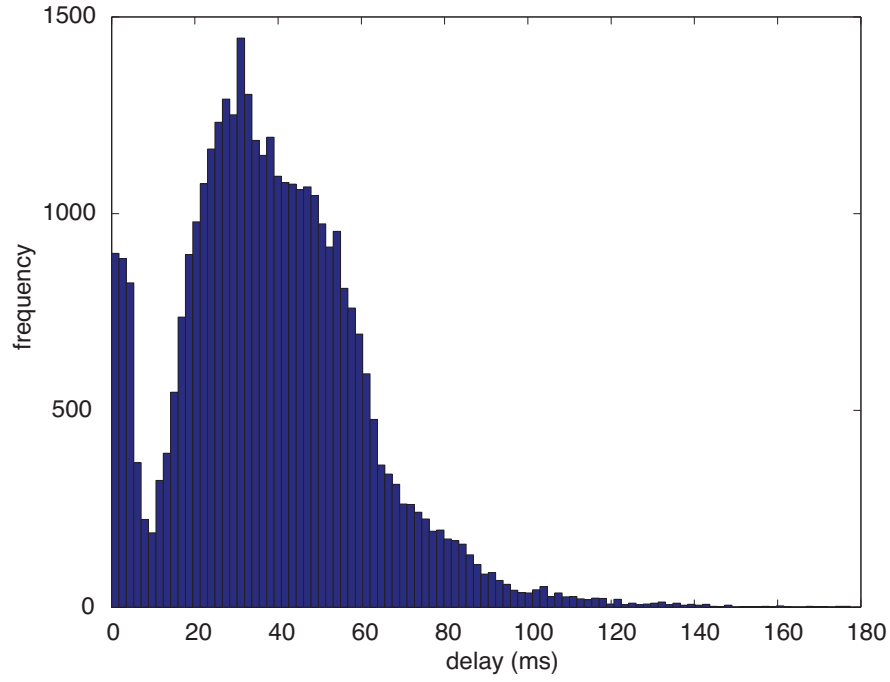
Hybrid coding approach with delayed feedback



10Mbps/25ms link, 20% loss rate, 50MB file transfer

Measured packet delay, less RTT, hybrid FEC/feedback.

Hybrid coding approach with delayed feedback

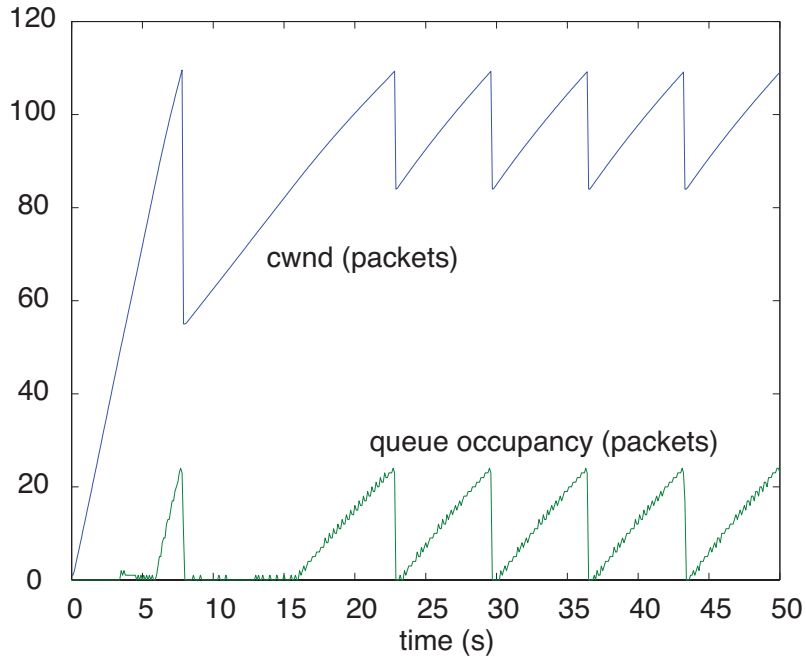


10Mbps/25ms link, 20% loss rate, 50MB file transfer

Measured packet delay, less RTT, hybrid FEC/feedback.

Congestion control on lossy paths

- Modify AIMD backoff on loss to $cwnd \leftarrow cwnd \times \frac{RTT_{min}}{RTT}$
- Never ignores packet loss
- Reverts to standard TCP on links without noise losses
- On lossy links yields dramatic improvement in throughput by avoiding *cwnd* collapse.
- An important source of the x10-x20 thoughput gains observed.

