Enabling Supernova Computations on Dedicated Channels

Malathi Veeraraghavan
University of Virginia
mv@cs.virginia.edu

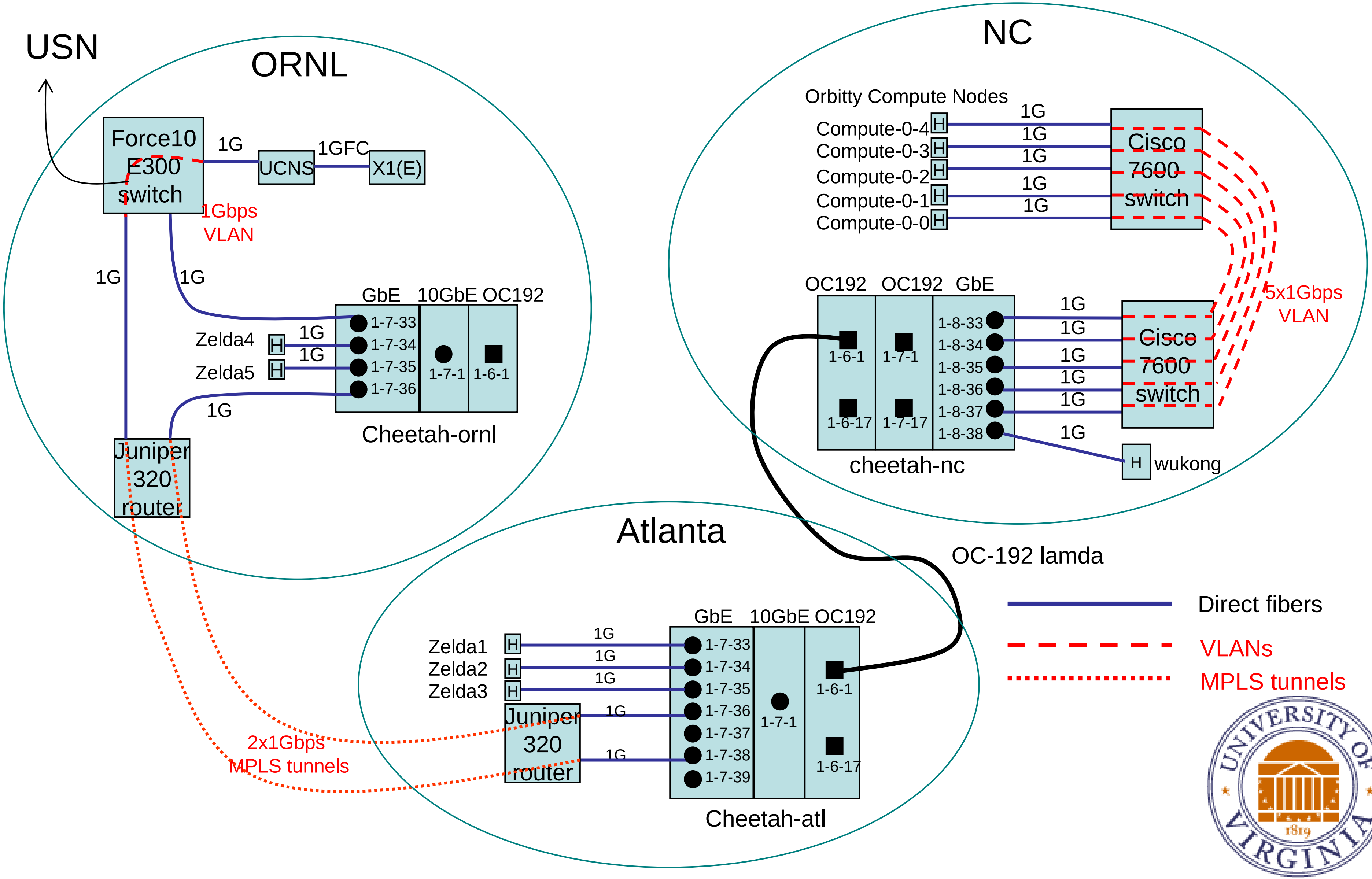
Acknowledgment

- Thanks to the DOE MICS R&D program
  - DOE DE-FG02-04ER25640
- Graduate students
  - Zhanxiang Huang
  - Anant P. Mudambi
  - Xiuduan Fang
  - Xiangfei Zhu
- Postdoc fellow
  - Xuan Zheng
- CHEETAH project co-PIs:
  - ORNL, NCSU CUNY

