Loss Tolerant TCP (LT-TCP): Implementation and Evaluation

Koushik Kar
Rensselaer Polytechnic Institute

Bishwaroop Ganguly
MIT Lincoln Labs

Nathan Hourt
Rensselaer Polytechnic Institute

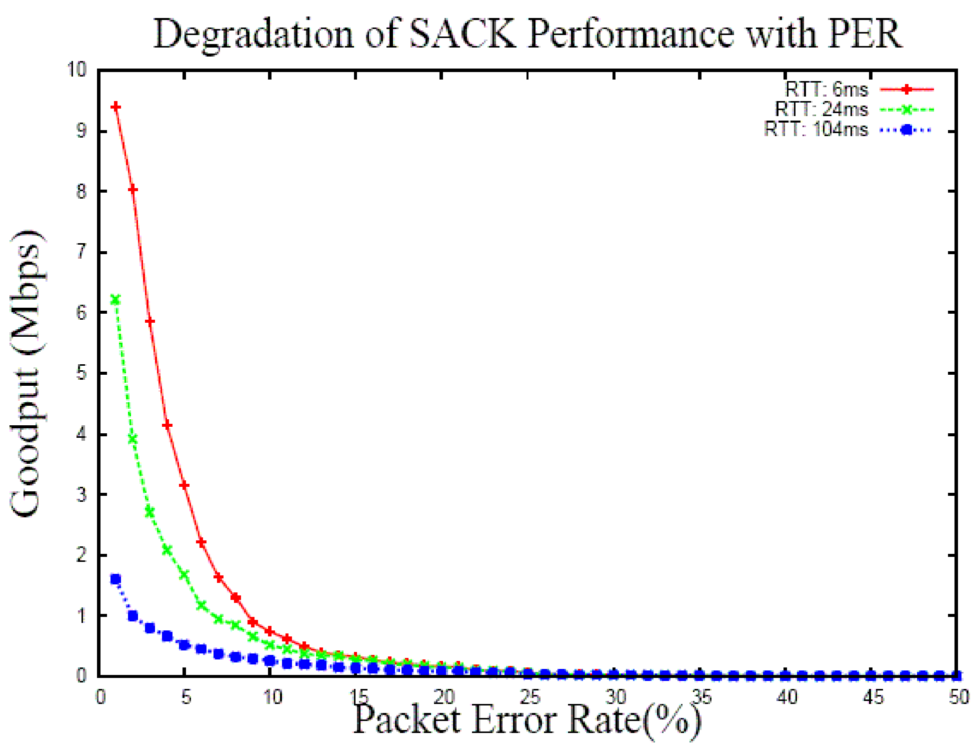
Outline

- Motivation
- LT-TCP overview
- Performance experiments and results
- Ongoing efforts and Future directions
- Short demo

LT-TCP: History & Acknowledgements

- Protocol proposed in 2007; ns-2 simulation study
- Linux kernel implementation effort since 2011
  - Joint effort between RPI and MIT Lincoln Labs
- Key collaborators:
  - Shiv Kalyanaraman (RPI; now at IBM), K.K. Ramakrishnan (AT&T)
  - Vijay Subramanian, Vicky Sharma, Brian Molnar, Buster Holzbauer, Nico Sayavedra, Jeff Wright, Jay Chamberlain, Kevin Battle (RPI students)

TCP under Lossy Conditions



TCP-SACK Degradation with increased erasure rate and RTT (i.i.d. erasure probabilities. 10 MB/s capacity, one flow)

- Observations:
  - Drastic falloff in performance with PER
  - Performance *very* bad for high loss, delay:
    - 5%+ loss rate
    - 100 ms+ RTT
- Causes:
  - TCP can not distinguish between congestion loss and link loss
    - Backs off on each loss
  - Recovers from link losses through retransmissions

How to fix TCP ?