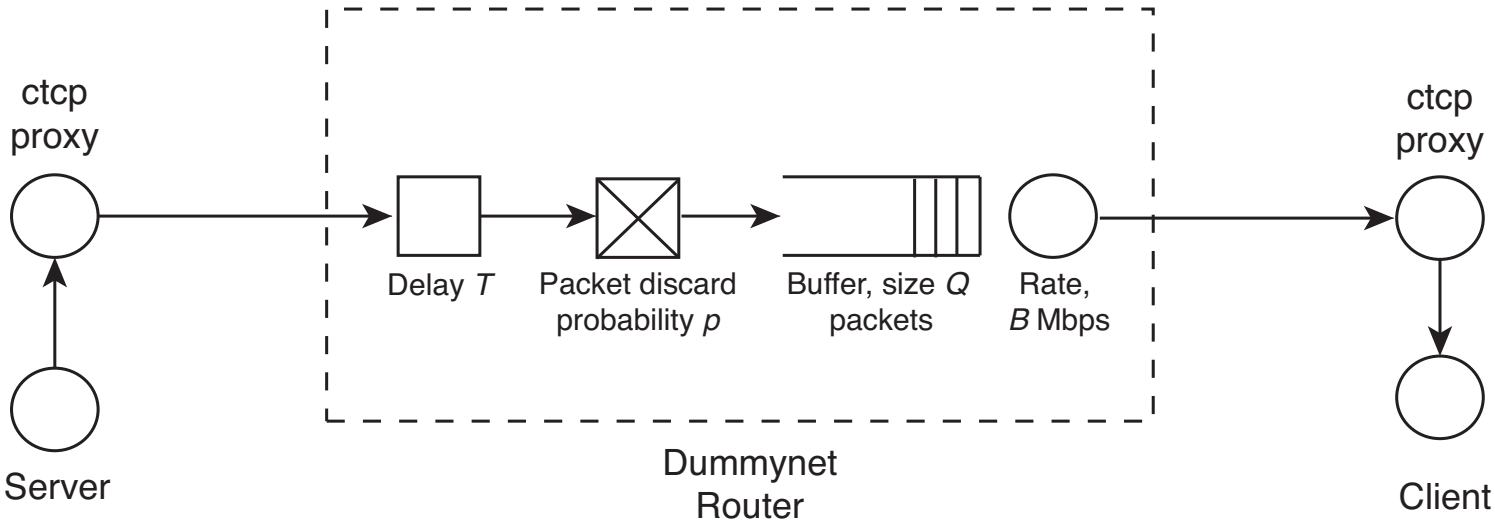


Implementation Options

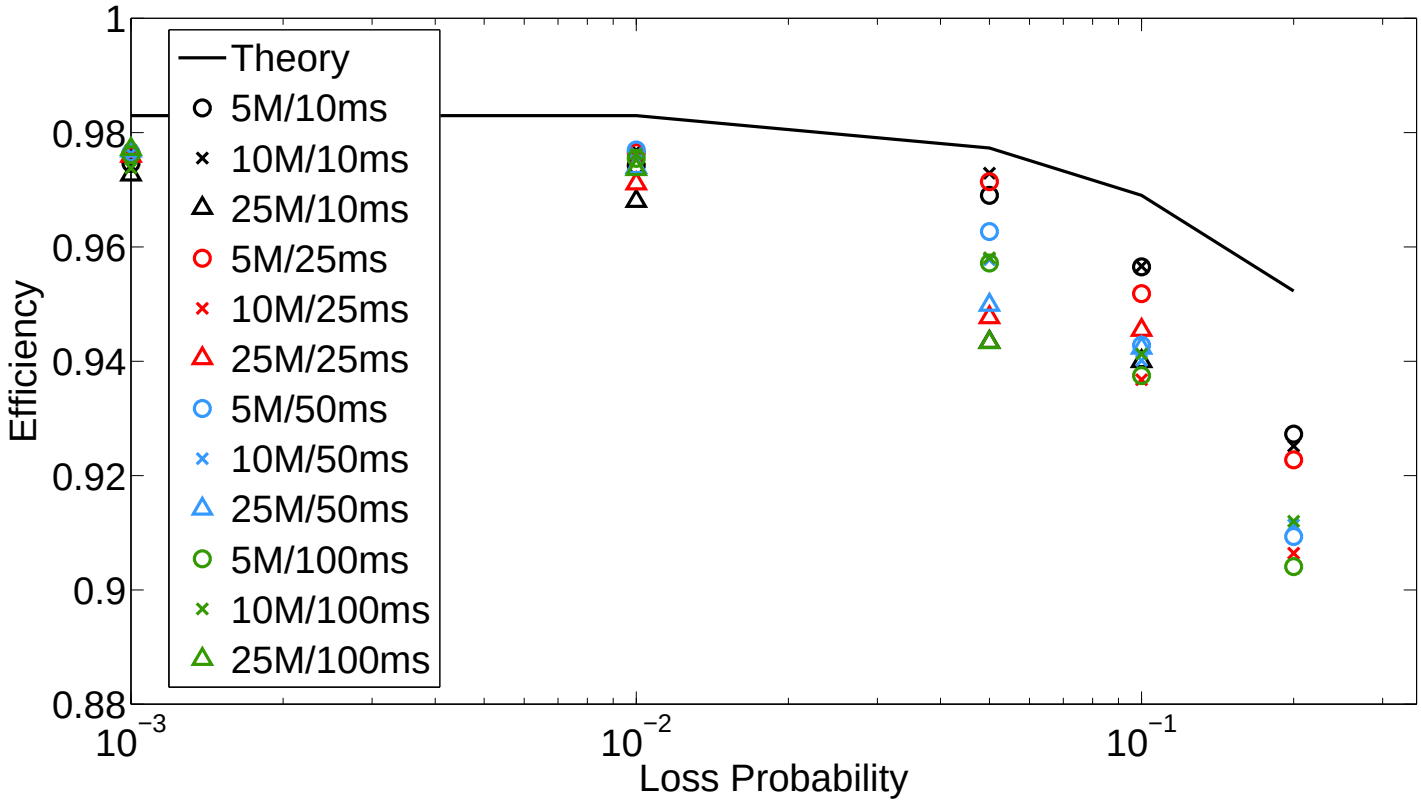
- User-space vs kernel – user-space offers greater portability and no need for root priviledge
- Proxy vs tunnel – user-space and lack of root access restricts use to standard sockets, so proxy-like approach
- UDP to carry CTCP packets – TCP changes require kernel modifications.
- Link layer coding, within driver/kernel, may indicate different design decisions ...

Link layer-agnostic measurements

Testbed setup:

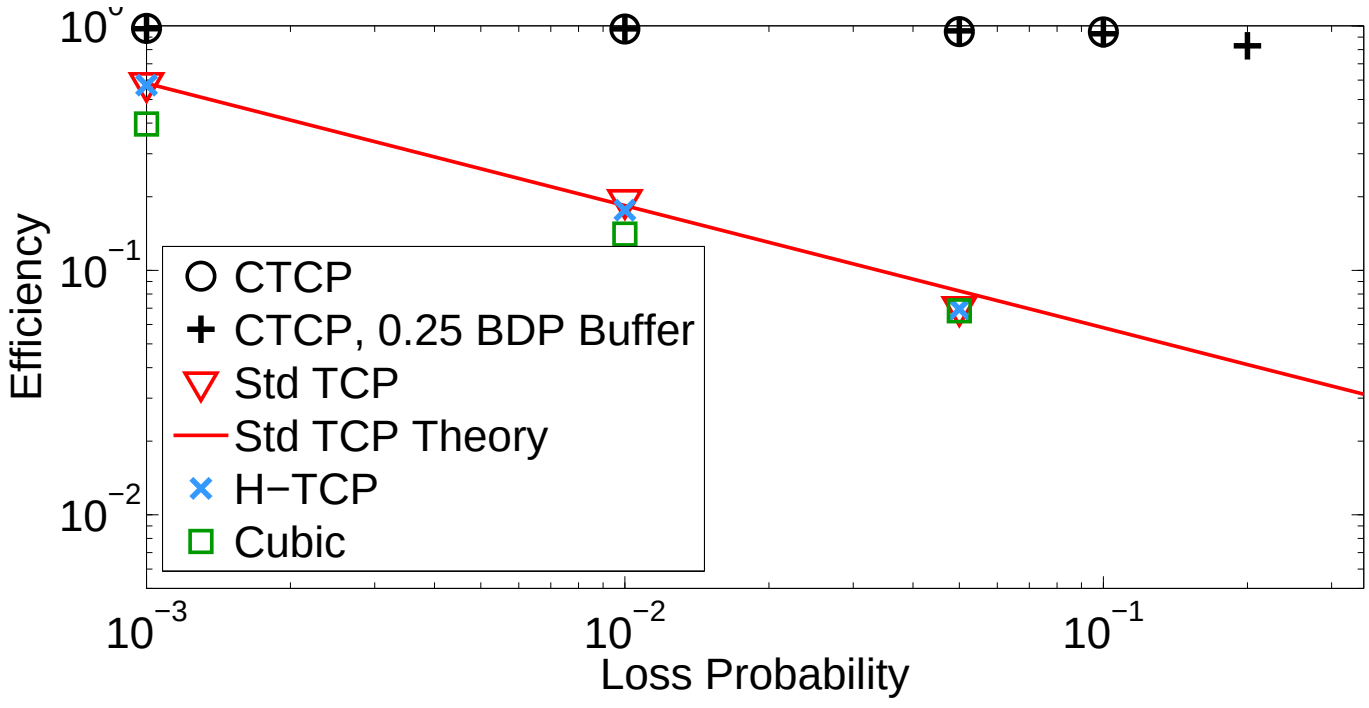


Link layer-agnostic measurements



Link 25Mbps, RTT 20ms

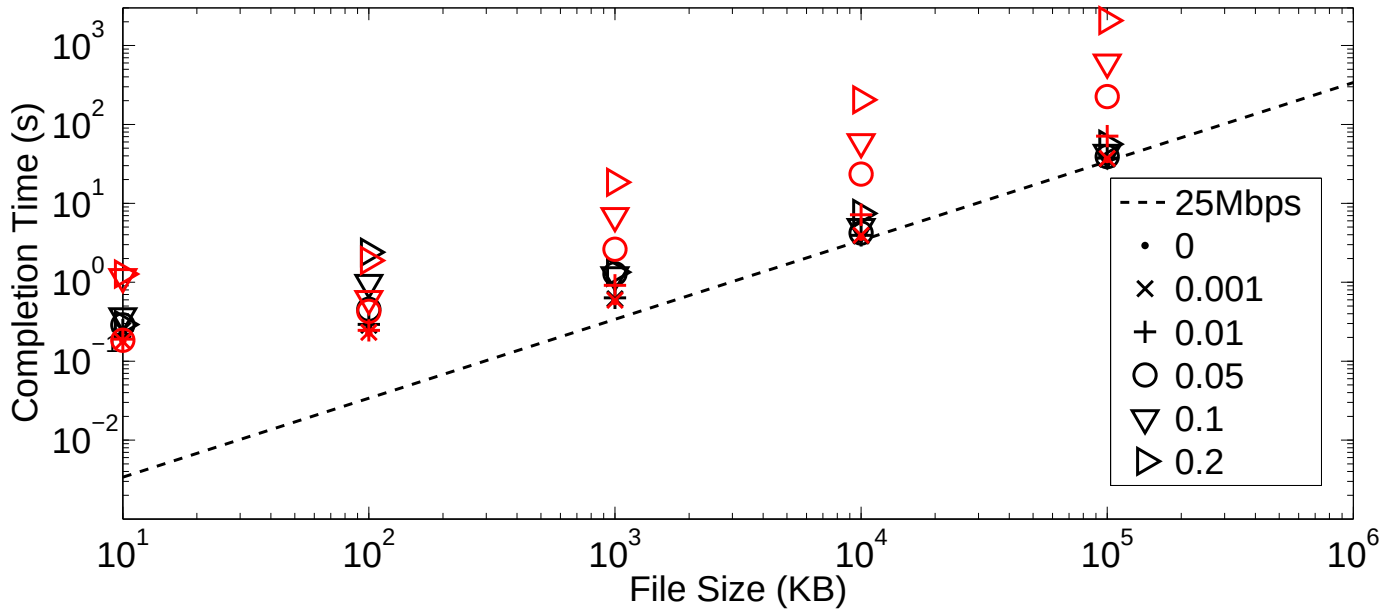
Link layer-agnostic measurements



Link 25Mbps, RTT 20ms

Link layer-agnostic measurements

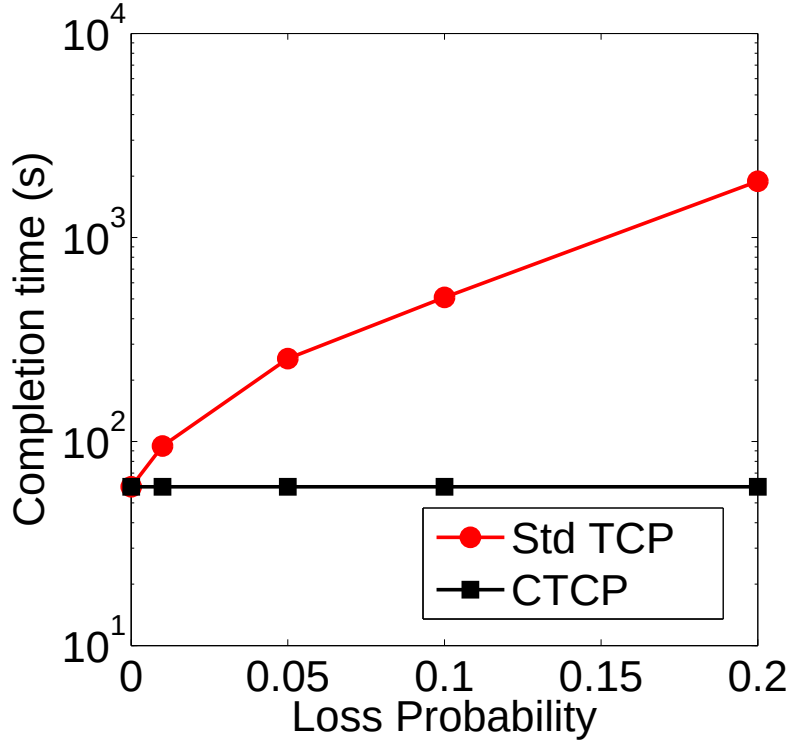
