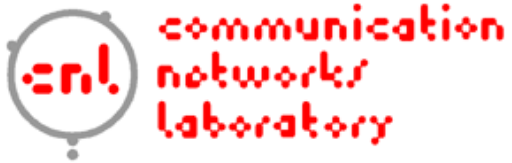
TCP ACK Control for Wireless Networks

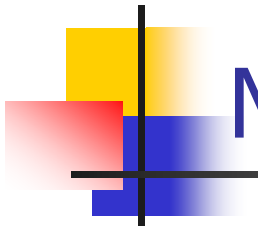
Wan Gang Zeng
wgzeng@sfu.ca

Communication Networks Laboratory
http://www.ensc.sfu.ca/research/cnl
Simon Fraser University



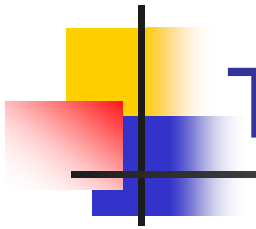
Road map

- Motivation
- TCP and wireless networks
- Wireless TCP performance
- Proposed algorithm: ACK Controller
- Simulation and conclusions
- References



Motivation

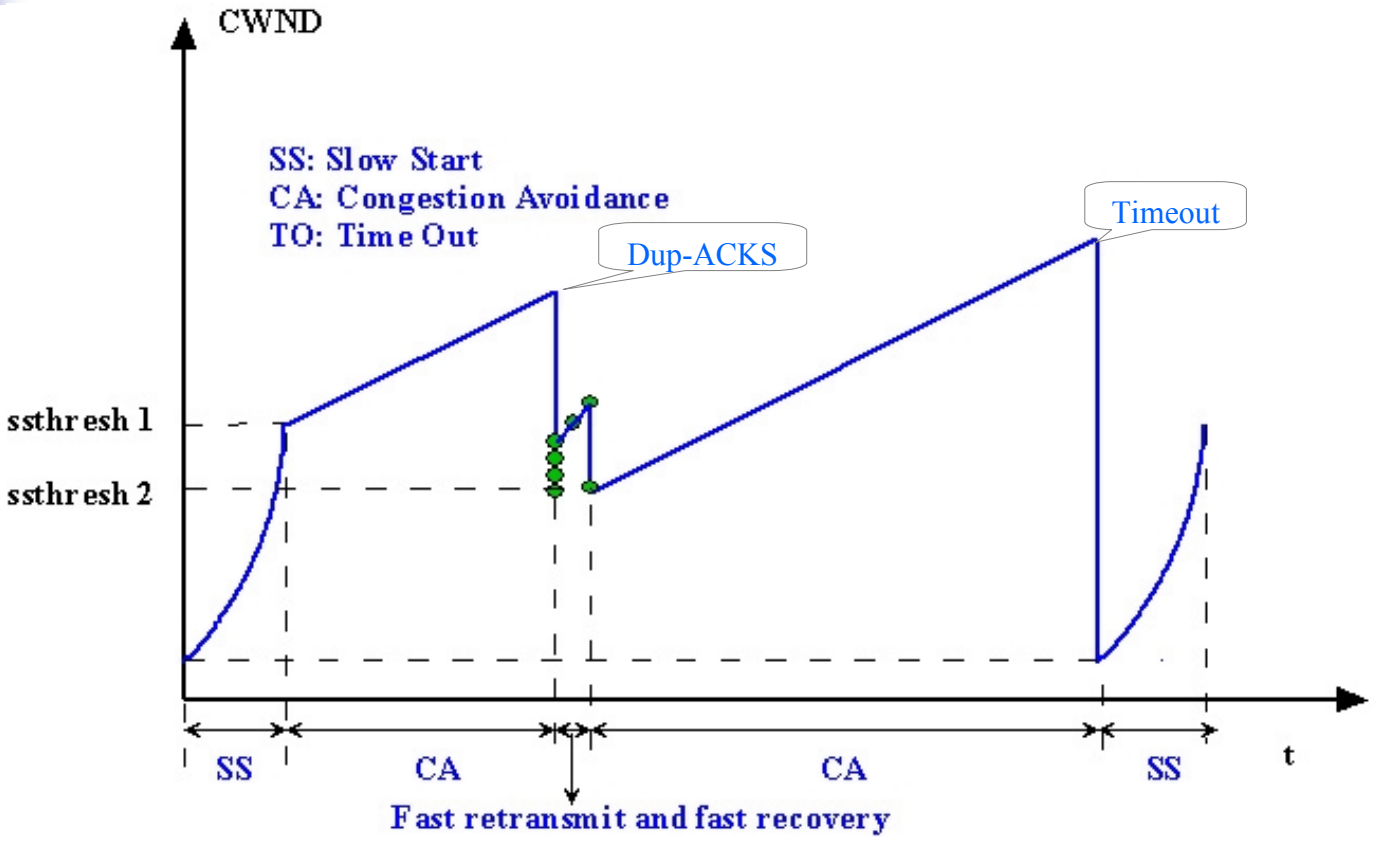
- Transmission Control Protocol (TCP) is the most widely used transport protocol in wireline networks:
  - it carries 95% of Internet traffic
- TCP is suitable for data applications
- Important to support application over wireless and wireline links:
  - laptops, PDA, data-capable cell phones, 3G
- Mobile devices require TCP features similar to wireline networks
- TCP on wireless links demands special attention



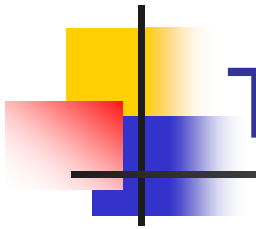
TCP

- Transport protocol
- Typically designed for wireline networks
- Essential: ACK paced transmission, window management, and timer based retransmission
- Services:
  - reliability
  - congestion control
  - connection management
  - flow control

TCP: congestion control



cwnd: congestion window
rwnd: receiver's advertised window



TCP: congestion control

- Monitor packet losses:
  - 3-dup ACKs, retransmission timeouts
  - due to congestion in network
- Congestion control:
  - congestion windows (cwnd) size decreases
  - amount of data sent into network decreases
    - sending window = min (cwnd, rwnd)
  - throughput decreases

TCP: RTT, RTO and Karn's algorithm

- RTO is calculated based on RTT and RTT deviation
- Karn's algorithm:
  - sample RTT: sampleRtt is measured for data packets
  - smoothed RTT:

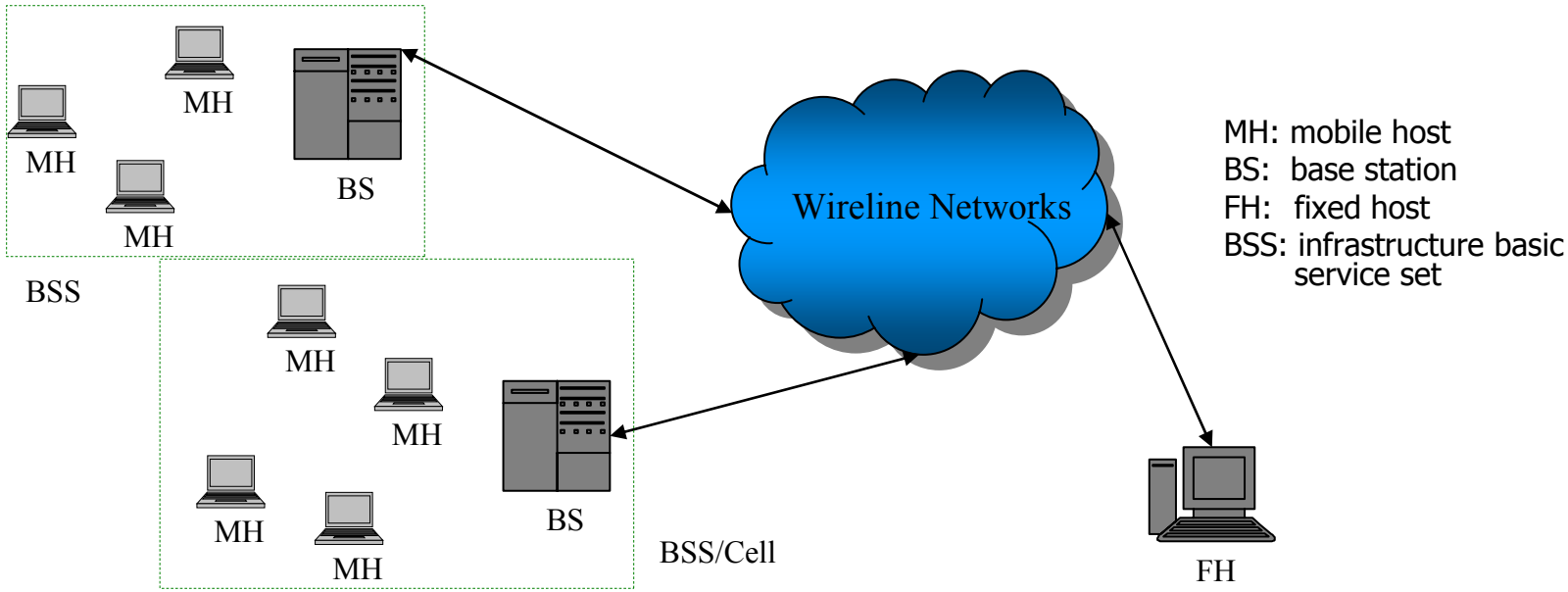
    srtt = (1 – g) * srtt + g * sampleRTT
  - mean deviation:

