TCP HACK: TCP Header Checksum Option to improve Performance over Lossy Links

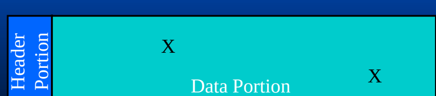
R. K. Balan, B. P. Lee,
K. R. R. Kumar, L. Jacob,
W. K. G. Seah, A. L. Ananda
Centre for Internet Research
School of Computing
National University of Singapore

Problem

- Lossy / wireless links are common
- TCP performs poorly when corruption occurs
- No distinction between corruption and congestion
  - Reduces sending rate, timeouts and slow start
  - Wrong behaviour !!
- Correct behaviour
  - Send multiple copies of packet
  - Keep sending rate the same

Key Observation

- Data portion usually **much larger** than header portion
  - ⇨Corruptions far more likely in data portion
- Packets with corrupted headers unlikely to reach destination



Our Solution

- Corrupted packets may still contain valid headers
- We recover that information
  - Better than throwing the packet away after it has done so much work!!
- Header information used to generate “special” ACKs
- Performs much better than SACK!!
- Orthogonal to other methods

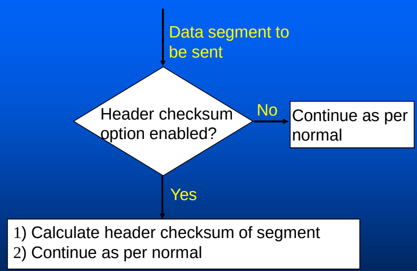
Outline of Talk

- Algorithm
- Experimental Setup / Error Model
- Experimental Results
- Potential Deficiencies
- Conclusion

Algorithm

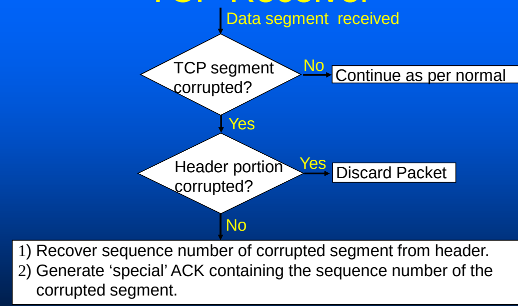
- Add an extra option to every TCP packet
  - Contains checksum for just the header
- On detecting a corrupted packet
  - Checks if header checksum is okay
  - If it is, send a special ACK to sender containing sequence number of corrupted packet
- On receiving a special ACK
  - Retransmit corrupted packet
  - Do not half congestion window

TCP Sender



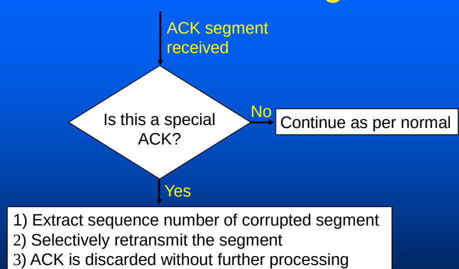
Modifications to the TCP sender

TCP Receiver



Modifications to the TCP receiver

ACK Processing

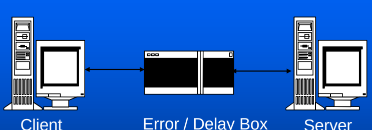


Modification to the ACK processing

Experimental Setup

- Linux 2.2.10 kernel
- Test bed was set up comprising of 3 machines
- All experiments were run at 10 Mb/s
- Iperf was used to generate TCP bulk traffic

Experimental Test bed



- Client and server were running TCP HACK
- Error / delay box used to simulate latencies and lossy links using modified rshaper kernel module

Error Model

- Packet corruption percentages of 2%, 5% and 10%
  - Single packet corruption
  - Burst corruption with burst lengths of 2, 5 and 10 packets
- We corrupted the data packets in 2 different ways
  - In the 1st way, ≈ 95% of the headers were corrupted
  - In the 2nd way, 0% of the headers were corrupted
  - True header corruption probability is somewhere in between