Application performance - HTTP



25Mbps link, RTT 10ms
Standard TCP (red) and CTCP (black)

Link layer-agnostic measurements

Application performance - Video streaming

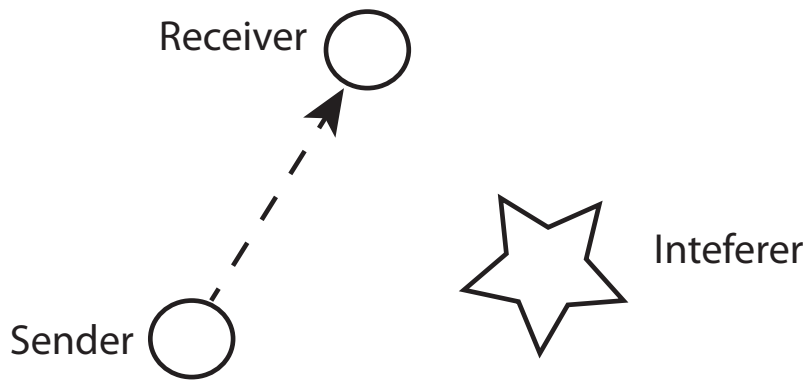


25Mbps link, RTT 10ms, 60s video playout
Standard TCP (red) and CTCP (black)