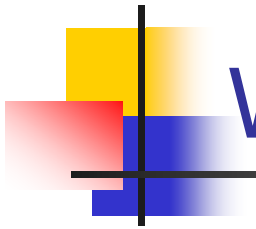
    rtttvar = (1 – h) * rtttvar – h * | sampleRTT – srtt |
  - recommended values for g, h

    g = 1/8 = 0.125, h = ¼ = 0.25
  - RTO = srtt + 4 rttvar

Wireless network: architecture

- Conventional cellular networks and wireless LAN (WLAN)
- Assumption: MHs are directly connected to BS connected to a wireline Internet.



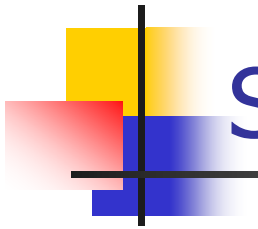


Wireless network: properties

- Properties:
  - high bit-error rate (BER): random loss
  - bursty traffic: mixed voice/data, channel access asymmetry
  - disconnections: handoffs, interferences
- Impact on TCP:
  - fast retransmit, timeouts, large and varying delay

TCP performance: link errors

- Cannot differentiate packet losses caused by congestion from link errors
- Assumes that packet loss is due to congestion and resolves it by decreasing sending rate
- In wireless networks:
  - high probability of packet loss caused by transmission error or mobility
  - temporary
  - network can recover itself
  - congestion control degrades TCP performance



Solutions

- General approach:
  - hide error losses from the sender
  - inform the sender of the cause of packet loss
- Specific designs:
  - split-connection: I-TCP, M-TCP
  - link layer solution: Snoop
  - end-to-end: TCP Westwood, TCP-Jersey

TCP performance: large sudden delay and delay variation

- Large sudden delay and delay variations are caused by:
  - wireless link properties: limited bandwidth, randomness of wireless channel
  - protocols: link layer or media access control (MAC) retransmission schemes
  - queuing algorithms
  - device mobility
  - handoffs
  - different traffic priorities

Large sudden delay and delay variation: adverse effects

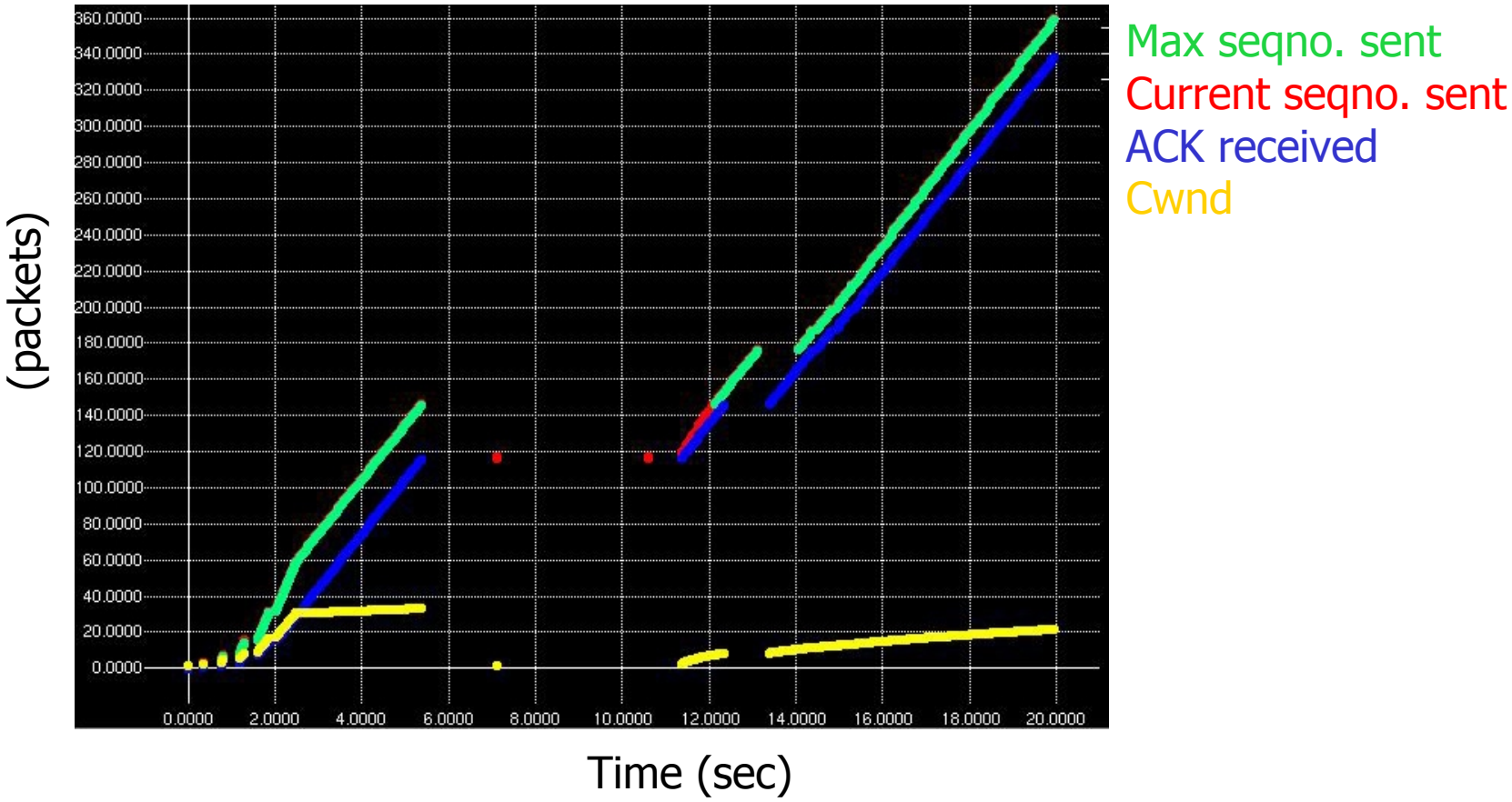
- TCP does not react well to large sudden delay and delay variations on links
- Delay variation causes:
  - spurious fast retransmit
    - caused by packet reordering
    - TCP receives 3-duplicate ACKs
  - ACK compression
    - TCP sender receives accumulated ACKs
    - increases traffic burstiness and the chances of bursty packet losses

Large sudden delay and delay variation: adverse effects

- Large sudden delay causes:
  - ACK compression
  - spurious timeouts
    - TCP's RTT prediction cannot react fast enough to the sudden delay change
  - spurious fast retransmit
    - triggered by TCP's retransmissions after timeouts

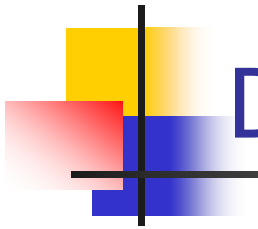
SeqNo, ACK, and Cwnd (ns-2 simulation)

- Large sudden delay -> spurious timeout -> retransmission -> duplicate ACKs (spurious fast retransmit) -> longer delay



