DualRTT: Enhancing TCP Performance During Delay Spikes

Mohammed Atiquzzaman, Ph.D.
School of Computer Science
University of Oklahoma.

Email: atiq@ieee.org
Web: www.cs.ou.edu/~atiq

Presentation at
Tohoku University, Sendai, Japan.
Aug 6, 2002.

Problem Statement

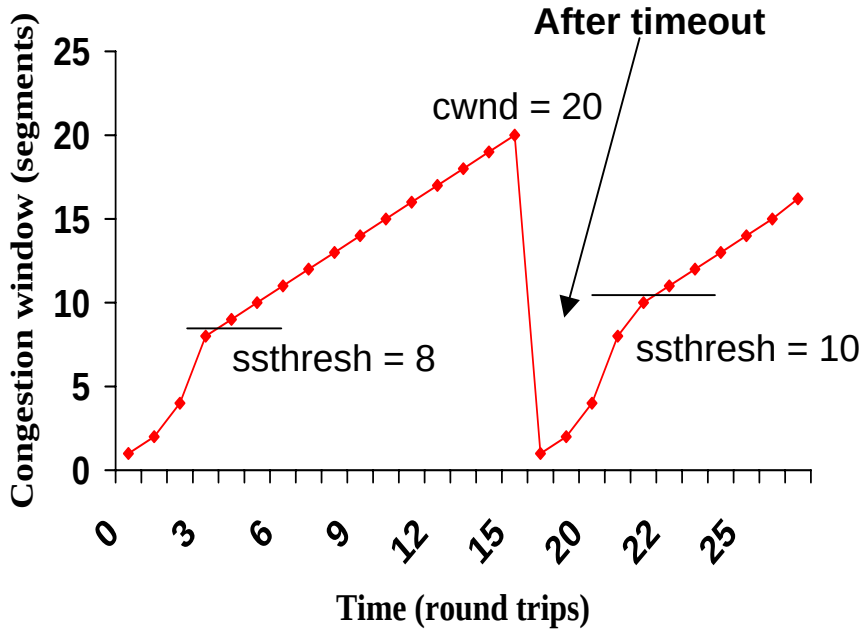
- TCP is a reliable transport protocol
  - Retransmits lost packets
- Non-receipt of acknowledgement within a certain time is treated as loss
  - Retransmission Timeout timer for measuring time
- Sudden long delays (delay spikes) confuse the timer and cause
  - unnecessary retransmission of packets and
  - throttling of packet transmission rate
- TCP throughput is seriously affected by delay spikes

Causes of Delay Spike

- Handoff between cells
  - Delays due to channel allocation
  - Physical disconnection
- RLC layer recovery of errors
  - Packet error is a sudden phenomenon
  - Long time for retransmission at link layer
- Preemption of data traffic
  - Sudden arrival of voice call

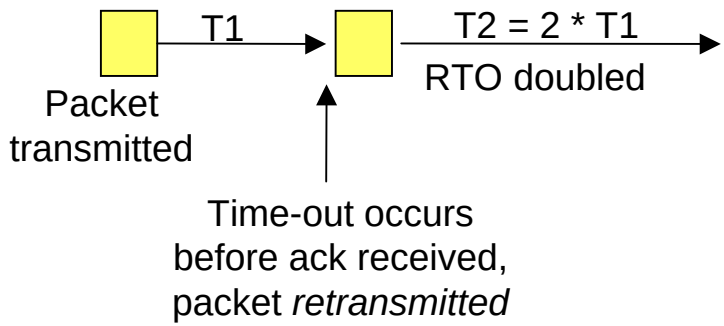
TCP congestion control

- Slow Start
  - *cwnd* reduced to one on timeout.
- Fast Retransmit
  - Three DupACKS
  - *cwnd* reduced to half.