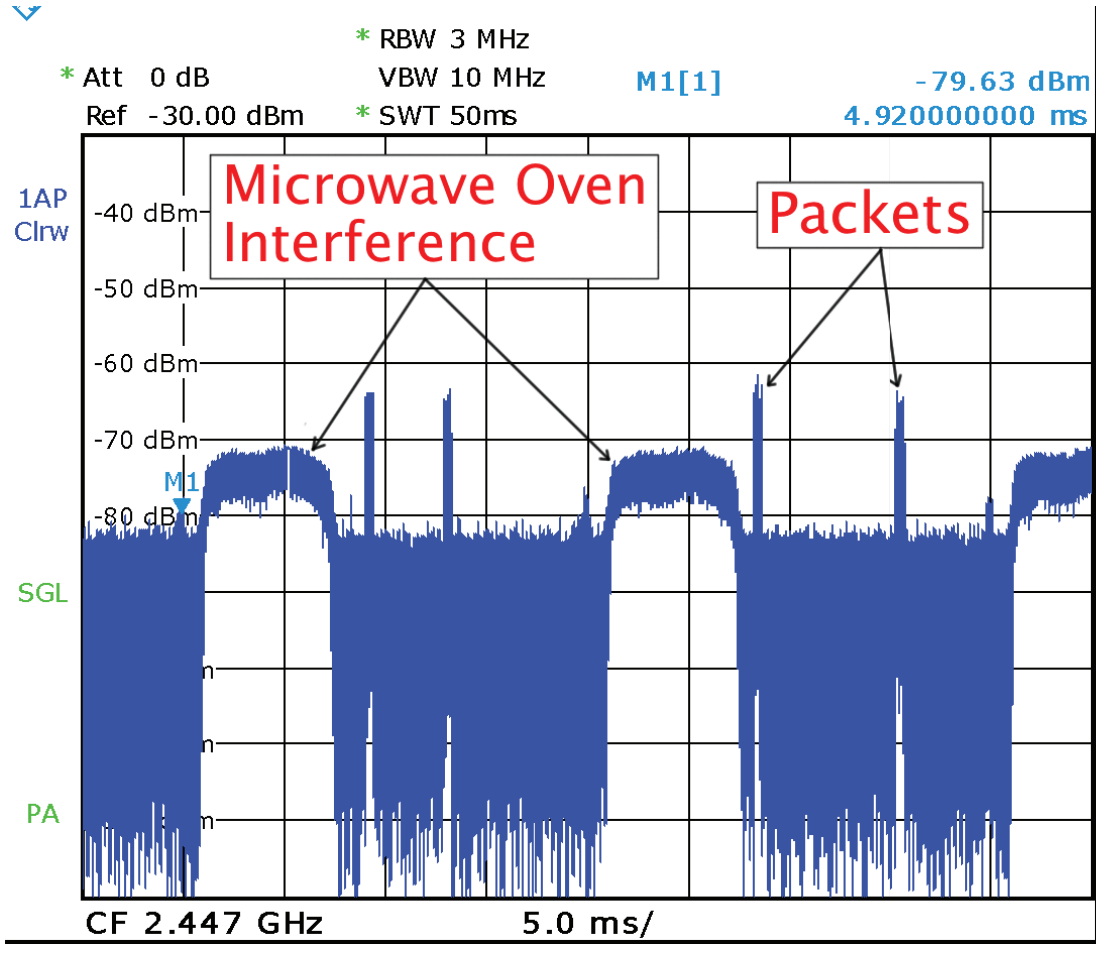
802.11 wireless measurements

Testbed setup:

- Proprietary 802.11 features disabled
- 802.11 rate control manual
- Cubic as standard TCP.