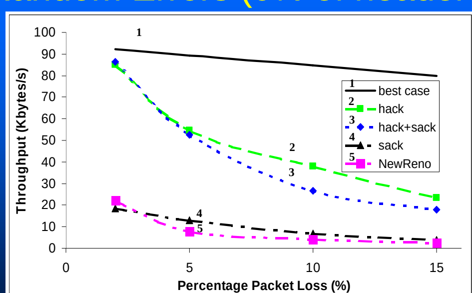
Testing Methodology

- TCP HACK compared with TCP NewReno and TCP SACK
- 2 different latencies
  - Short (10ms)
  - Long (300ms)
- Send/receive windows set large enough

Experiment Sets

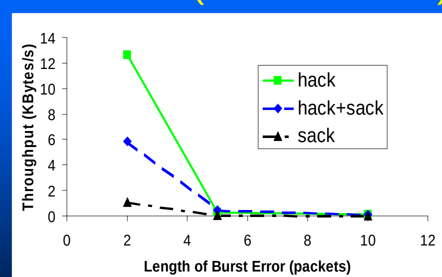
Header Corruption % / Error Type	0%	95 %
Random Errors (long and short latencies)	Next few slides	Results in Paper
Burst Errors (long latency)	Next few slides	Two Results, Rest in Paper

Random Errors (0% of headers)



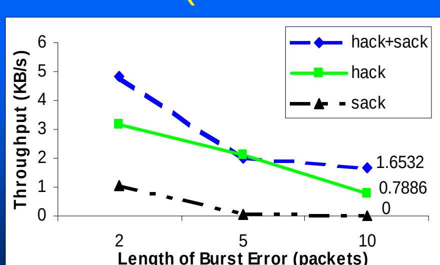
Long Latency (300ms) with 256KB of data transferred

Burst Errors (0% of headers)



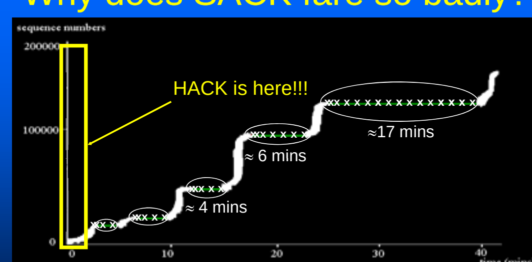
~ 20 - 70% packet corruption

Burst Errors (95% of headers)



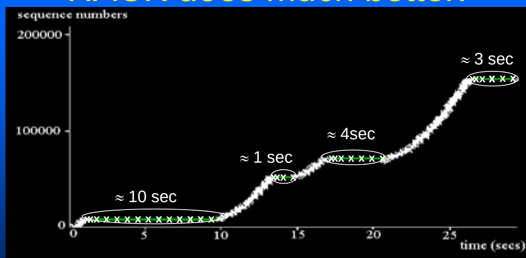
~ 20 - 70% packet corruption

Why does SACK fare so badly?



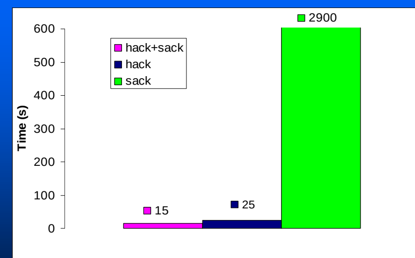
- Too many timeouts!!!!
- They are very long as well!

HACK does much better!



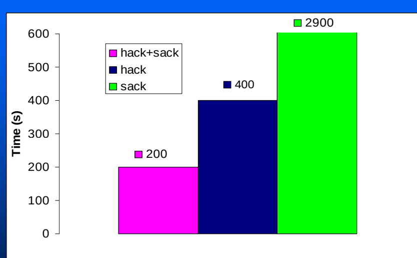
- Still some timeouts
- They are much shorter!!!! (≈100 times less)
- Adding SACK helps a bit

Time Taken (0% of headers)



HACK is 115x better than SACK!!!

Time Taken (95% of headers)



HACK is still 6x times better than SACK!