- We have proposed Loss Tolerant TCP (LT-TCP)
- Key ideas:
  - Use Explicit Congestion Notification (ECN)
    - TCP-like congestion control algorithm, but only responsive to ECN, not arbitrary losses
  - Use Forward Error Correction (FEC) to correct for erasures
    - Proactive FEC (PFEC): sent pre-emptively to minimize recovery latency
    - Reactive FEC (RFEC): sent later as required (i.e. PFEC proves insufficient)
    - Use loss estimation for FEC provisioning
  - Separation of reliability and congestion control
    - The reliability mechanism (FEC provisioning) can be viewed as "sitting above" the window control mechanism
- We have implemented LT-TCP as a peer to TCP in the Linux kernel

Key Considerations for Robust Transport

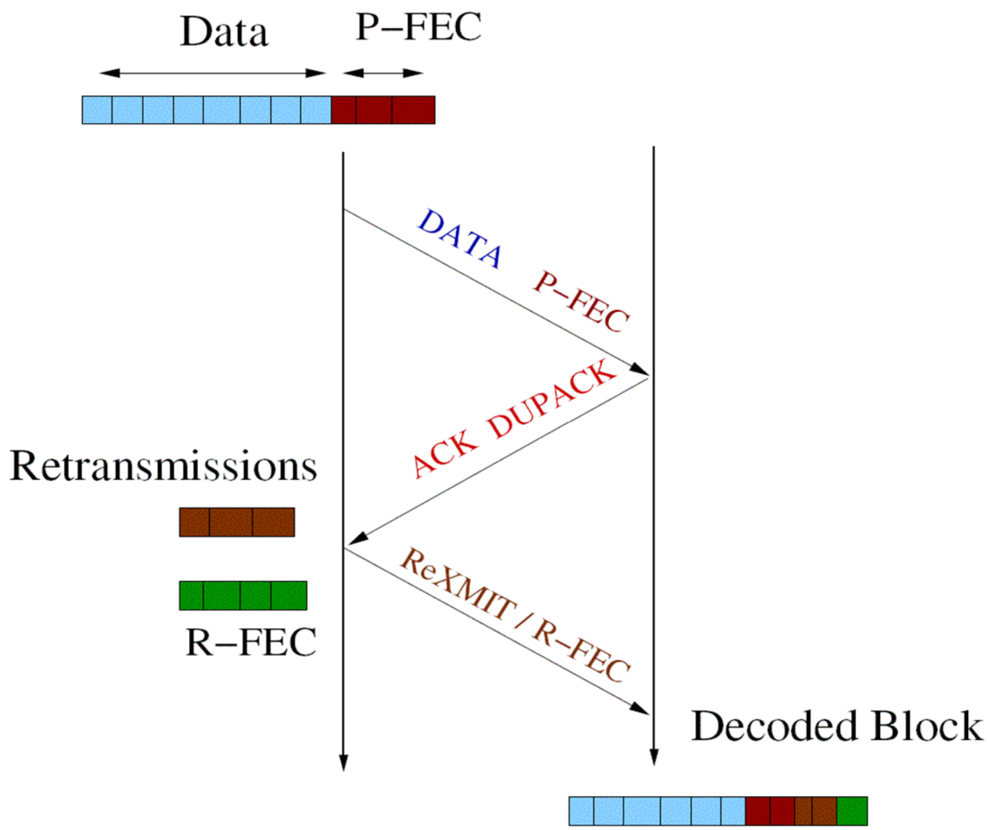
- Robust to difficult (e.g. lossy, long delay, bandwidth-limited) networks
  - MANET, Airborne, SATCOM
- Performs in stable networks
  - Internet, high-rate links
  - Match TCP performance
- Minimal reprogramming complexity for applications
  - Low effort level for reprogramming of TCP applications
  - Minimum of network knowledge required from programmer
- End-to-end
  - Minimize support from internal network components
- Implemented in the kernel