Road map

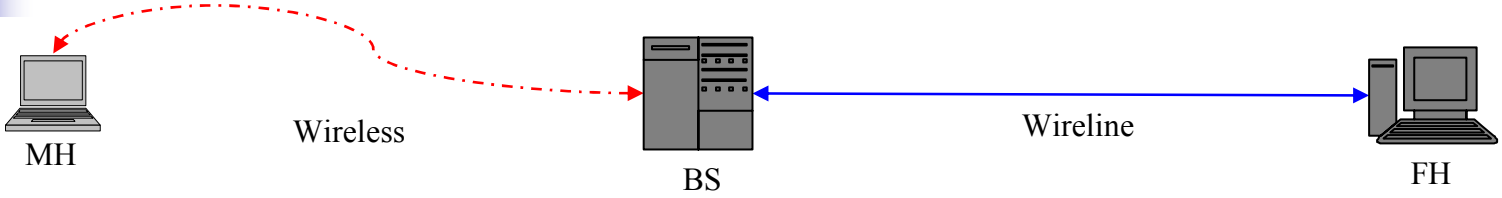
- Motivation
- TCP and wireless networks
- Wireless TCP performance
- Proposed algorithm: ACK Controller
- Simulation and conclusions
- References



Design goals

- We consider wireless link characteristics:
  - large sudden delay and delay variation
  - handoff or short disconnections
  - short TCP connections
- The approach is to introduce minimum changes in TCP:
  - simple to implement
  - easy to deploy

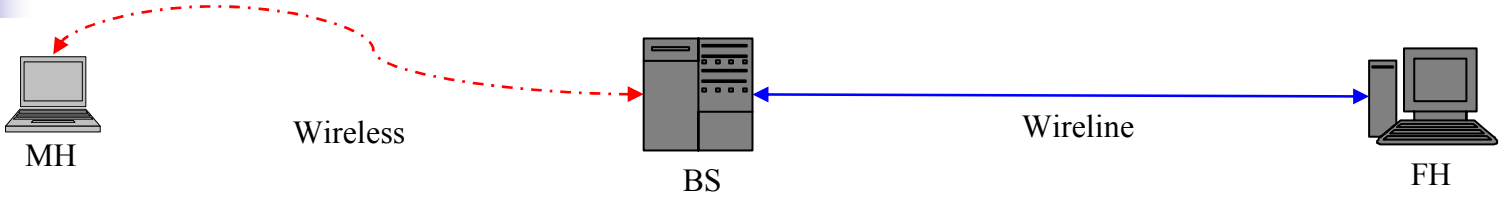
Proposed algorithm: spurious fast retransmission



- Modify BS
- Perform ACK control for delay variation:
  - estimate delay between BS and MH:
    - record timestamp for each packet received
    - for each received ACK (non-dup ACK), update smoothed RTT in BS for RTTs between BS-MH

      sRTT = (1 – g) * sRTT + g * sampleRTT, for some g
  - enforce ACK returning rate:
    - based on sRTT, control ACK returning rate to FH
  - duplicate ACK filter:
    - redefine "3-dup ACK" for wireless link (dupAckThresh)

Proposed solution : spurious fast retransmission



- Parameters g and dupAckThresh are:
  - global for the wireless link between BS and MH
  - are connection independent
  - they reflect properties of the link
- Possible performance improvements for multiple connections

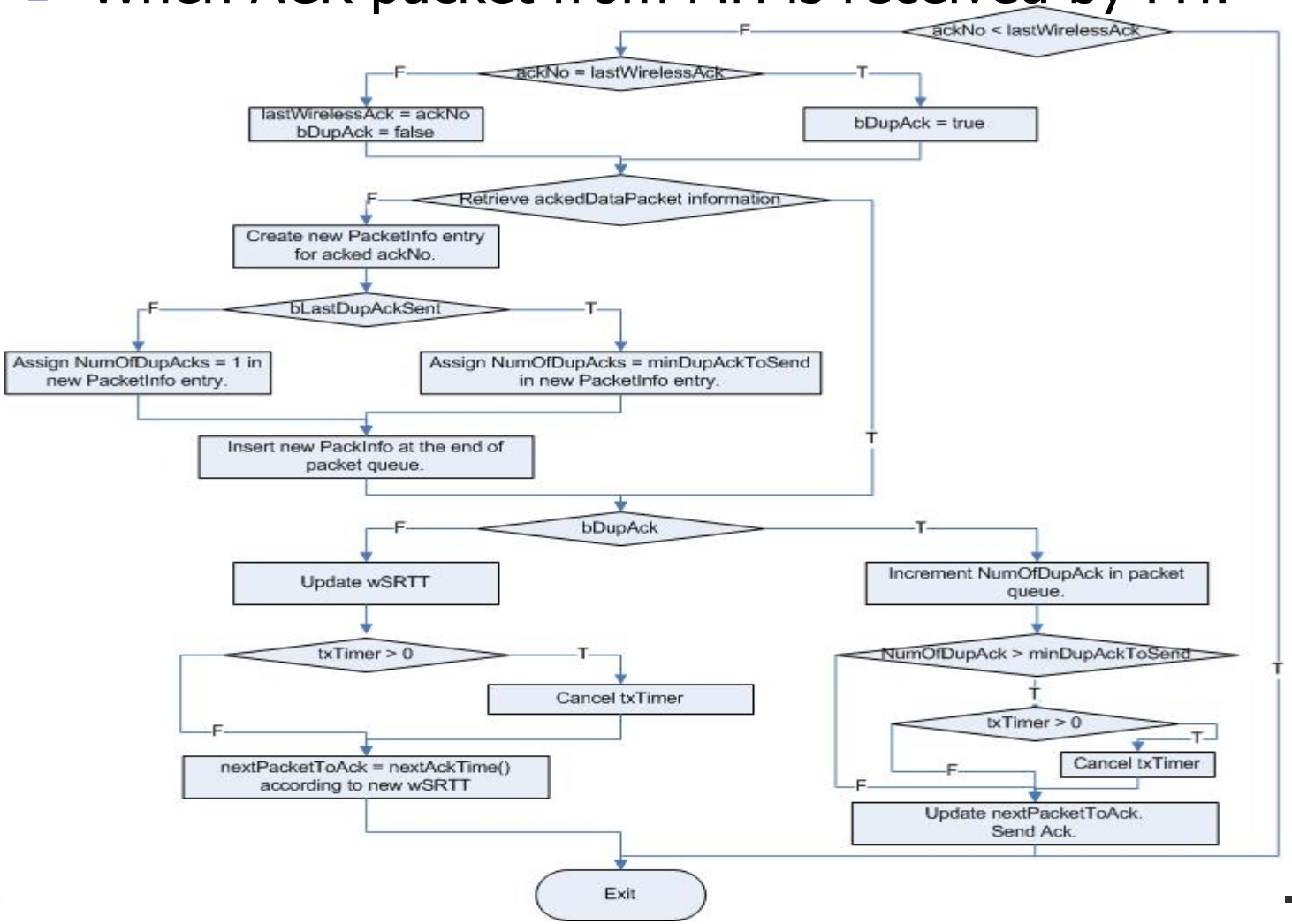
Implementation: receiving data packet

- When data packet received from FH to MH:
  - record seqno, current time in packet queue

```
struct PacketInfo
{
  int m_iSeqno;
  //time packet was recieved
  double m_dSendTime;
  int m_iNumOfDupAcks;
}
```

Implementation: receiving ACK

- When ACK packet from MH is received by FH:

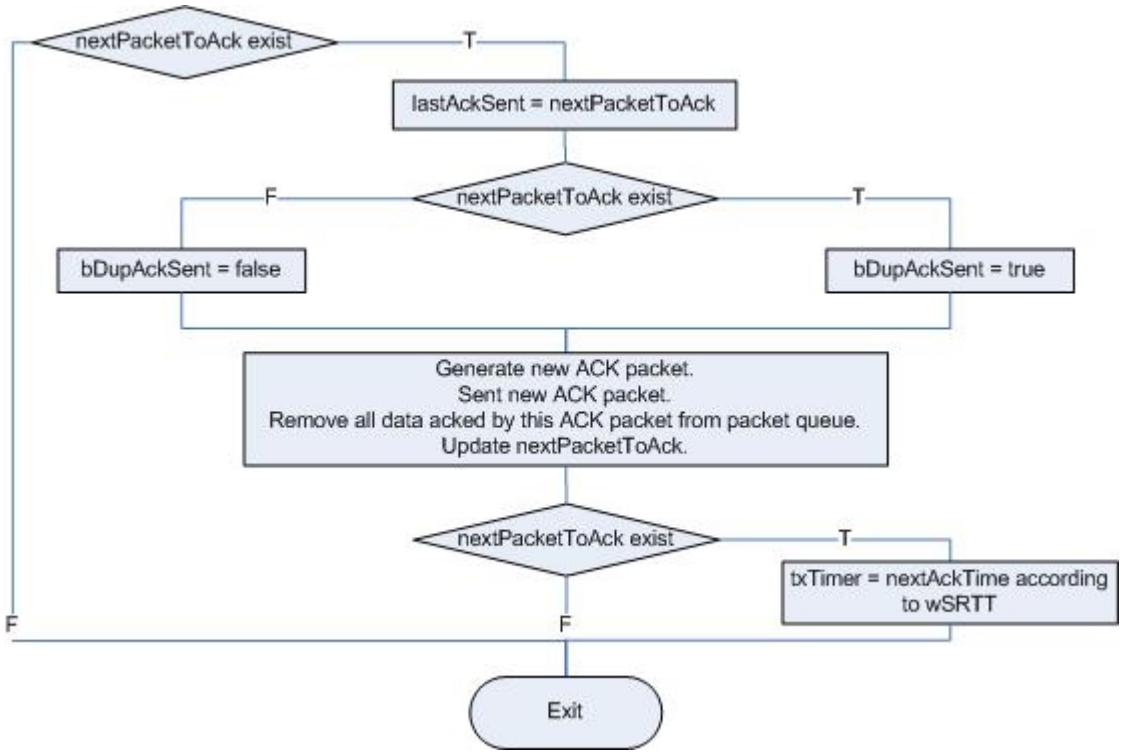


Implementation: receiving ACK

- What it does:
  - filters duplicate ACKs
  - redefines duplicate ACK threshold to detect packet losses
  - estimates wireless domain response time, and use it to pace the ACK return rate

Implementation: sending ACK

- When txTimer expires: send ACK packet and reschedule next ACK packet to send



Proposed algorithm: spurious timeouts

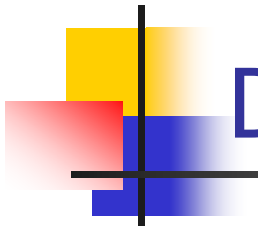
- Idea: prevent chain reaction caused by spurious timeouts
  - unnecessary packet retransmission
  - spurious fast transmission
- For every data packet received from FH
  - case 1: the packet is new
  - case 2: packet is a retransmission but has not been ACKed
    - handle it with the algorithm described before

Proposed algorithm: spurious timeouts

- case 3: packet is a retransmission and has been ACKed
  - store the last ACK sent to FH (lastAckSent)
  - compare lastAckSent with the seqno of packet received
  - if packet is acked, drop the packet: avoids additional DUPACKs, reduces response time and saves wireless bandwidth
  - (to consider the case when ACK packet is really lost on wireline link, send an ACK from BS instead, but do not send 3 consecutive ACKs to trigger spurious fast retransmission)

Design approach

- Why TCP option, as opposed to end-to-end solution?
  - TCP is the most successful and most tested transport protocol
  - wireless links require services similar to those provided by TCP in wireline links
  - any new protocol would require thorough validation and would face difficulties of deployment in the existing network
- Introduced ACK "queue" in BS does not require memory storage

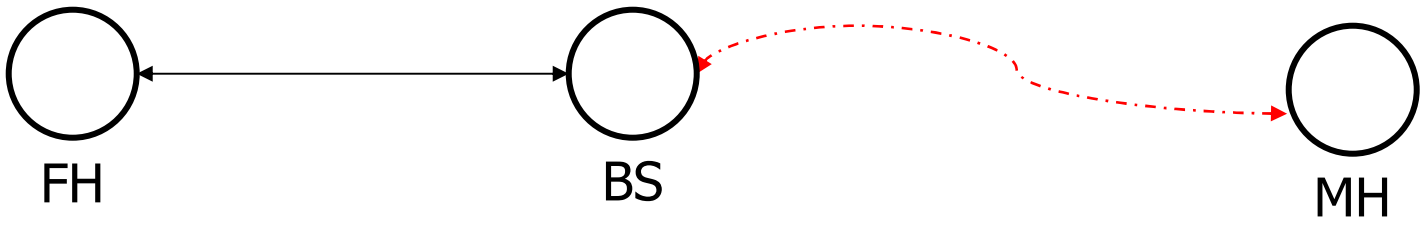


Design approach

- Proposed design does not have the drawback during handoff operations
- Example:
  - Snoop requires implementation of a data packet queue in BS, which needs to be transferred to the next BS during handoffs
  - this additional memory transfer increases handoff time

Simulation setup

- Simulator: ns-2.26 (ns-2.1b10 for old version number)
- OS: RedHat Linux 9
- Network setup

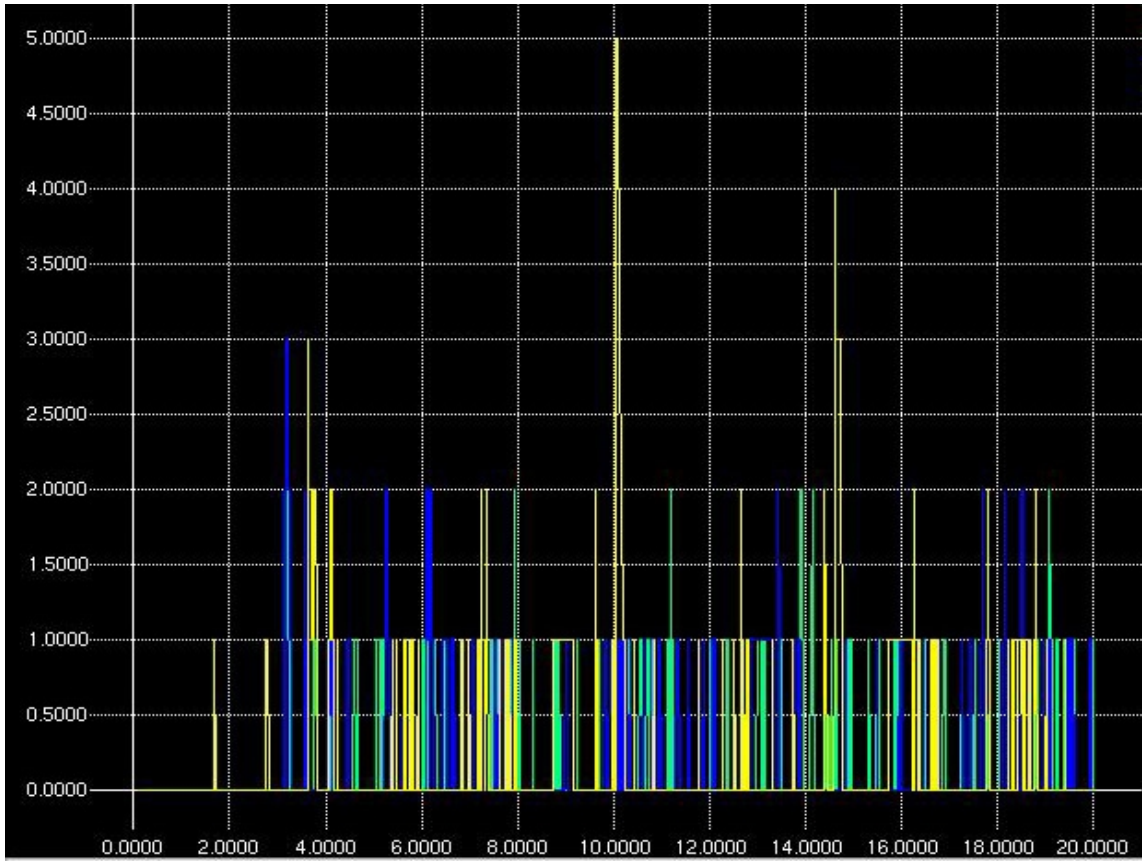


Scenario 1: delay variation

- Scenario: delay variation with small delay
  - FTP data is sent from FH to MH for 20sec
  - TCP Reno is used
  - TCP data packet size: 1040 bytes (default in ns-2)
  - link FH-BS:
    - link capacity: 250Kbps, 5ms delay
    - queue: DropTail
  - link BS-MH:
    - link capacity: 250Kbps, around 190ms of variable delay, delay variation simulated since time 0.5sec
    - queue: DropTail