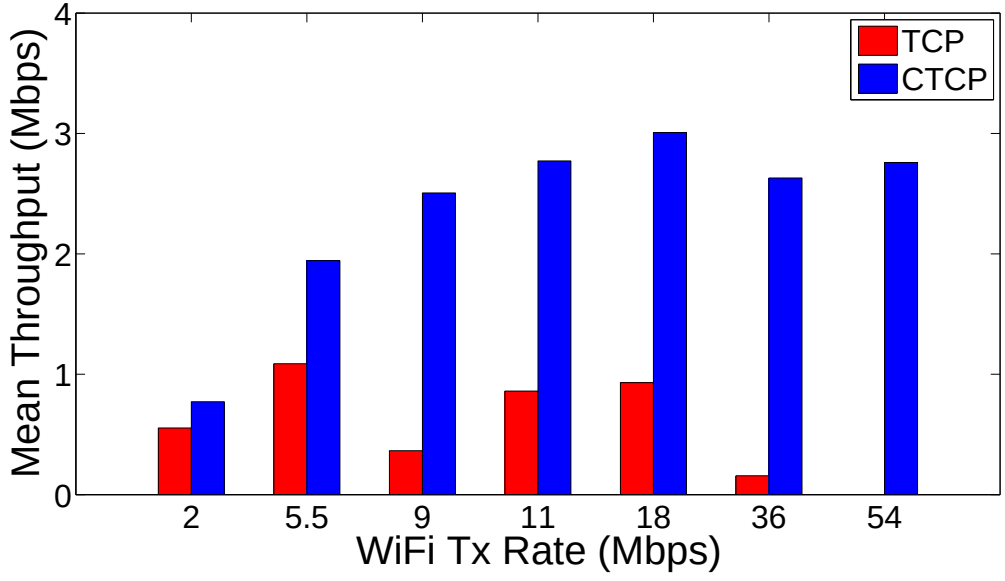


802.11 wireless measurements



802.11 wireless measurements

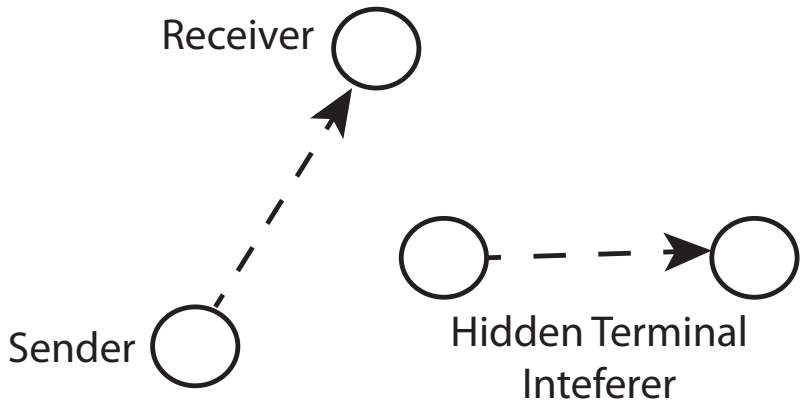
Microwave oven interference



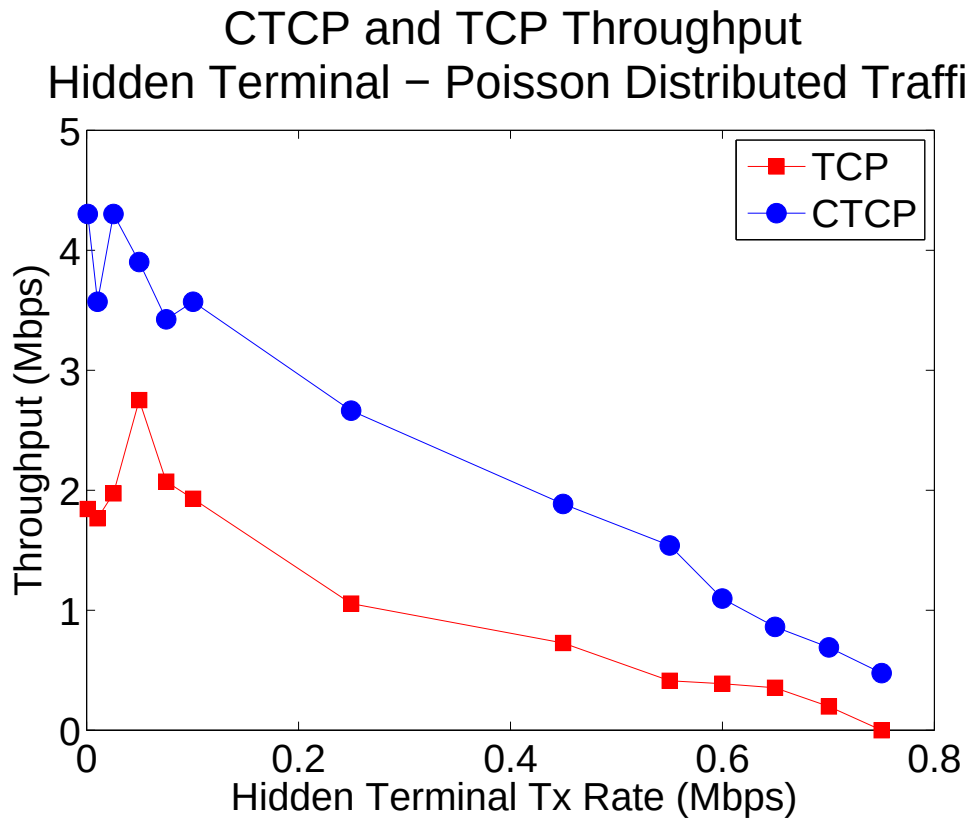
802.11 wireless measurements

Hidden terminal setup:

- Modified 802.11 driver to disable carrier sense
- Poisson interference traffic



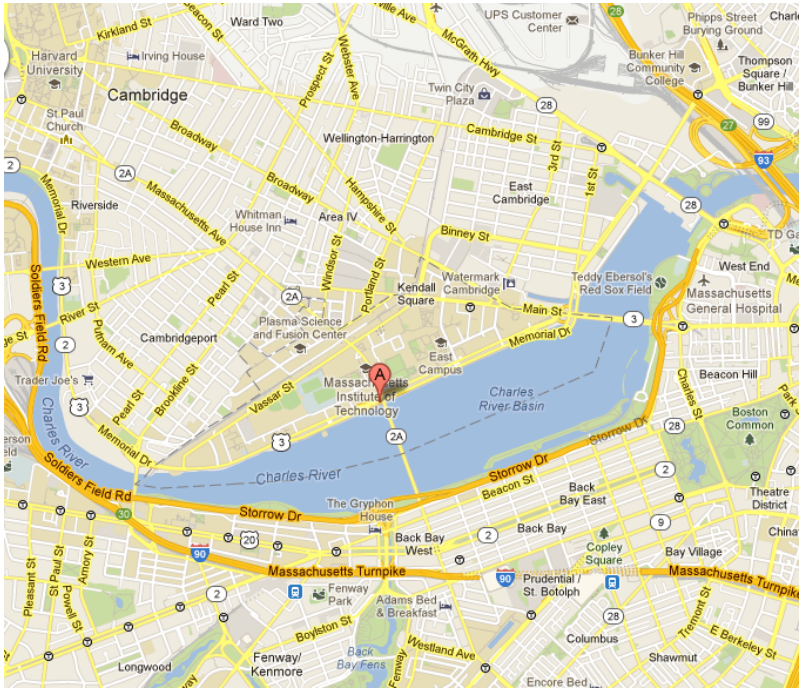
802.11 wireless measurements



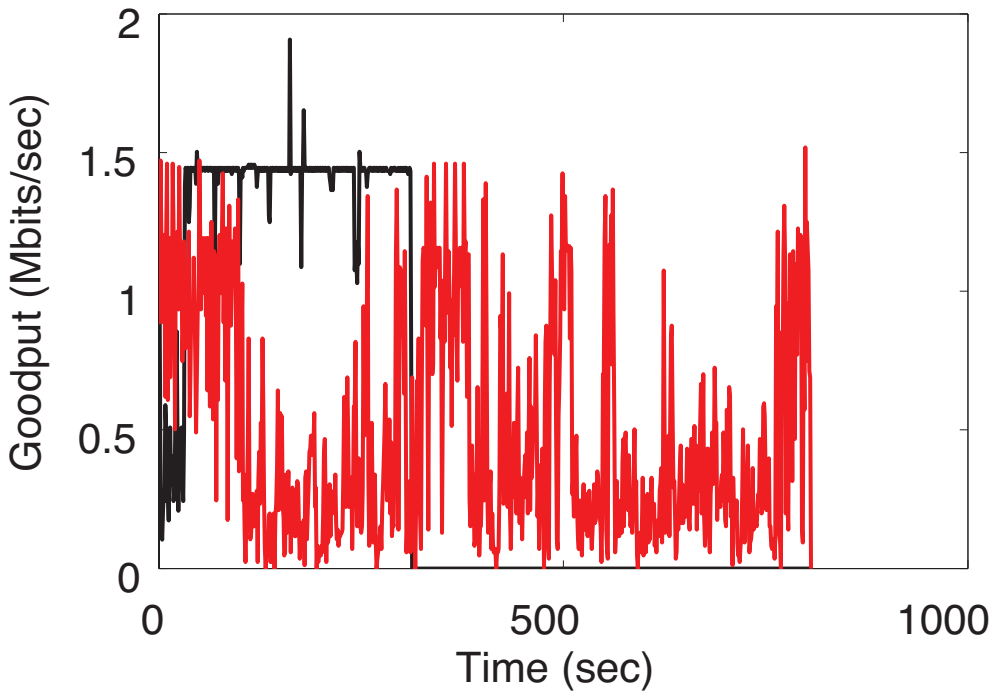
Throughput vs intensity of hidden terminal interference when using standard TCP (Cubic TCP) and CTCP over an 802.11b/g wireless link.

802.11 hot spot measurements

- Various public WiFi networks in the greater Boston area
- Downloaded a 50 MB file from a server located on MIT camput to a laptop
- Default operating system (Ubuntu) settings are used for all network parameters on client and server.



802.11 hot spot measurements



Standard TCP (red), CTCP (black)

Workshops

Organised by Code On:

1. Oct 15-16, Berlin
2. Nov 5-6, Palo Alto

See http://www.codeontechnologies.com/training/ for more details.