Related Work

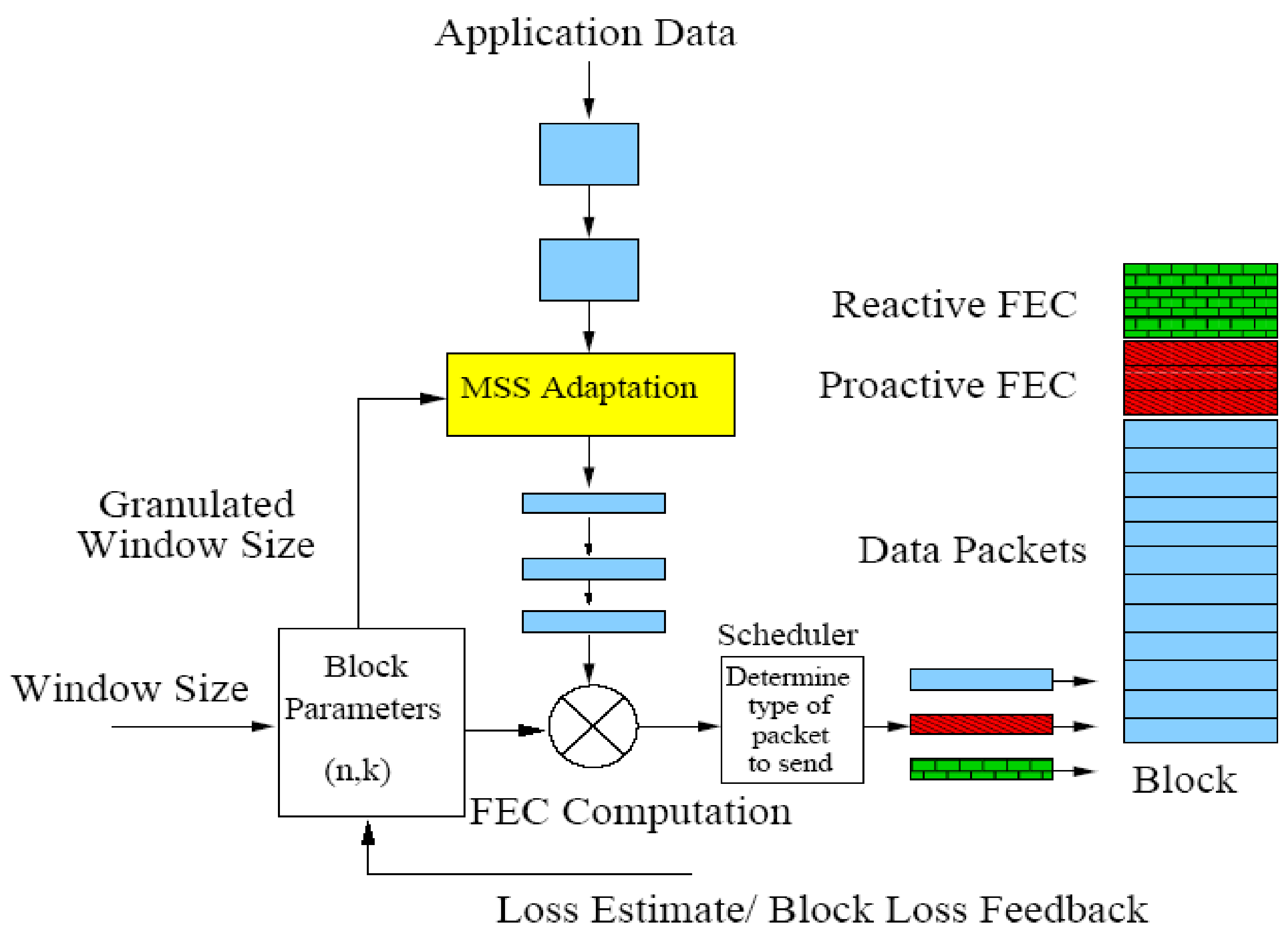
	Distinguish congestion and link losses	*Loss mitigation*	*TCP performance compatibility*	*TCP-like programming interface*	*Loss measurement based adaptation*	*Kernel Implementation*
RFC 2760 (2000)	Uses ECN		Modifications to TCP window control			
Ad hoc TCP(ATCP) (2001)	Uses ECN		Thin layer between TCP and IP			
TCP Westwood (2001)	Send-side b/w estimation from ACK return rate		Largely similar to TCP Reno		Loss rate based window adaptation	In kernel
TCP+ adaptive FEC (2004)		Proactive and reactive FEC	Adds a redundancy layer on TCP		Loss estimate based FEC provisioning	
RFC 5740 (NORM) (2009)		Mainly reactive FEC, proactive optional	Congestion control options			
Coded TCP (CTCP) (2012)	RTT Estimation	Proactive and reactive FEC	Alternative congestion control		Loss estimate based FEC provisioning	
LT-TCP (2013)	Uses ECN	Proactive and reactive FEC	Behaves as TCP-SACK at zero loss rates		Loss estimate based FEC provisioning	Research Implementation