UVA work items - 3 tracks

- Provisioning across CHEETAH and UltraScience networks
- Transport protocol for dedicated circuits
- Extend CHEETAH concept to enable heterogeneous connections – "connection-oriented internet"

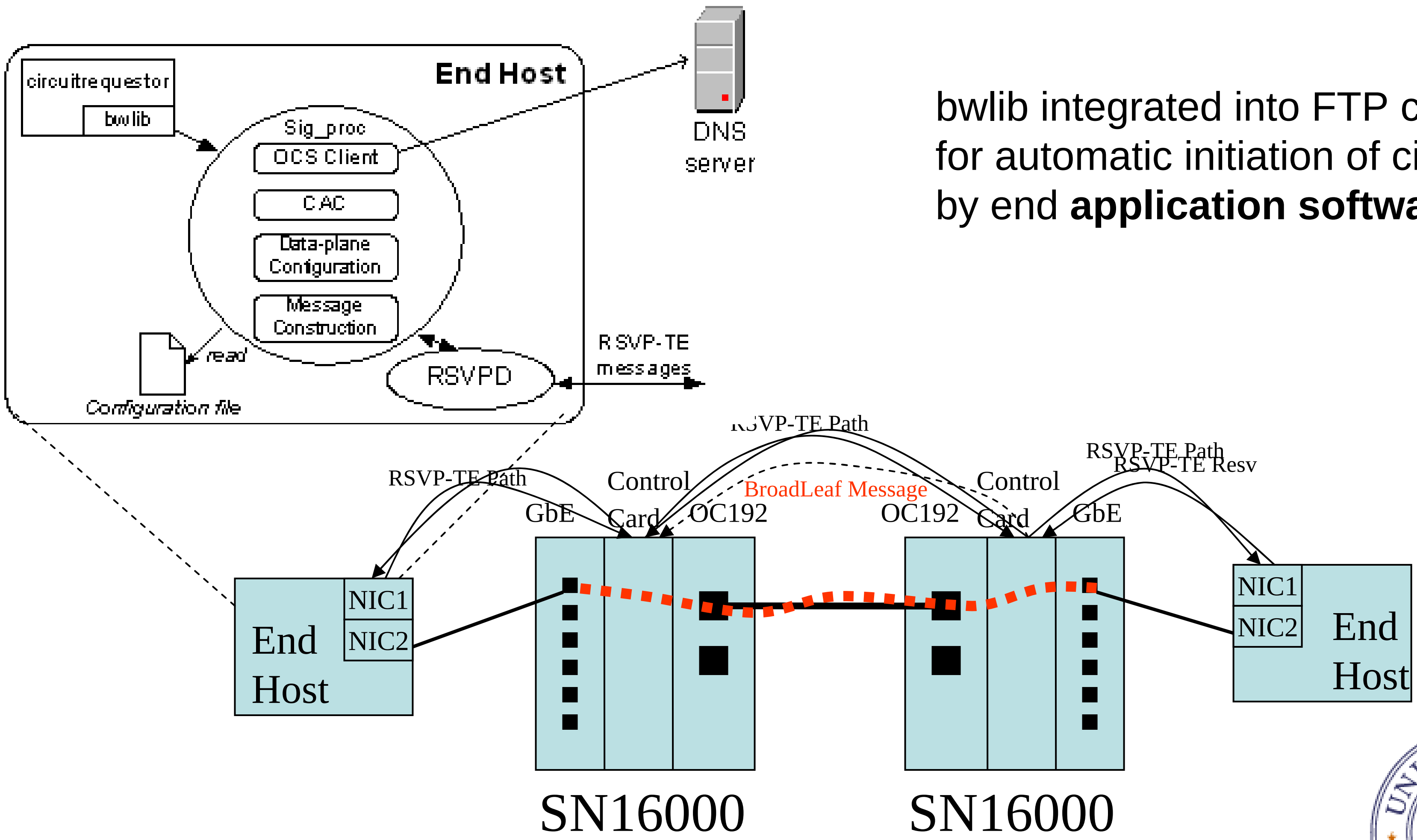
CHEETAH Network (data-plane)