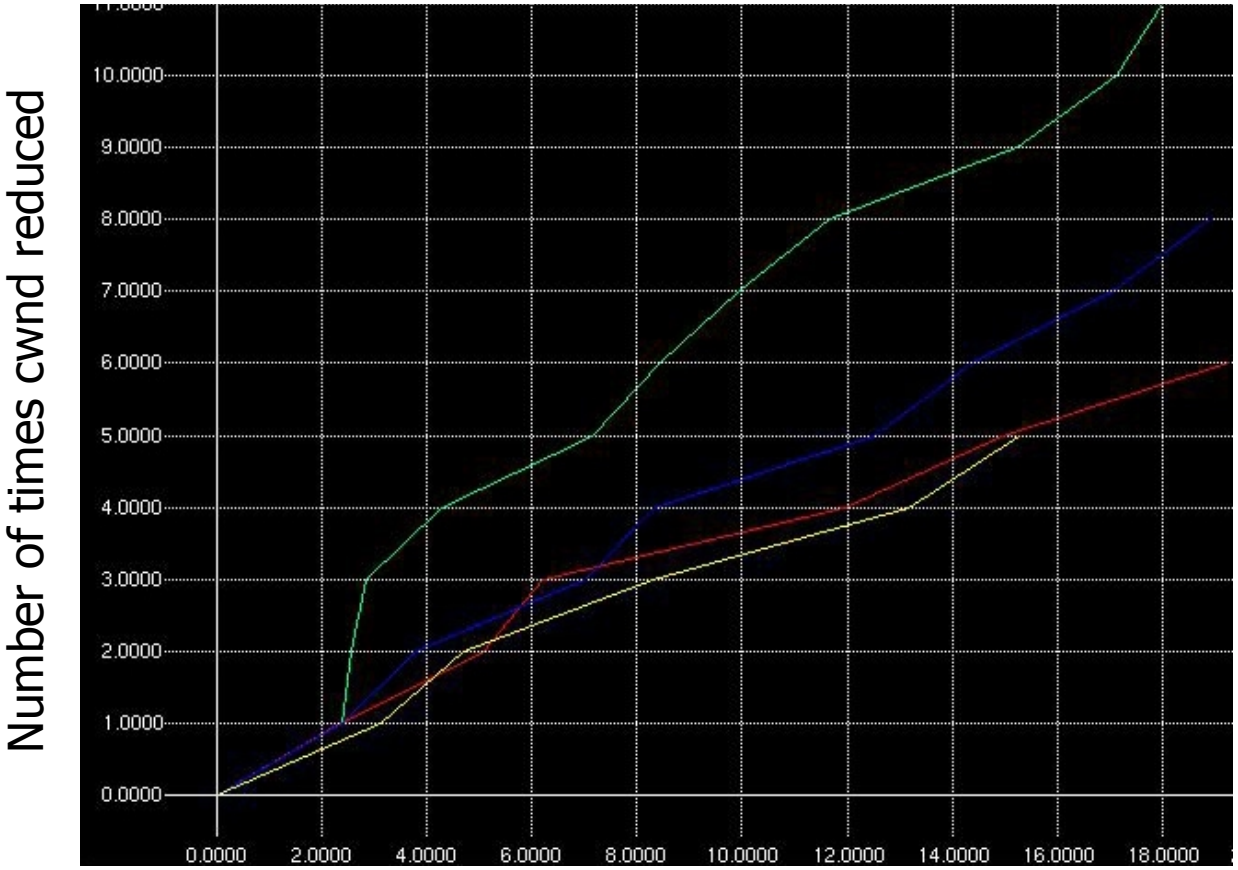
ns-2 simulations: ACK queue

- Larger dupAckThresh results in larger ACK queue



ns-2 simulations: cwnd reductions

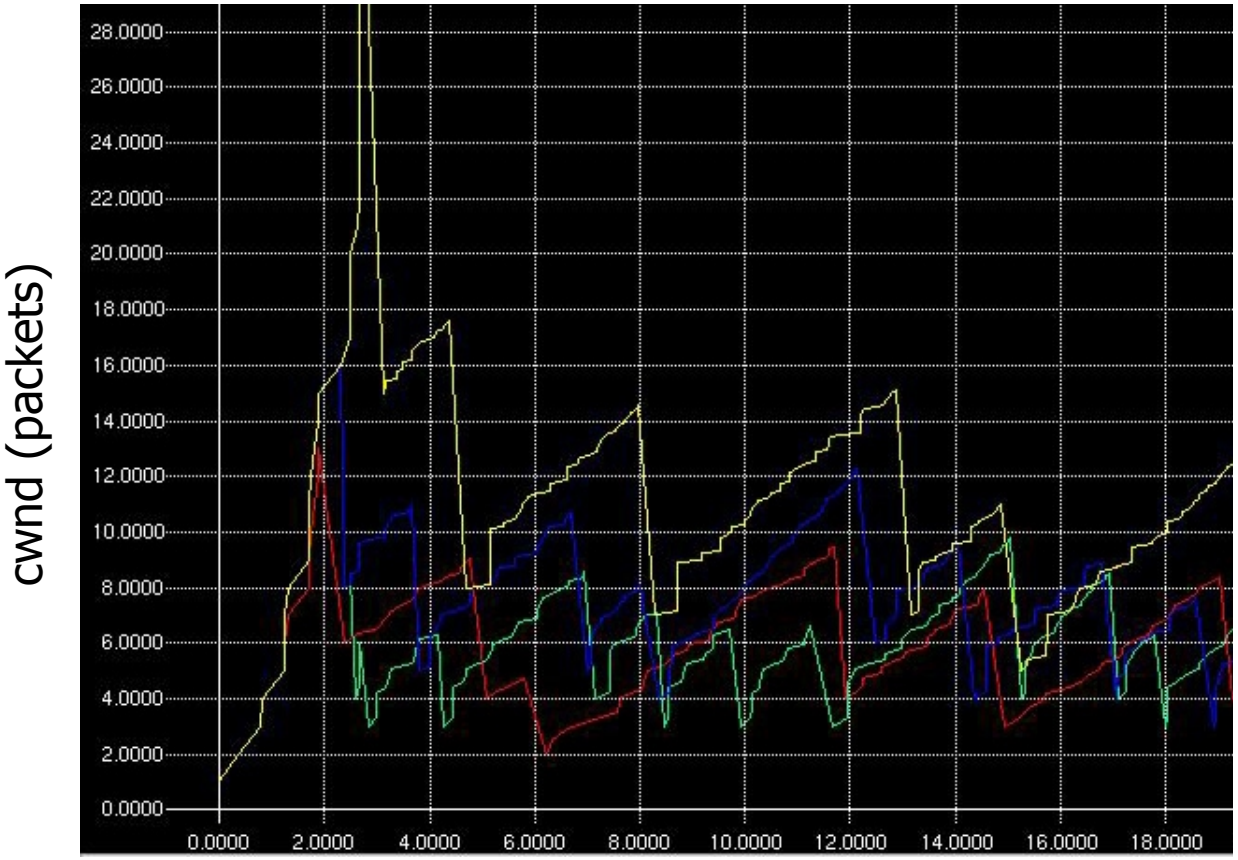
- Larger dupAckThresh results in fewer cwnd reductions