LT-TCP: Proactive and Reactive FEC

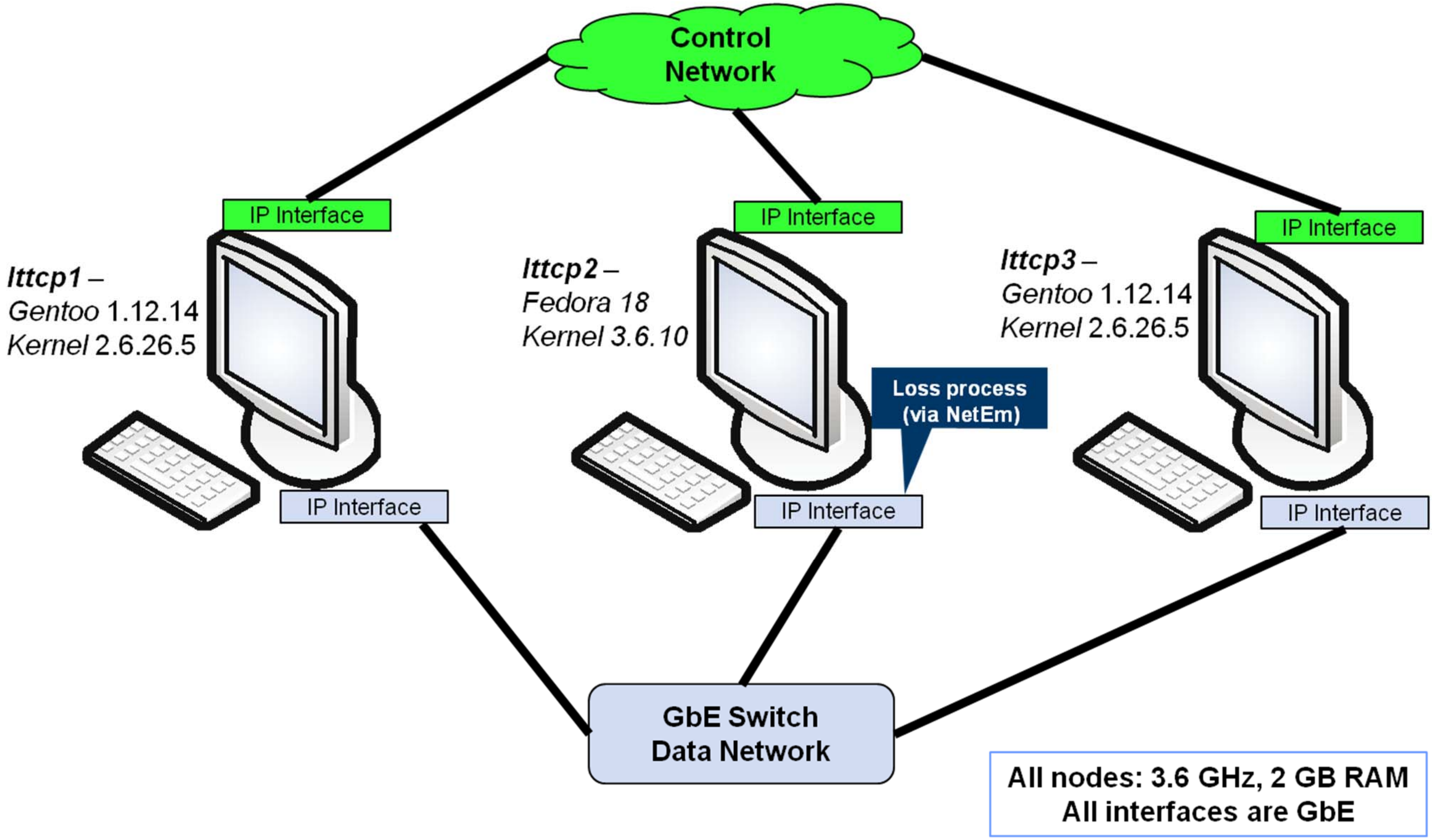
- Properties:
  - Data encoded in blocks
    - Erasure coding used
  - Data + PFEC sent in the initial transmission
  - Received data + PFEC + RFEC used to recover original data
    - Block recoverable as long as the number of packets (Data or PFEC/RFEC) received is no less than the number of data packets in block
  - Receiver feedback used to compute loss estimate
    - Used to determine how much PFEC, RFEC should be sent