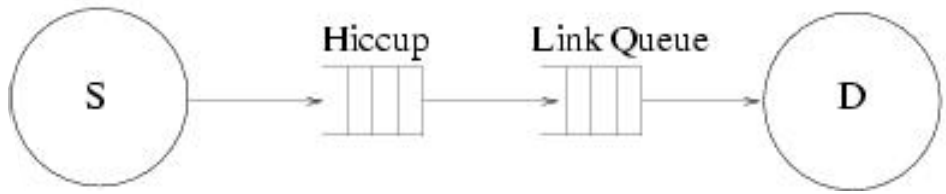
TCP error recovery

- TCP timer mechanism
  - Packets are retransmitted on RTO expiry
  - Double RTO on every timeout
- RTO updates are slow
  - RTO can not adapt to sudden changes of RTT
- Karn's algorithm solves retransmission ambiguity
  - Ignore RTT measurements from retransmitted packets.



Effect of Delay Spikes on TCP

- Retransmission ambiguity
  - TCP sender unable to distinguish between the acknowledgement of original and retransmitted packet.
- Spurious Timeout and Spurious Retransmission

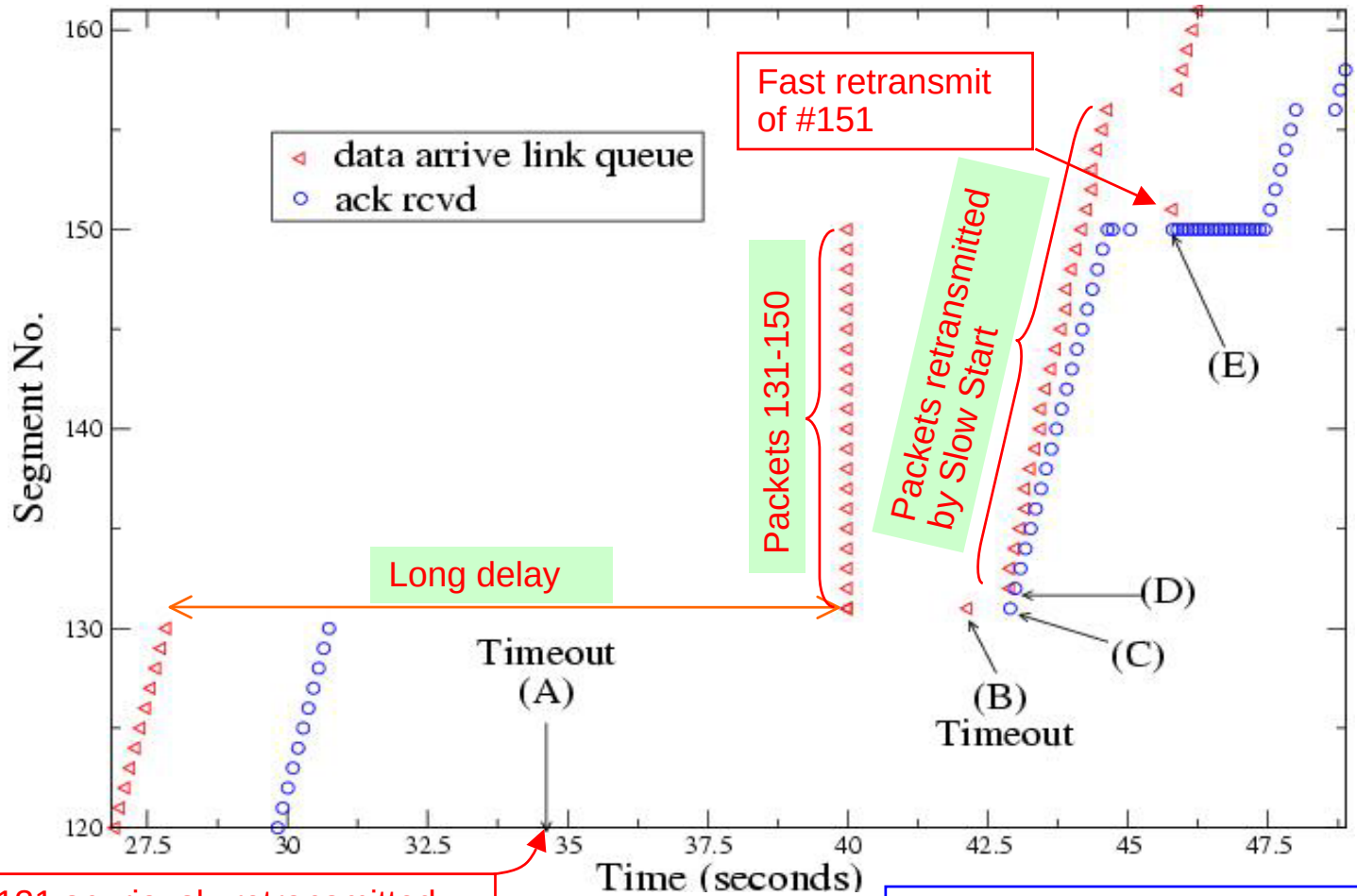


Spurious timeout: Timeout which would not have happened if the sender waited long enough.