No ACK control
dupAckThresh = 3
dupAckThresh = 5
dupAckThresh = 8

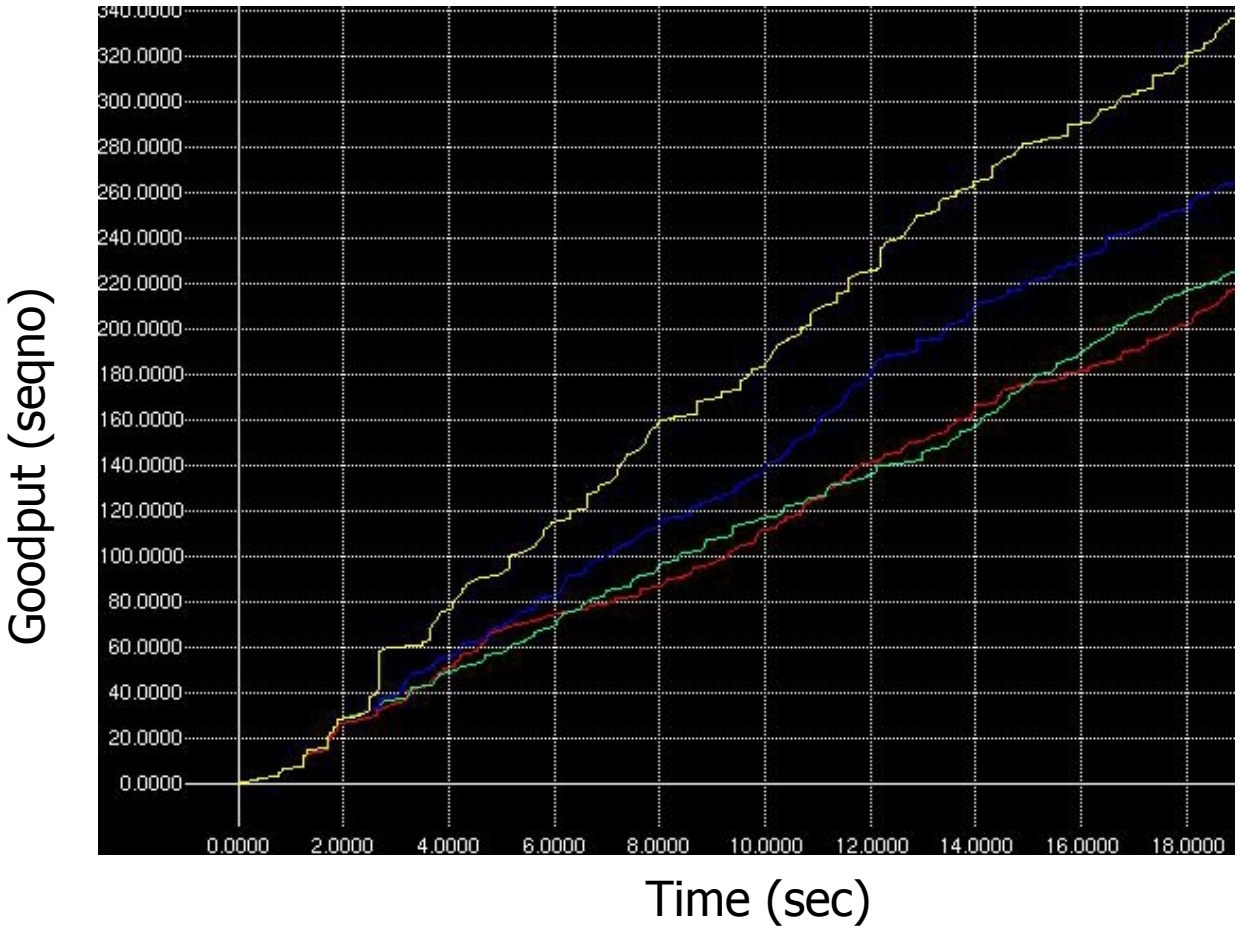
ns-2 simulations: cwnd

- Larger dupAckThresh results in higher cwnd



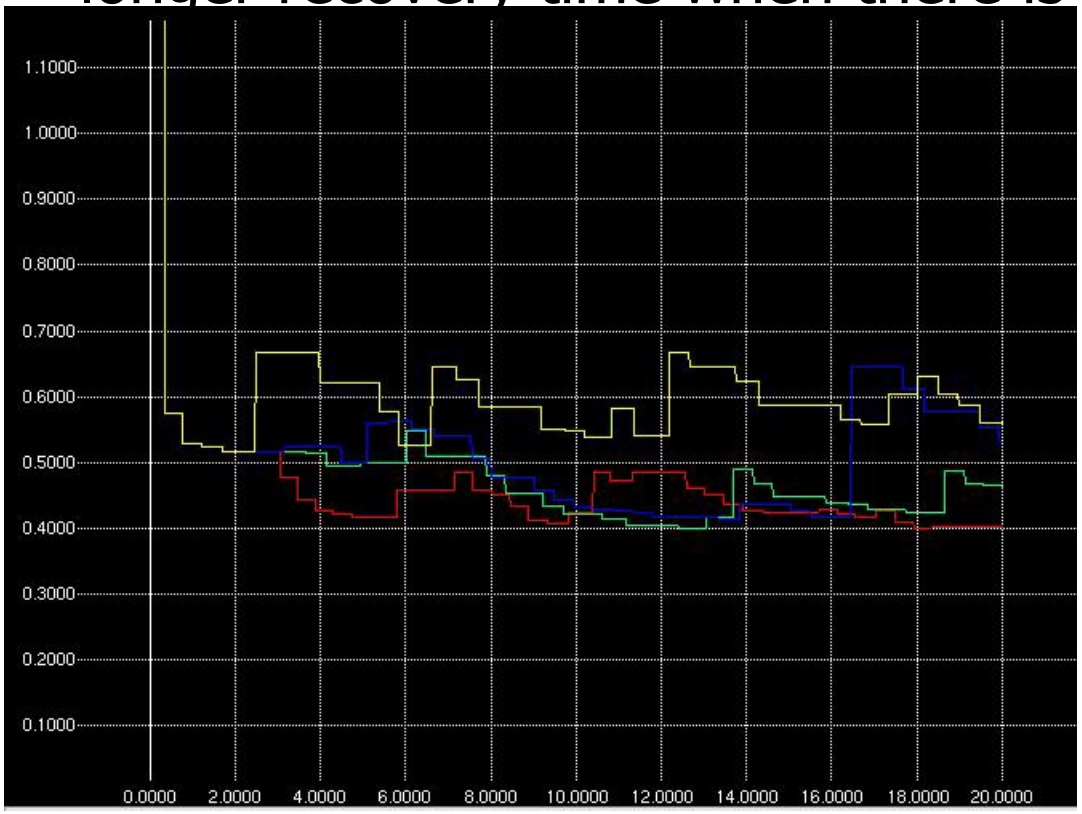
ns-2 simulations: goodput

- Larger dupAckThresh results in higher goodput



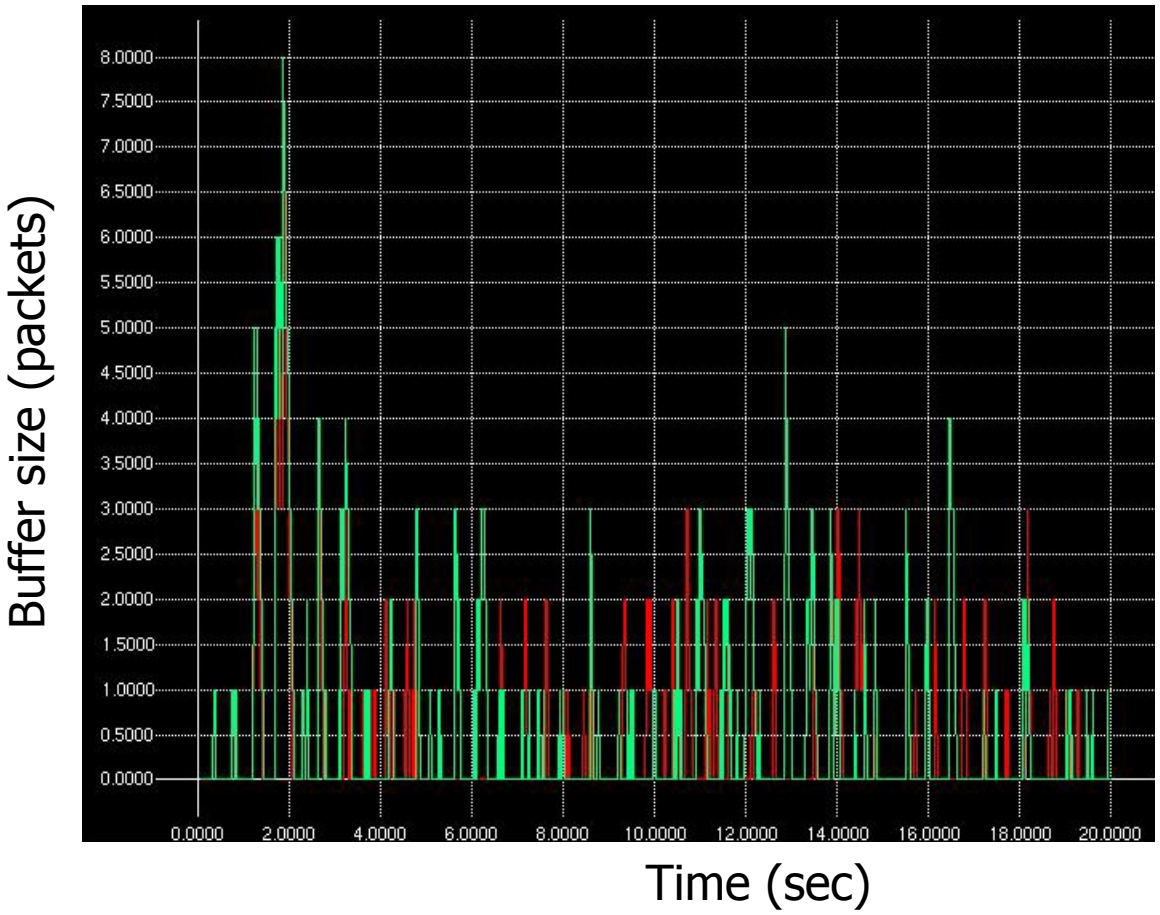
Discussion: RTO (ns-2 simulation)