LT-TCP Components



LT-TCP Testbed



Performance Comparison Description

- Overview: Set of 10MB file transfer results over the same testbed for three transport protocols
  - TCP-SACK
  - LT-TCP
  - NORM
- Parameters
  - Packet erasure rate (correlated, uncorrelated)
- Configuration
  - No congestion
  - NORM protocol was parameterized with line rate of testbed

NORM Details

- Transport protocol for both multicast and unicast proposed and implemented by Naval Research Laboratory (NRL)
  - Provides robust performance in the presence of packet losses
  - Implemented as user-space code
  - Can be called as a library or in “proxy” mode; we used library
  - Download:src-norm-1.5b1.tgz; Site:http://downloads.pf.itd.nrl.navy.mil/norm/
    - Used normFileSend.cpp, normFileRecv.cpp applications
- Summary
  - Uses FEC to repair errors, FEC also sent proactively in implementation
  - Has some form of congestion control (not used here)
  - Leverages user-supplied information for flow control

At high loss rates, TCP-SACK performance is extremely poor/crashes;
NORM is a better performance comparison candidate

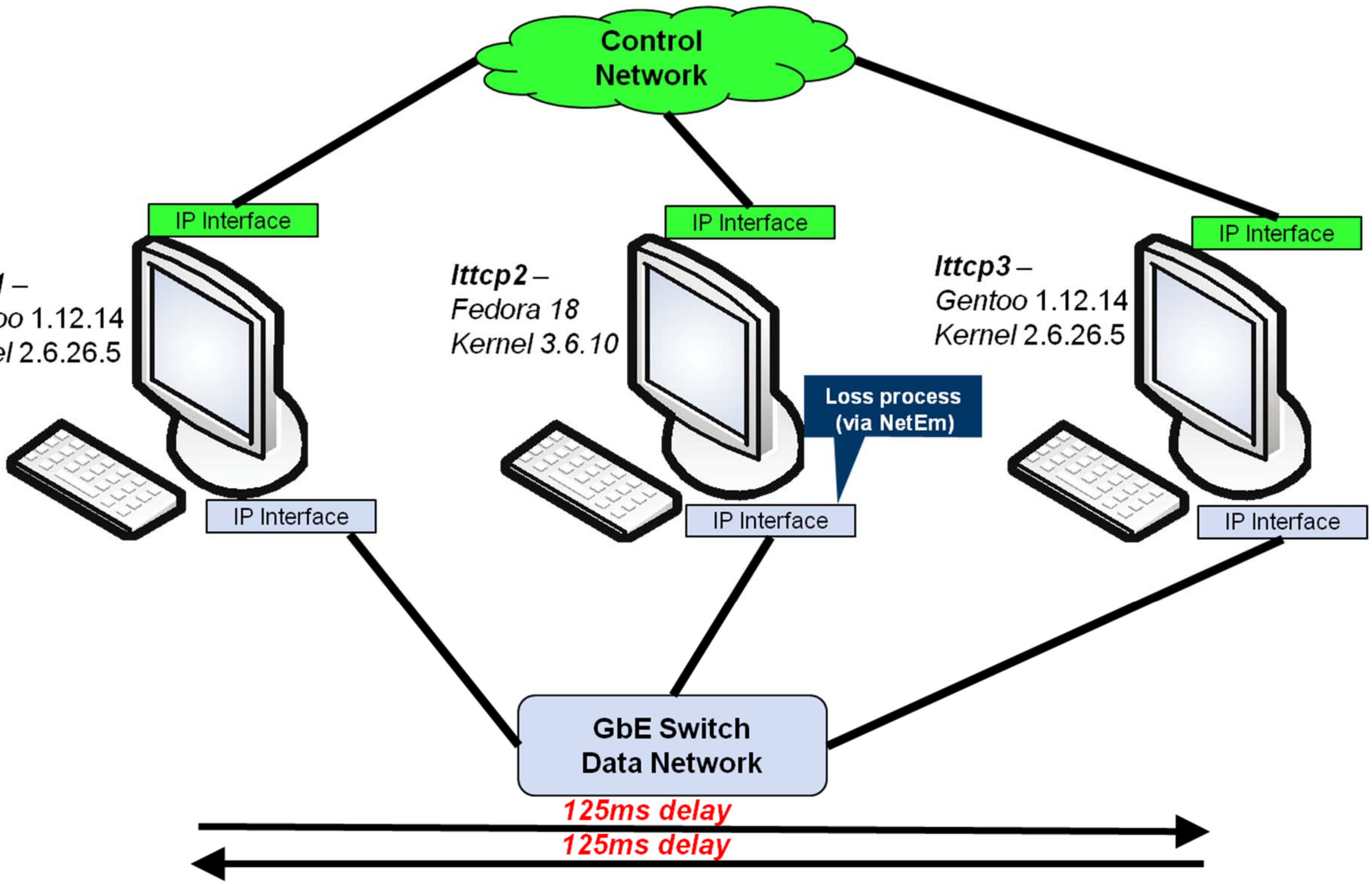
Performance under Correlated Losses

Erasure Rate [E_{Uncorr}]	LT-TCP Uncor-related	LT-TCP E=2	LT-TCP E=5	LT-TCP E=10	NORM Uncor-related	NORM E=2	NORM E=5	NORM E=10	TCP-SACK Uncor-related	TCP-SACK E=2	TCP-SACK E=5	TCP-SACK E=10
	Increasing correlation →				Increasing correlation →				Increasing correlation →			
0% [N/A]	.17	N/A	N/A	N/A	.34	N/A	N/A	N/A	.85	N/A	N/A	N/A
5% [1.05]	.46	.45	.44	.48	.78	.91	.71	.75	6.92	27.94	174.15	350.57