CHEETAH nodes at the three PoPs

- Sycamore SN16000 intelligent optical switch
  - GbE, 10GbE, and SONET interface cards (we have OC192s)
  - Switch OS - BroadLeaf – implements GMPLS protocols since Release 7.0
    - Pure SONET circuits – excellent GMPLS signaling and routing support
    - Ethernet-to-SONET GMPLS standards not officially released yet
    - But Sycamore provided us a proprietary solution - it is quite stable

Distributed signaling implemented
(RSVP client software for end host done)



bwlib integrated into FTP code for automatic initiation of circuit by end **application software**

Call setup delay measurements

Circuit type	End-to-end circuit setup delay (s)	PATH processing delay at the *SN16k-NC*	RESV processing delay at the *SN16k-NC*
OC1	0.166103	0.091119	0.008689
OC3	0.165450	0.090852	0.008650
1Gbps Ethernet/EoS	4.982071	4.907113	0.008697

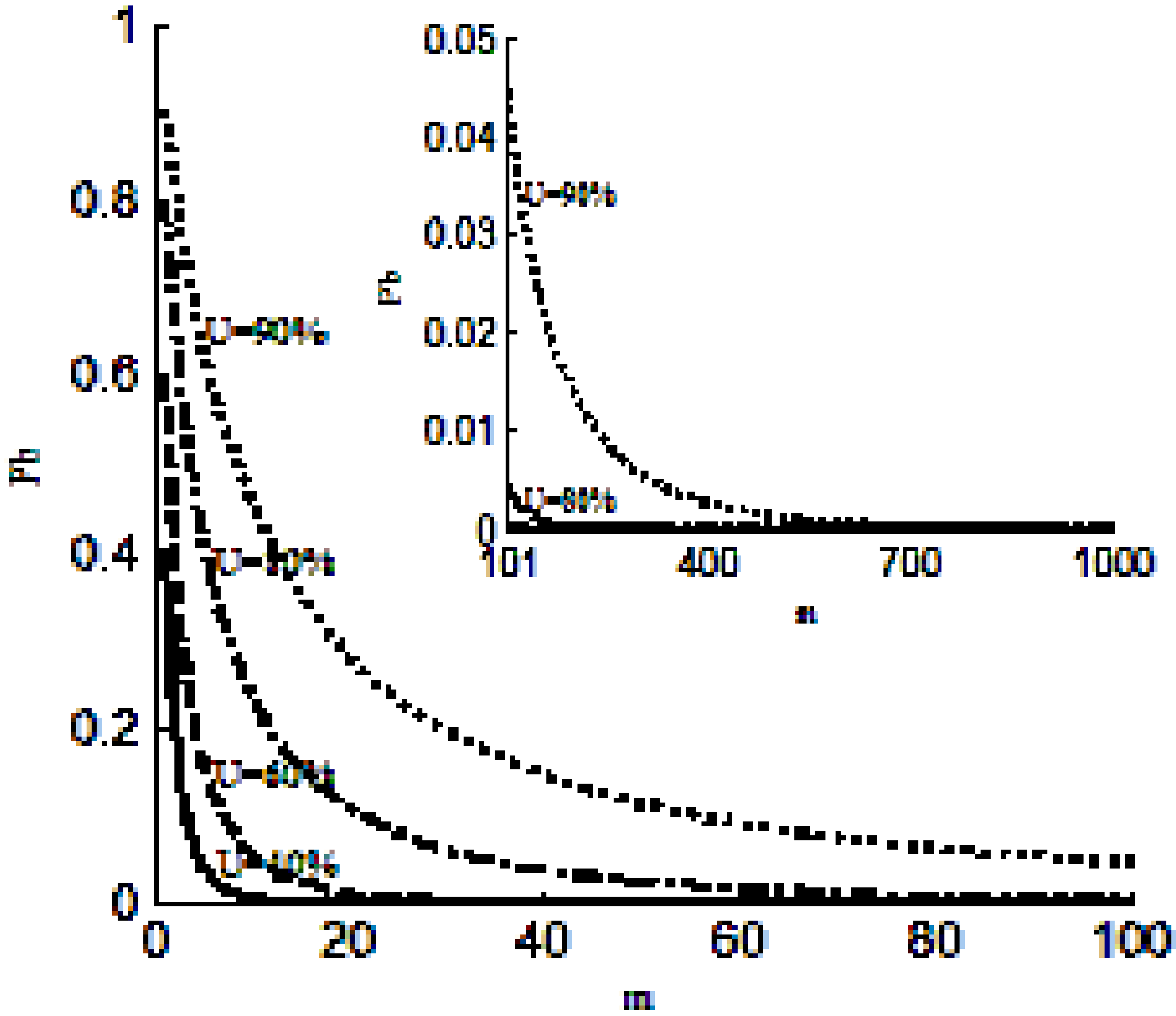
21 OC1 on SONET side are set up one at a time
21 x 0.166 + 21 x 0.025 = ~4sec