Experiment Summary

Header Corruption % / Error Type	0%	95 %
Random Errors (long and short latencies)	**5-10x** better than SACK	Equal to SACK
Burst Errors (long latency)	**100x** better than SACK	**6x** better than SACK

Does SACK help?

- Yes and No
- Fills in holes in the senders window
- Inefficiencies due to implementation
  - SACK may reduce cwnd as well
- SACK can co-exist very nicely with HACK
  - orthogonal in nature

Other Issues

- End-2-end protocol
  - Suitable for Ad-Hoc environment
  - No base station support required
- Sending corrupted packets to TCP is hard
- Link layer protocols can be efficient
  - **But**, they give no information to TCP
  - Spurious timeouts may occur as a result
  - RTT estimates can fluctuate as well

Future Work

- Test TCP HACK over a real lossy link
  - satellite link experiments are planned
- Compare TCP HACK with
  - Snoop, ECN etc.
  - Implement and test hybrid mechanism
  - TCP Hack with Snoop etc.
  - TCP Hack with link layer protocols etc.
- Determine the % of corrupted packets with intact headers on real lossy links

Conclusion

- Recovering header information can help
- TCP HACK does better than SACK under various error conditions
  - Up to a factor of 100 reduction in time taken to complete transfer!!!
- HACK is particularly useful under burst error conditions
  - Recovering even a small % of the headers helps dramatically

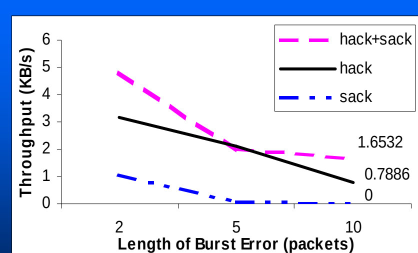
Thank You!

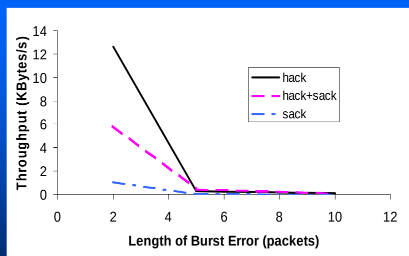
Header Corruption %

- Tested using old 2 Mbit Lucent Wavelan Cards
  - Direct sequenced
- Approximately 90-95% of the corrupted packets had intact headers under reasonable error rates
- UDP lite (Larzon, Degermark, Pink) reports that about 0.8% of normal UDP fail the checksum at the receiver

Effect of Window Size

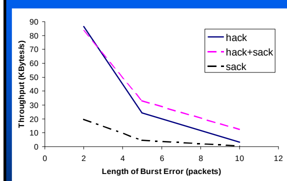
- Effect of different window sizes investigated
- 16KB and 64KB windows were used
- Results were similar



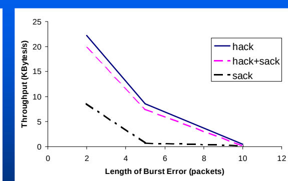


Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 31

Burst Errors (0%)



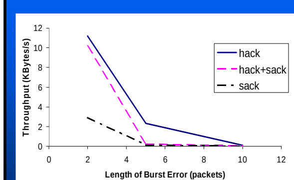
Throughput for **2% burst error** for various burst lengths



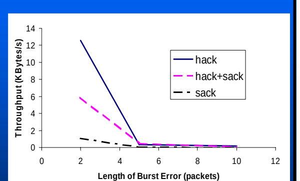
Throughput for **5% burst error** for various burst lengths

Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 32

Burst Errors (0%) (2)



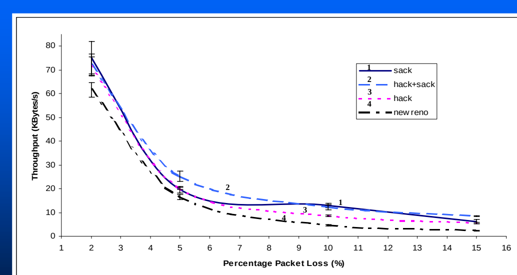
Throughput for **10% burst error** for various burst lengths