10% [1.18]	.62	.68	.68	.83	1.3	1.1	1.0	1.0	26.03	111.52	508.58*	>1000*
20% [1.25]	1.18	1.12	1.25	1.41	1.7	1.6	1.9	2.2	152.96	∞	∞	∞

Transfer time results for 10MB file transfer(seconds)

**Average of completed trials; some did not complete*

SATCOM Configuration Testbed



Performance under Long Delays

	LT-TCP				NORM				TCP-SACK			
Erasure Rate	Uncor-related	E=2	E=5	E=10	Uncor-related	E=2	E=5	E=10	Uncor-related	E=2	E=5	E=10
	Increasing correlation →				Increasing correlation →				Increasing correlation →			
0%	3.24	N/A	N/A	N/A	6.78	N/A	N/A	N/A	4.45	N/A	N/A	N/A
5%	3.75	3.69	3.44	3.46	11.55	10.76	9.78	11.06	254.36	171.04	150.65	200*
10%	3.74	3.74	3.75	3.88	12.52	11.78	15.03	11.63	488.97	550*	∞	∞
20%	4.36	4.30	4.17	4.04	15.08	17.57	21.42	20.75	∞	∞	∞	∞
30%	4.80	4.68	4.62	4.67	27.85	29.25	30.07	34.34	∞	∞	∞	∞
40%	4.81	4.92	5.04	5.32	45.59	46.52	49.01	47.87*	∞	∞	∞	∞
50%	5.83	5.81	5.93	6.60	78.15	70.15	79.84*	∞	∞	∞	∞	∞

Transfer time results for 10MB file transfer(seconds)

**Average of completed trials; some did not complete*

Summary and Directions

- LT-TCP implementation/evaluation summary
  - Familiar socket programming model
  - File transfer performance robust to loss rate, loss correlation
  - File transfer performance robust to long RTT
  - Comparisons to TCP-SACK, NORM (plan to do CTCP soon)
- Ongoing efforts and future directions
  - Completion of portability upgrade
  - Testing of ECN reaction code
  - Exploration of alternate congestion control techniques
  - Integration with applications and performance testing
- Demo
  - Image (file) transfer comparison between TCP and LT-TCP

Thank you!

Questions?