propagation + emission delays

Key results

- Distributed GMPLS signaling + routing using off-the-shelf switches works well!
- GMPLS control-plane protocols suitable for
  - Only call blocking mode of bandwidth sharing
  - Immediate-request calls, not book-ahead
- If the number of circuits (m) sharing the link is small (order of 10), e.g. 1Gbps on 10Gbps link
  - Either call blocking probability will be high
  - Or link utilization will be low

Call blocking probability as a function of utilization, U, and number of circuits, m



UVA work items - 3 tracks

- Provisioning across CHEETAH and UltraScience networks
- ➢ Transport protocol for dedicated circuits
- Extend CHEETAH concept to enable heterogeneous connections – "connection-oriented internet"

FTP over TCP across circuit seems to do best!

	zelda4 - compute-0-0		zelda4 - zelda3		zelda4 - wukong		zelda3 - compute-0-0		zelda3 - wukong		wukong - compute-0-0	
RTT (ms)	32				13.7				8.75		1	
Memory-to-memory transfers												
Iperf TCP	938	924	938	938	931	900	933	934	934	N/A	N/A	933
Iperf UDP	888	913	957	957	646	830	800	913	653	727	750	645
Disk-to-disk transfers (1.3GB file)												
FTP	752	552	585	585	702	458	878	479	702	722*	N/A	620
SFTP	25	18.8	34.9	35.1	17.3	17.9	41	18.7	26.3	24.3	26.4	130
SABUL	640	770	488	624	470	404	638	848	463	610	520	479
Hurricane	524	545	537	422	456	368	530	542	264	282	N/A	N/A
BBCP	N/A	500	607	657	N/A	N/A	N/A	611	425	379	87	513
FRTPv1	629	600	853	610	510	664	644	787	515	620	368	388

First attempt

- Fixed-Rate Transport Protocol (FRTP)
  - User-space implementation
  - UDP-based solution
  - Null flow control
    - Thought we could estimate receive rate (disk bottleneck)
    - Use this as circuit rate
    - Stream data from sender at this (fixed) circuit rate
  - Hard to do the above because of
    - Variability in disk receive rate
    - Multitasking hosts (general-purpose)
  - CPU-intensive solution for maintaining constant sending rate

Decided to modify TCP

- Reasons
  - Due to failure of null flow control solution, we decided on window based flow control
    - best out of three: ON/OFF and rate based
    - need kernel-level implementation for window control
  - Self-clocking in TCP works well to maintain fixed sending rate
    - Busy wait discarded due to high CPU utilization
    - Signals and timers unreliable
  - TCP implementation + Web100 stable and fast

Protocol Design

- Congestion control: Control plane function
  - TCP's data-plane congestion control redundant
- Flow control: Window based
- Error control: Required for reliable transfer
  - Thought we could use just negative ACKs because of in-sequence delivery characteristic of circuits
  - But cost of retrieving errored frames from disk high
  - Need positive ACKs to remove packets held in retransmission buffers
- Multiplexing: TCP's port solution usable with circuits
- Conclusion: TCP's mechanisms suitable for all but congestion control

Hence Circuit-TCP
How does it differ from TCP?