Spurious Fast Retransmission: Occurs when segments get re-ordered beyond the DUPACK-threshold in the network before reaching the receiver

Link bandwidth = 46.8Kbps
Link delay=1.4s
Start of delay spike = 28s
Length of delay spike = 12s

Spurious Timeout and Retransmission



#131 spuriously retransmitted and delayed by delay spike

Segments received by receiver:
131, 132, … 150, 131, 132….,150,… 151

Congestion Window

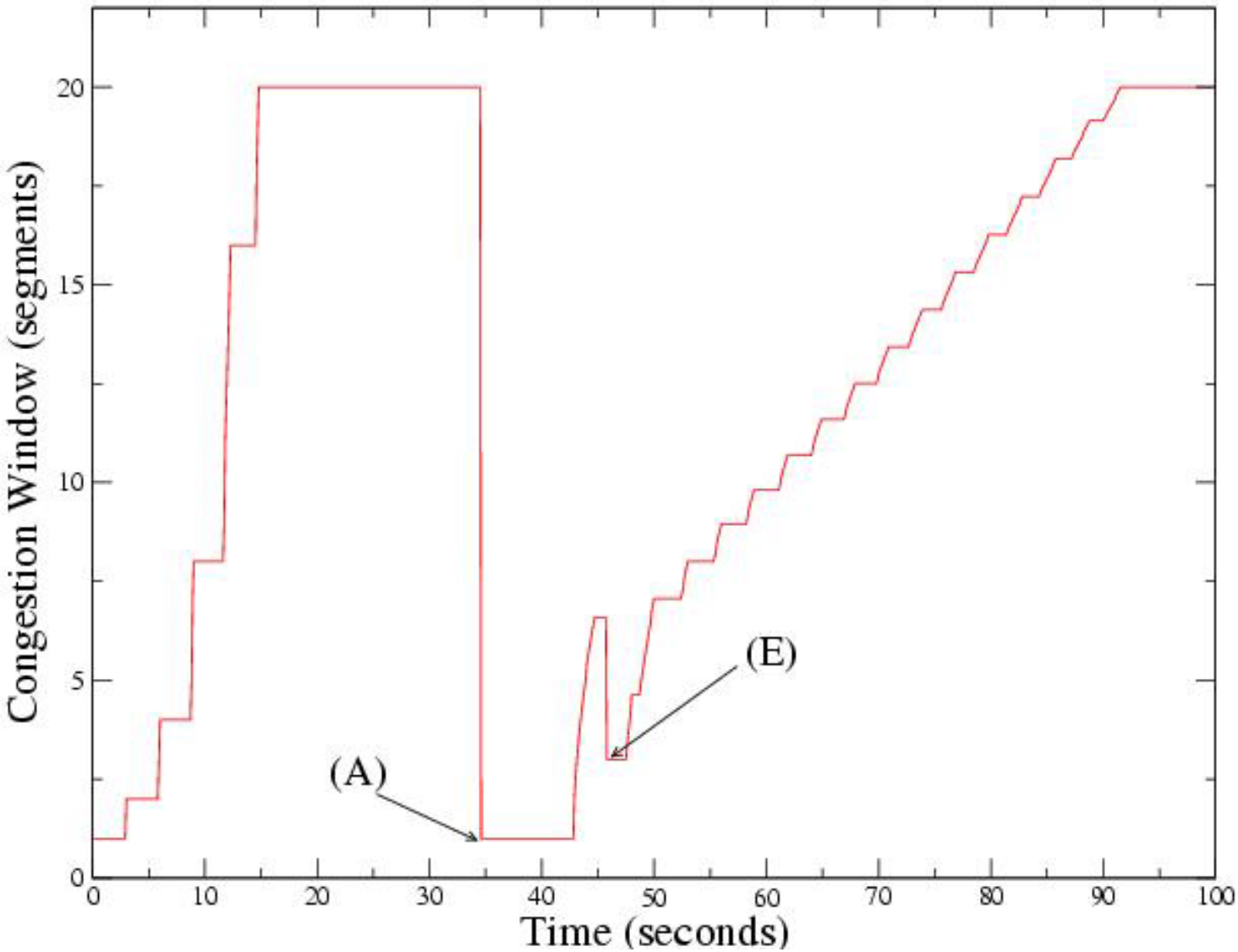


Illustration of *RTO* and *RTT*

Time	*RTO*	*RTT*
17.545	4.6	2.9
30.840	3.8	2.9
42.905	15.2	2.9
43.004	15.2	2.9
43.102	15.2	2.9
43.791	15.2	2.9
43.890	15.2	2.9
43.988	15.2	2.9
44.481	15.2	2.9
44.579	15.2	2.9
47.780	4.0	3.3
52.704	3.8	2.9

Karn's Algorithm: RTT measurements from retransmitted segments are ineligible for updating *RTO*

- Main idea of Eifel:
  - TCP timestamp option to solve retransmission ambiguity
  - Every packet is timestamped (12-bytes of header)
  - If the ACK's timestamp is "smaller" than sending time, previous timeout and retransmissions were spurious.
  - ***Action*** on detection of spurious timeout***:***
    - *cwnd* and *ssthresh* are restored to their values before spurious timeout.
    - Sender sends new packets instead of going thru go-back-N.
- Advantages:
  - Enhances TCP throughput during delay spikes.
- Disadvantages
  - **Deployment issues**: Requires receivers to implement timestamp option
    - Only 15% of Internet hosts implement timestamp
  - **Bandwidth wastage**: Extra header field
    - Not desirable in wireless networks, for which Eifel has been developed !!!

DualRTT Objectives

- Wireless friendly
  - Need to be bandwidth efficient
  - No extra header fields
- No change at receiver or Internet infrastructure
  - All changes should be at the sender side only

Principle of DualRTT

- Based on the dynamics of packet arrivals during delay spike
  - Segments are consecutively queued at the wireless sender during delay spikes
  - Segments arrive at the receiver back to back.
  - Interval between arrival of consecutively delayed packets is small
- Two new variables
  - *NRTT*: time between the "***most recent retransmission***" and the "***arrival of acknowledgement***" of the corresponding segment at the sender.
  - *MinRTT*: minimum value of RTT observed so far.
- Spurious timeout detection criteria:
  - *NRTT < Threshold * MinRTT*