- Larger dupAckThresh -> larger RTO -> possible longer recovery time when there is timeouts



Discussion: data buffer size (ns-2 simulation)

- Larger dupAckThresh -> larger buffer required



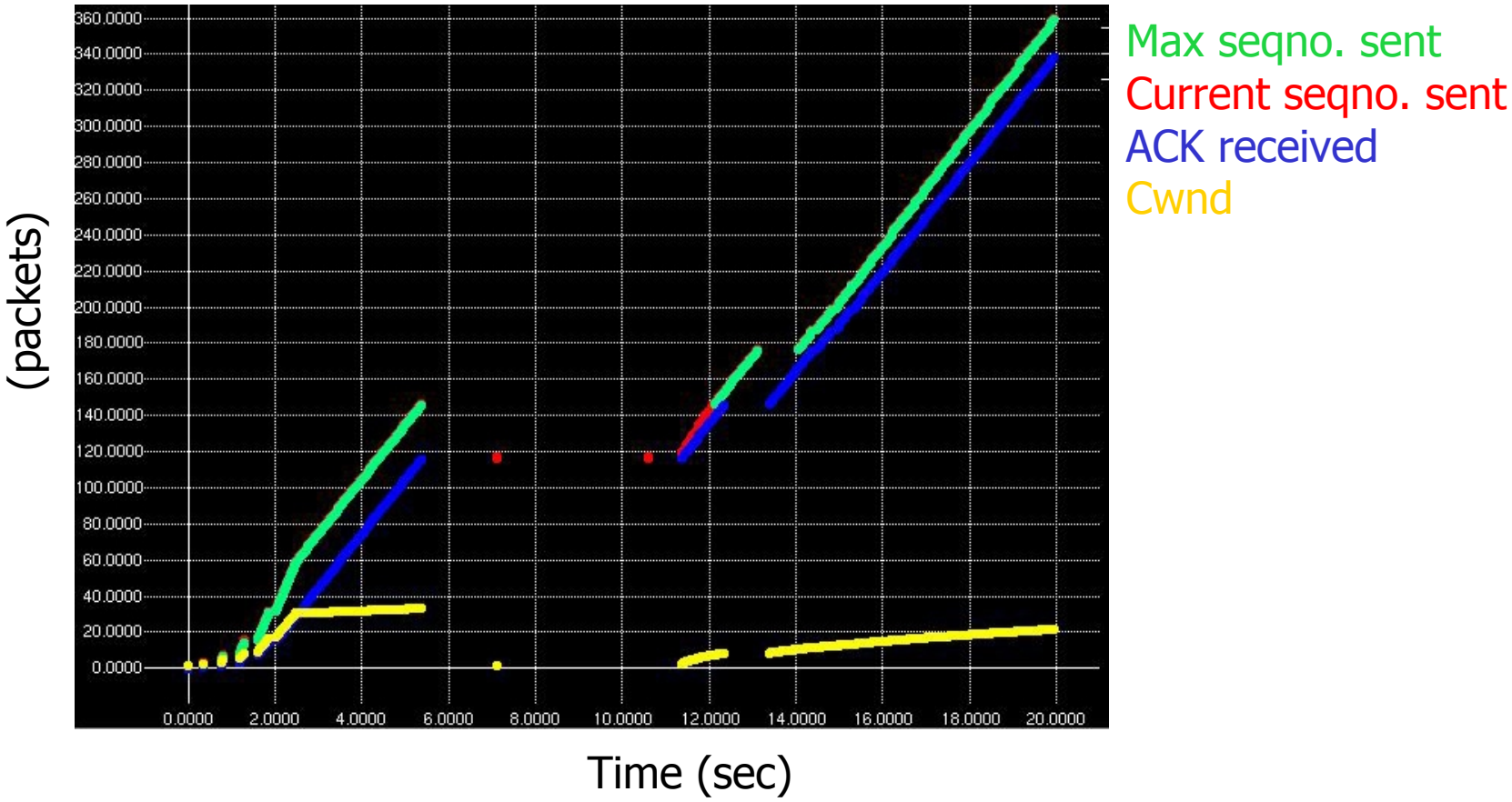
No ACK control
dupAckThresh = 8

Scenario 2: spurious timeouts

- Scenario: sudden large delay
  - Same configurations as in Scenario 1, but with a large delay at time 5 sec. Delay lasts for 6 sec.
- Purpose: to investigate how TCP react to sudden large delay increase and how ACK Controller may help

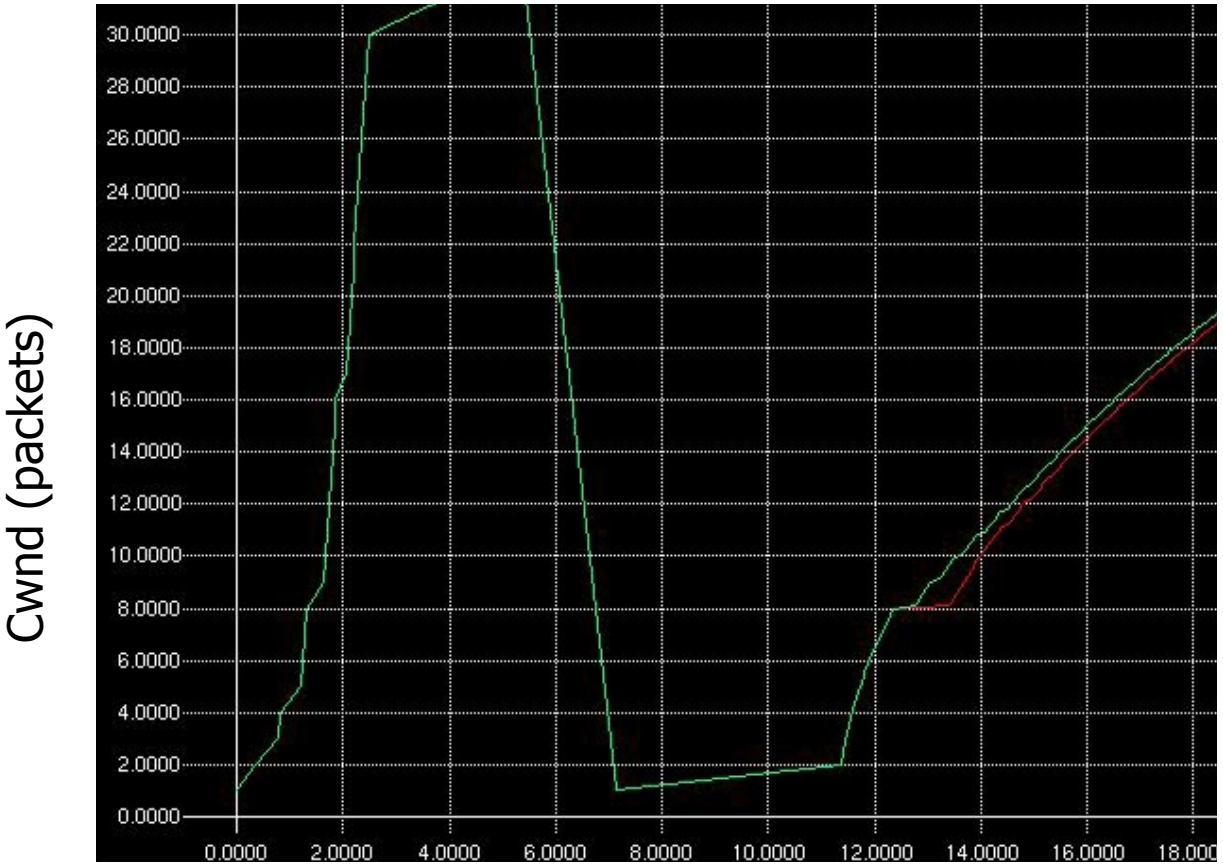
ns-2 simulation: SeqNo, ACK, and Cwnd

- Large sudden delay -> Spurious timeout -> retransmission -> duplicate ACKs (spurious fast retransmit) -> longer delay