- Data plane
  - Slow Start/Congestion Avoidance removed
  - Sender sends at the fixed rate of circuit
- Control plane
  - TCP has three-way handshake (host-to-host) to synchronize initial sequence numbers
  - Circuit-TCP determines rate to use for the transfer and initiates request for circuit

Implementation Issues

- Selecting the rate of the circuit
  - End-hosts usually a bottleneck, esp., for disk-to-disk transfers. Difficult to pin down a constant bottleneck rate for the whole transfer
  - Pragmatic approach:
    - Minimize sources of variability (e.g., avoid multitasking)
    - Use an empirical estimate of the receiver's disk write rate
- Maintaining the sending rate at the circuit rate

Linux implementation

Sender:

- Try to maintain fixed amount of outstanding data in network

ncap (>= BDP = circuit rate * RTT)

- In TCP implementation replace min(cwnd, rwnd) by min(ncap, rwnd)
- 2 requirements
  - C-TCP and TCP co-exist
  - If socket is C-TCP then the kernel needs to know ncap for the socket
- Web100 provides API that was extended to meet these 2 requirements

Linux implementation

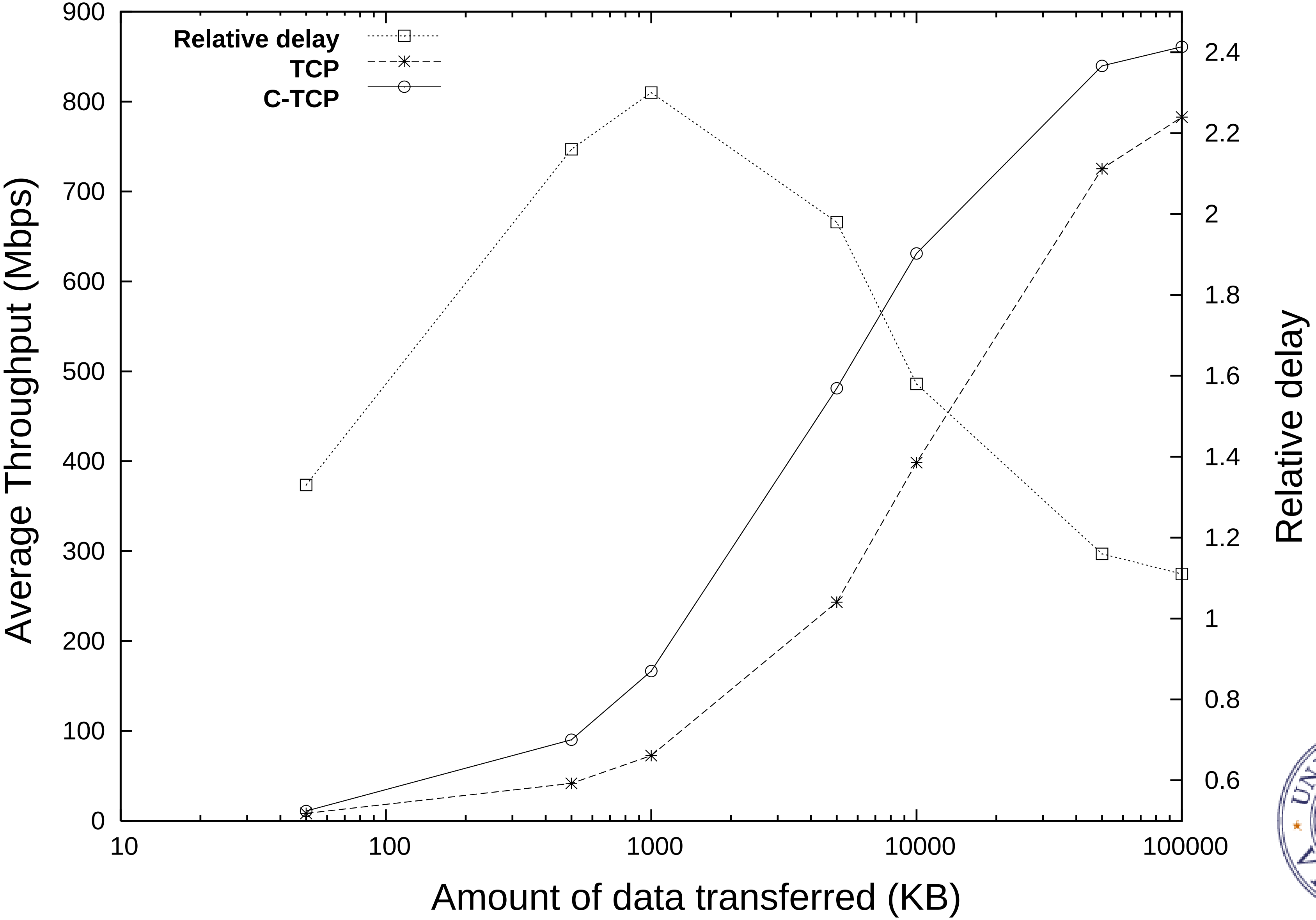
Receiver:

- Linux increments advertised window in a slow start-like fashion
- So for the initial few RTTs, at the sender min(ncap, rwnd) = rwnd, leading to low circuit utilization
- To avoid this, for C-TCP, we modified this code

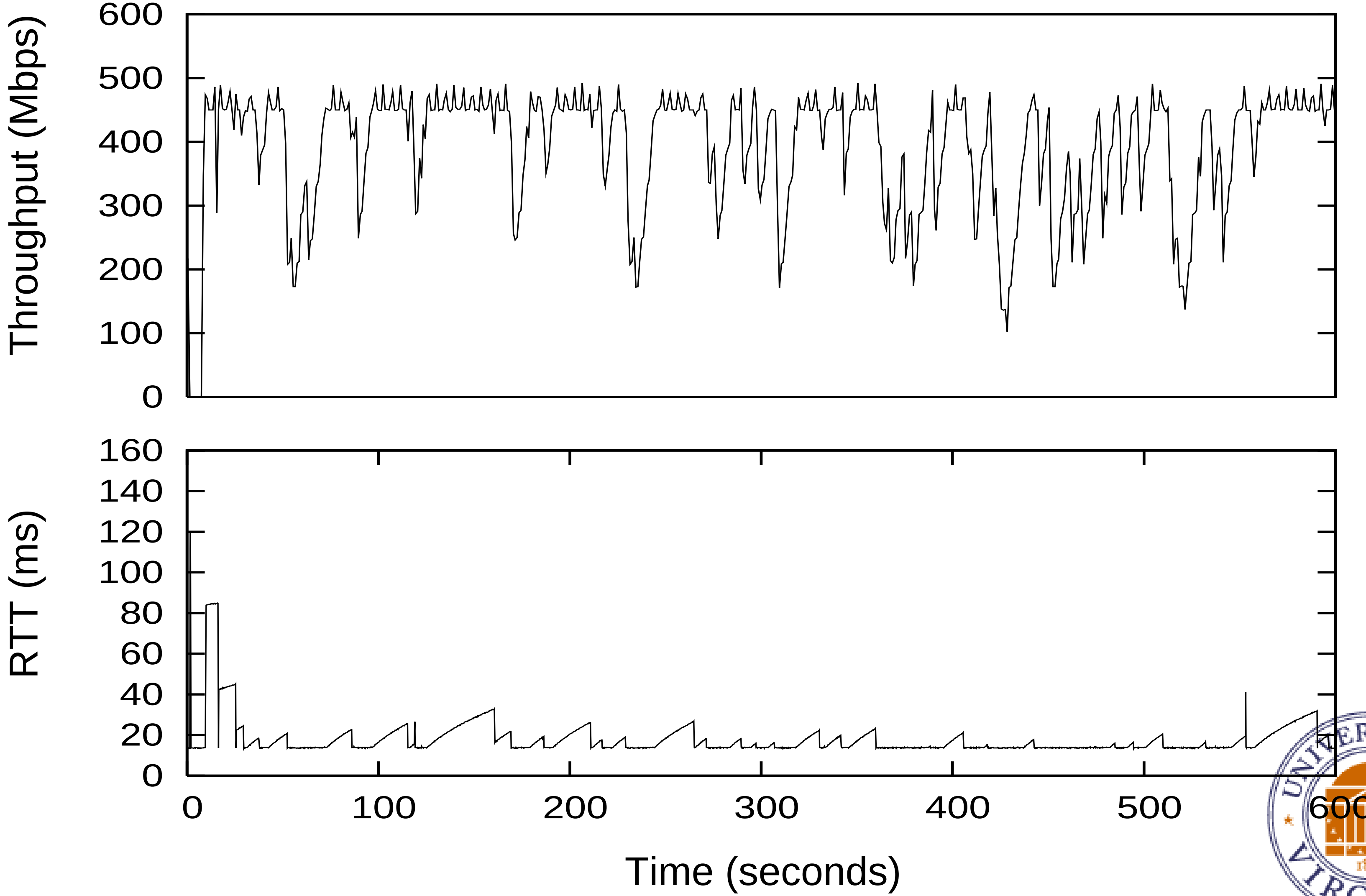
Three sets of results

- "Small" memory-to-memory transfers
  - up to 100MB
  - 1Gbps circuit, RTT: 13.6ms
- How RTT varies - sustained sending rate
  - 500Mbps SONET circuit; RTT: 13.6ms
  - Sycamore gateway will buffer Ethernet frames since sending NIC will send at 1Gbps
- Disk-to-disk: 1.6GB over a 1Gbps circuit