Detection of Spurious Timeout by DualRTT

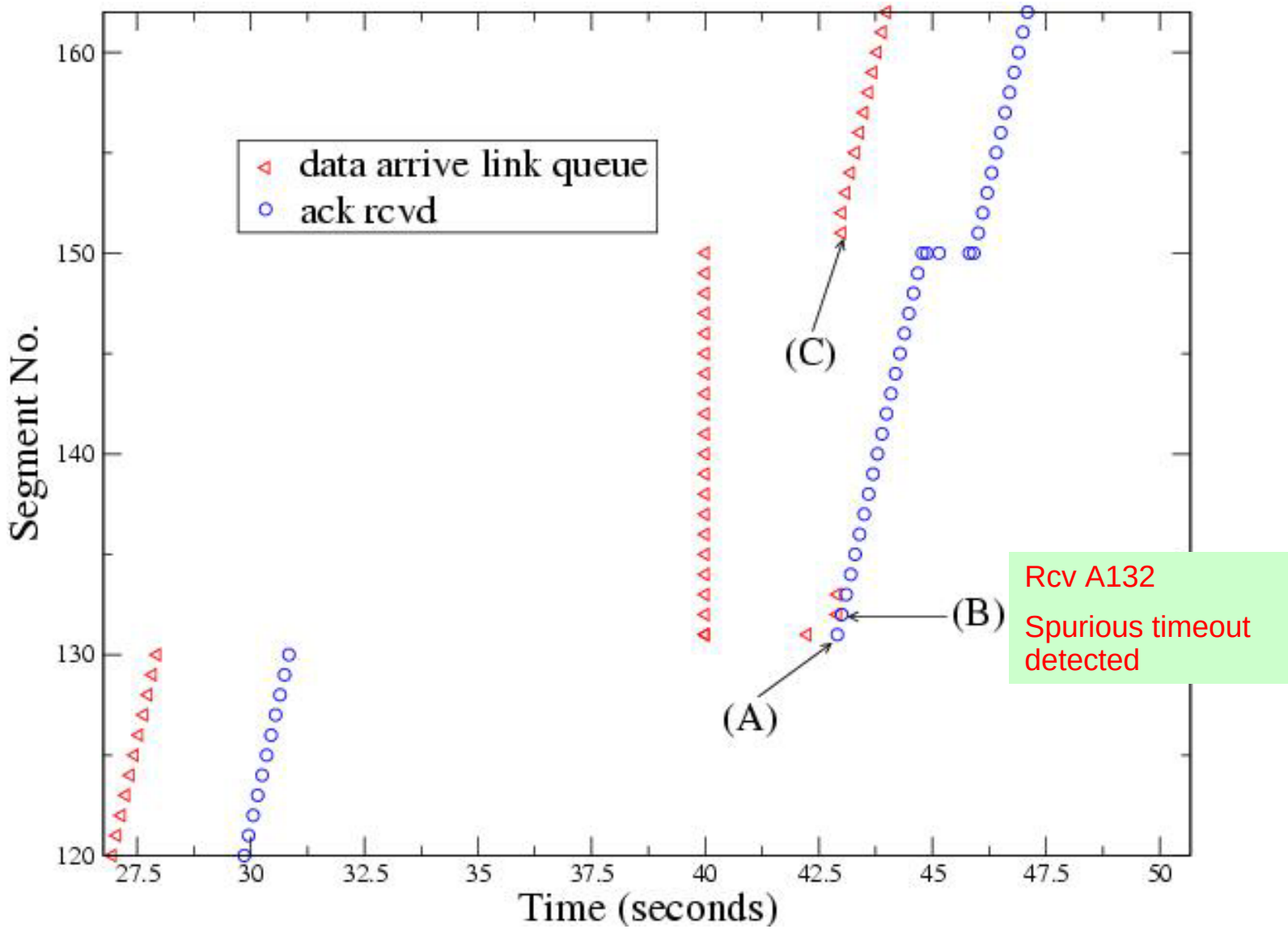


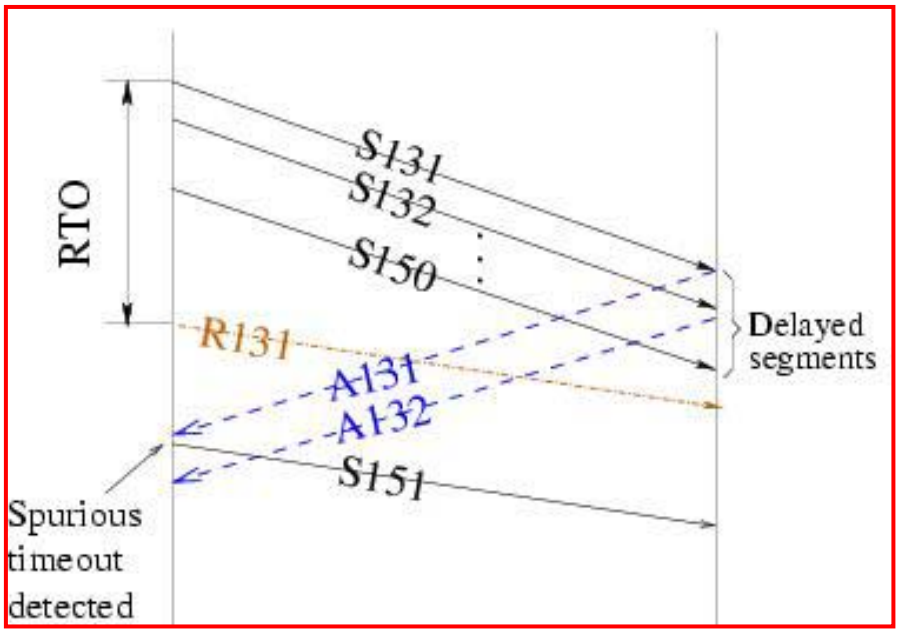
Illustration of *NRTT* and *MinRTT*

Time	*RTO*	*RTT*	*NRTT*	*MinRTT*	
17.545	4.6	2.9	2.9	2.8	
30.840	3.8	2.9	2.9	2.8	
42.905	15.2	2.9	13.9	2.8	Recv. A131, Snd S132, S133
43.004	15.2	2.9	0.1	2.8	Recv A132 Spurious timeout detected