Throughput for **15% burst error** for various burst lengths

Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 33

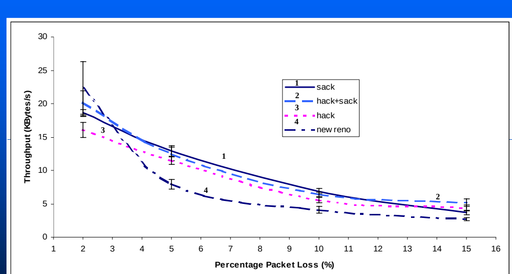
Low Latency Links (95%)



Throughput versus percentage packet loss for short latency (10 ms) link with random single packet errors

Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 34

High Latency Links (95%)



Throughput versus percentage packet loss for long latency (300 ms) link with random single packet errors

Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 35

Number of Slow Starts (95%)

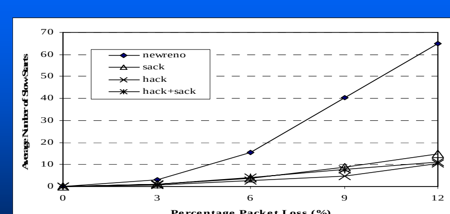
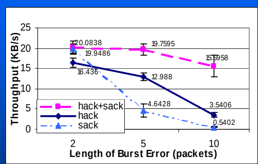


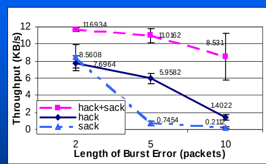
Fig. 7: Average number of slow-starts for long latency link with random single packet errors

Infocom 2001 (24/4/01) TCP HACK (rajesh@cs.cmu.edu) 36

Burst Errors (95%)

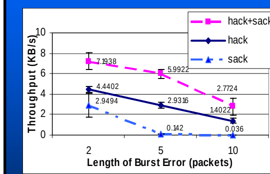


Throughput for **2% burst error** for various burst lengths

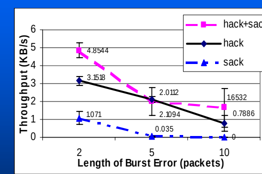


Throughput for **5% burst error** for various burst lengths

Burst Errors (95%) (2)

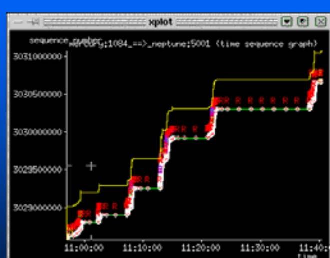


Throughput for **10% burst error** for various burst lengths



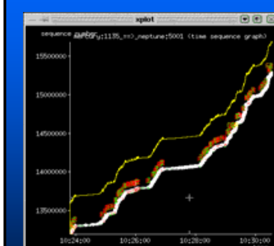
Throughput for **15% burst error** for various burst lengths

Time Sequence Graphs (95%)

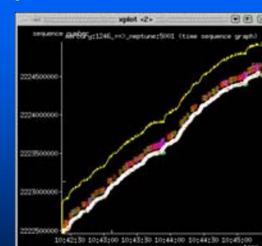


Time Sequence Graph for SACK (5% packet error rate with burst length of 5)

Time Sequence Graphs (95%) (2)

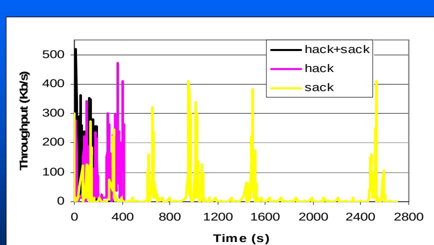


Time Sequence Graph for HACK (5% packet error rate with burst length of 5)



Time Sequence Graph for HACK+SACK (5% packet error rate with burst length of 5)

Throughput Versus Time (95%)



Throughput versus Time graph for various TCP implementations (5% packet error rate with burst length of 5)