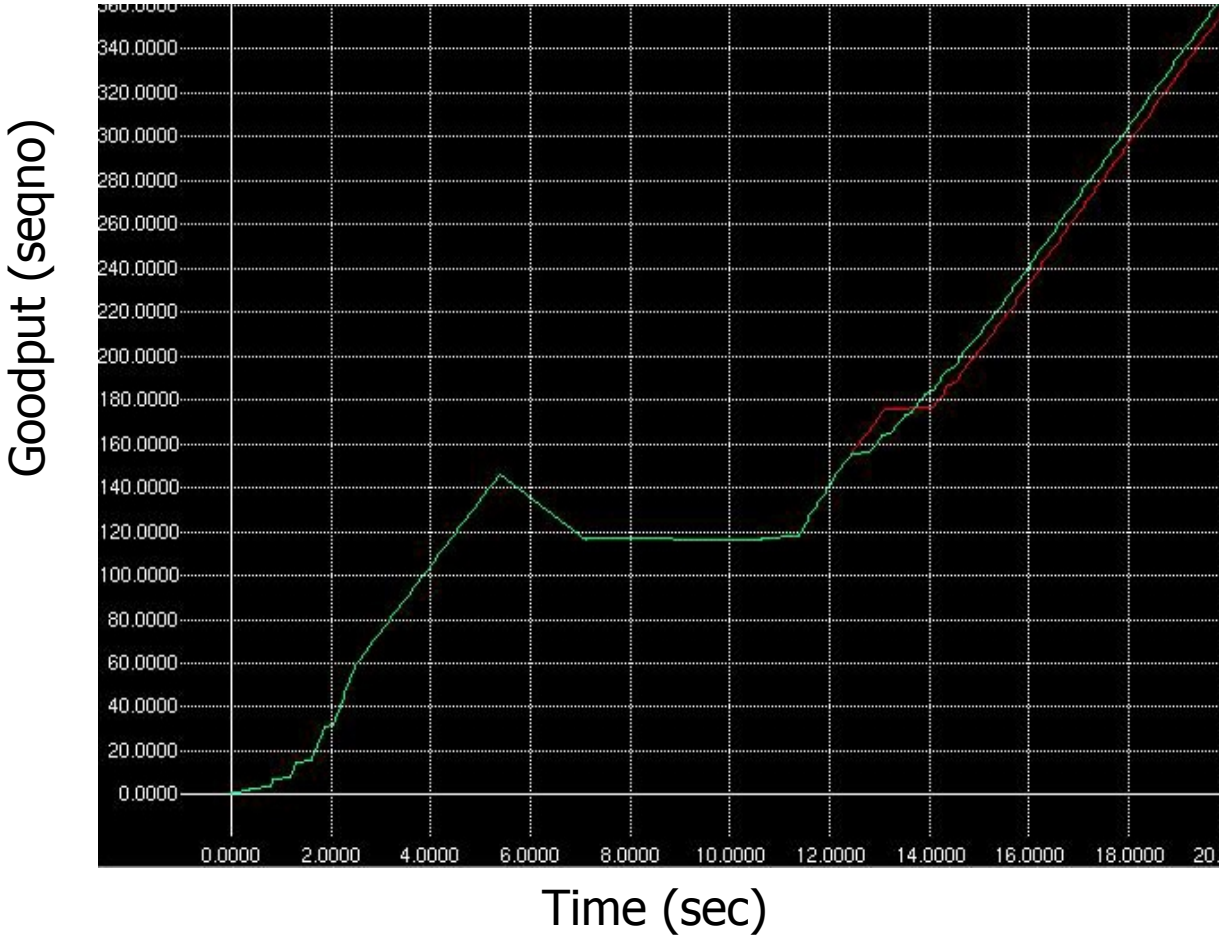
ns-2 simulation: Cwnd

- With ACK Control, higher cwnd is achieved.



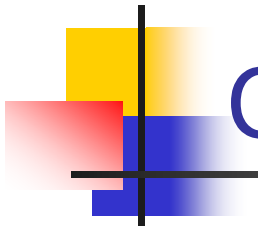
ns-2 simulation: SeqNo

- With ACK Control, higher goodput is achieved.



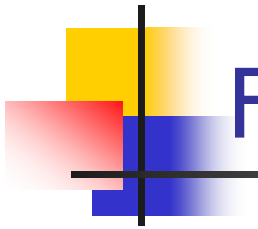
Conclusions

- Contribution: proposed ACK controller successfully reduces spurious fast retransmit and spurious timeouts caused by delay variation and large sudden delay in wireless link
- TCP goodput increased by over 60% in the used simulation scenario 1 (the exact improvement is highly depended on actual link characteristics)
- Chain reaction of spurious timeout is stopped and TCP performance is improved



Conclusions

- Proposed algorithm is easy to implement and deploy
  - modifications are required only in BS
- Proposed algorithm does not change TCP's congestion control semantics for end users
  - it should not pose restriction on future TCP evolution
- Has no drawbacks for handoffs
- Increased RTO may be of concern



Future work

- ACK compression issues
- Simulation: more accurate traffic delay generator
- Short-connections:
  - web traffic, short messaging service
  - packet loss usually results in timeouts
  - considerable costs to recover

Future work

- Consider short-connections:
  - approach: classify short connections and use wireless link information collected in BS to detect timeouts and use 3-dup ACKs to trigger retransmission
    - Cwnd reduction has little effect
    - retransmission of packets for short connections has little effect on network utilization

References

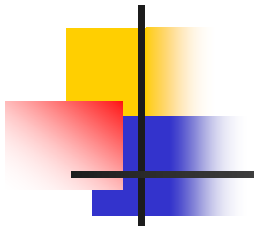
- A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proceeding of 15th International Conference on Distributed Computing Systems (ICDCS)*, pp. 136-143, May 1995.
- A. Gurtov, "Effect of Delays on TCP Performance," in *Proceedings of IFIP Personal Wireless Communications (PWC'01)*, August 2001.
- J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby "Performance enhancing proxies," Internet Draft: http://community.roxen.com/developers/idocs/drafts/draft-ietf-pilc-pep-04.html (accessed June 2003).
- K. Brown and S. Singh, "A network architecture for mobile computing," in *Proceeding of IEEE INFOCOMM*, pp. 1388-1396 , San Francisco, CA, March 1996.
- K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 5, pp. 19-42, October 1997.
- K. Xu, Y. Tian and N. Ansari, "TCP-Jersey for wireless IP communications,", *IEEE Journal on Selected Areas in Communications*, pp. 747-756, vol. 22, issue 4, May 2004.
- M. C. Chan and R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation," *ACM SIGMOBILE*, pp. 71-82, 2002.
- D. Chandra, R. Harris, and N. Shenoy, "TCP Performance for future IP-based wireless networks," Royal Melbourne Institute of Technology and Rochester Institute of Technology.

References

- H. Elaarag, "Improving TCP Performance over Mobile Networks," *Journal of ACM Computing Surveys*, vol. 34, no. 3, pp.357-374, September 2002.
- S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control," *IEEE Communications*, April 2001.
- H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, "TCP over second (2.5G) and third (3G) generation wireless networks," RFC-3481, February 2003.
- V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM*, 1988.
- R. Ludwig and R. H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmission," *ACM Computer Communications Review*, vol. 30, no. 1, pp. 30–36, January 2000.
- S. Karandikar, S. Kalyanaraman, P. Bagal, and Bob Packer, "TCP Rate Control," *ACM SIGCOMM Computer Communication Review*, vol. 30, no., January 2000.
- S. Fu and W. Ivancic, "Effect of Delay Spike on SCTP, TCP Reno, and Eifel in a WirelessMobile Environment," *International Conference on Computer Communications and Networks*, Miami, FL, Oct. 14-16, 2002, pp. 575-578.
- K. Luo, and A. Fapojuwo, "Impact of terminal mobility on TCP congestion control performance," University of Calgary, 2003.
- S. Singh, "Quality of service guarantees in mobile computing," *Journal of Computer Communications*, vol. 19, no. 4, pp. 359-371, April 1996.

References

- S. Singh, "Quality of service guarantees in mobile computing," *Journal of Computer Communications*, vol. 19, no. 4, pp. 359-371, April 1996.
- P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan, "WTCP: A reliable transport protocol for wireless wide-area networks," *Proceedings of ACM Mobicon'99*, Seattle, WA, pp. 231-241, 1999.
- W. Stevens, *TCP Illustrated*, Volume 1. Reading, MA: Addison-Wesley, Professional Computing Series, 1984.
- I. Stojmenovic, *Handbook of wireless networks and mobile computing*, John Wiley & Sons, New York, 2002.
- V. Tsaoussidis, and I. Matta, "Open issues on TCP for mobile computing," *The Journal of Wireless Communications and Mobile Computing*, John Wiley & Sons, vol. 2, issue. 1, February 2002
- OPNET documentation V.8.0.B, OPNET Technologies Inc., Washington DC.



Thank You!