43.102	15.2	2.9	0.1	2.8	
43.791	15.2	2.9	0.2	2.8	
43.890	15.2	2.9	0.3	2.8	
43.988	15.2	2.9	0.3	2.8	
44.481	15.2	2.9	0.4	2.8	
44.579	15.2	2.9	0.4	2.8	
47.780	4.0	3.3	3.3	2.8	
52.704	3.8	2.9	2.9	2.8	

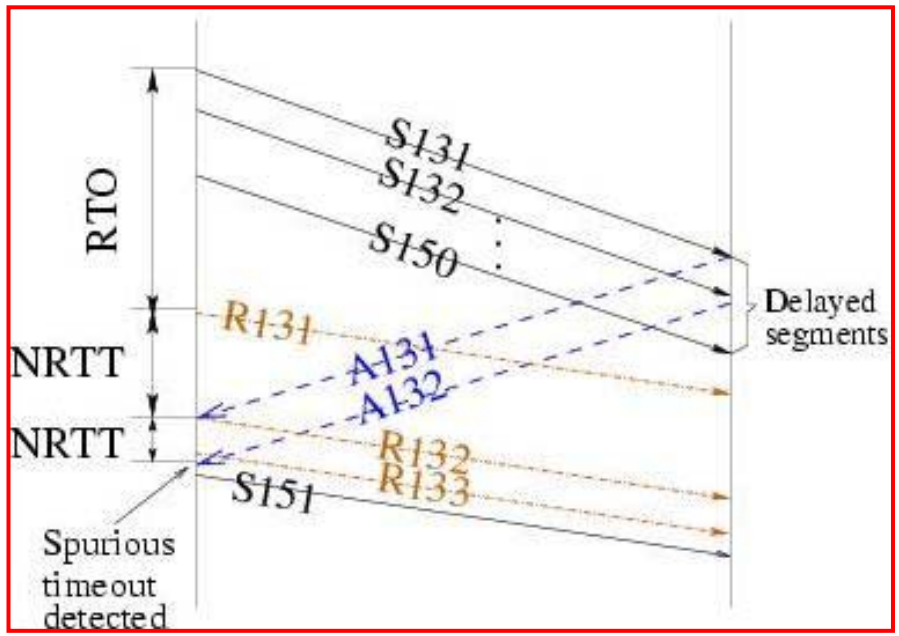
RTT measurements from retransmitted segments are ineligible for updating *RTO*

Eifel and DualRTT: Spurious Timeout Detection

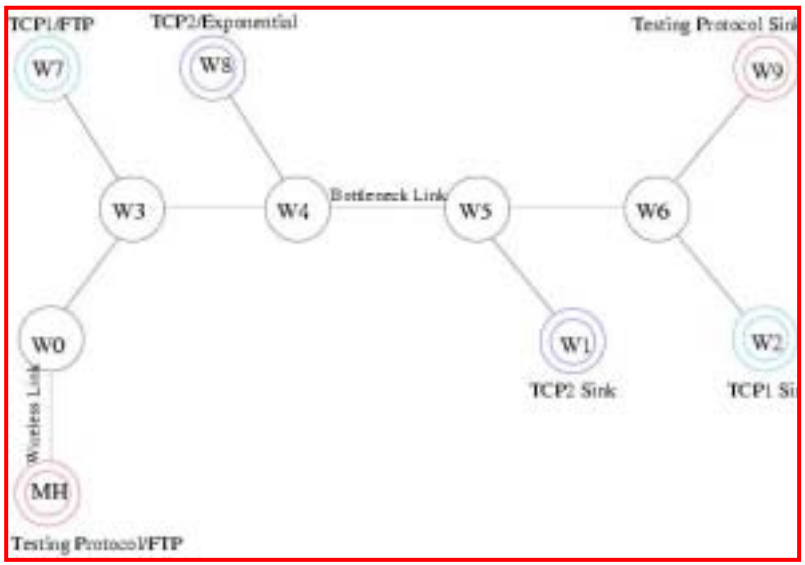
Eifel



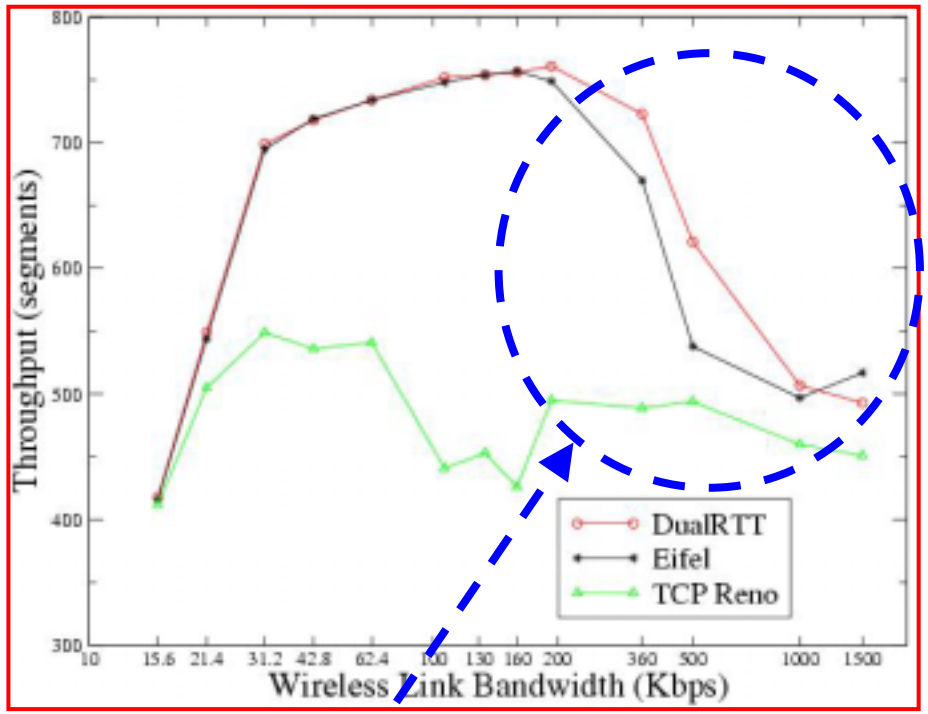
DualRTT



Performance of DualRTT



Links	Link BW (Kbps)	Prop. Delay (ms)
W0-W3, W3-W7, W5-W4 W4-W8 W5-W1, W6-W2, W6-W9	1500	200
MH-W0	15.6-1500	400
W4-W5	200-3500	200



Throughput drops due to ***congestion losses*** during delay spikes

Comparison: Throughput Improvement

Throughput = Number of packets received by the receiver during a period of time.

$$\textbf{Link Goodput} = \frac{\text{Size of payload}}{\text{Size of packet}}$$

Comparison of *Throughput*

Bandwidth (Mbps)	Eifel %	DualRTT %
0.2	33.30	35.66
0.5	45.56	47.82
1.0	34.44	34.46
1.5	36.15	36.24
3.5	36.17	36.20

Comparison of *Link Goodput*

Payload (Bytes)	Eifel	DualRTT	% Increase
8	0.133	0.167	**25.0**
16	0.235	0.286	21.4
32	0.381	0.444	16.7
64	0.552	0.615	11.5
128	0.711	0.762	7.1
256	0.831	0.865	4.1
512	0.908	0.928	2.2

Conclusions

- DualRTT enhances TCP throughput during delay spikes
- DualRTT
  - is wireless friendly
  - requires changes only at the sender → easy to deploy
  - does not require the receiver to implement the timestamp option
- Performance of DualRTT comparable or better than Eifel

- Acknowledgements
  - National Aeronatics and Space Administration (NASA)
  - Shaojian Fu

- Further Information

Dr. Mohammed Atiquzzaman

atiq@ou.edu, (405) 325 8077

- These slides are available at

www.cs.ou.edu/~atiq