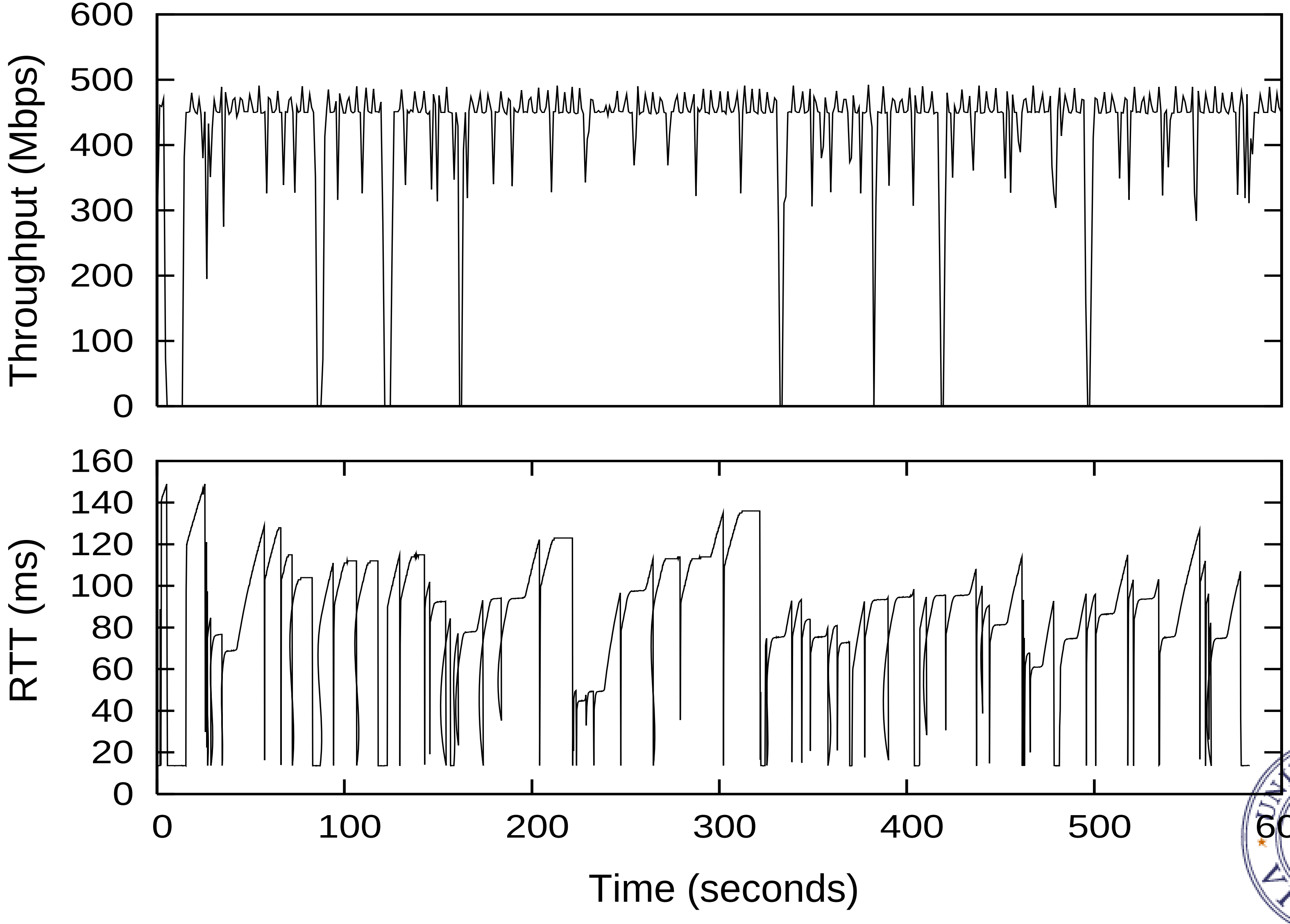
"Small" memory-to-memory transfers



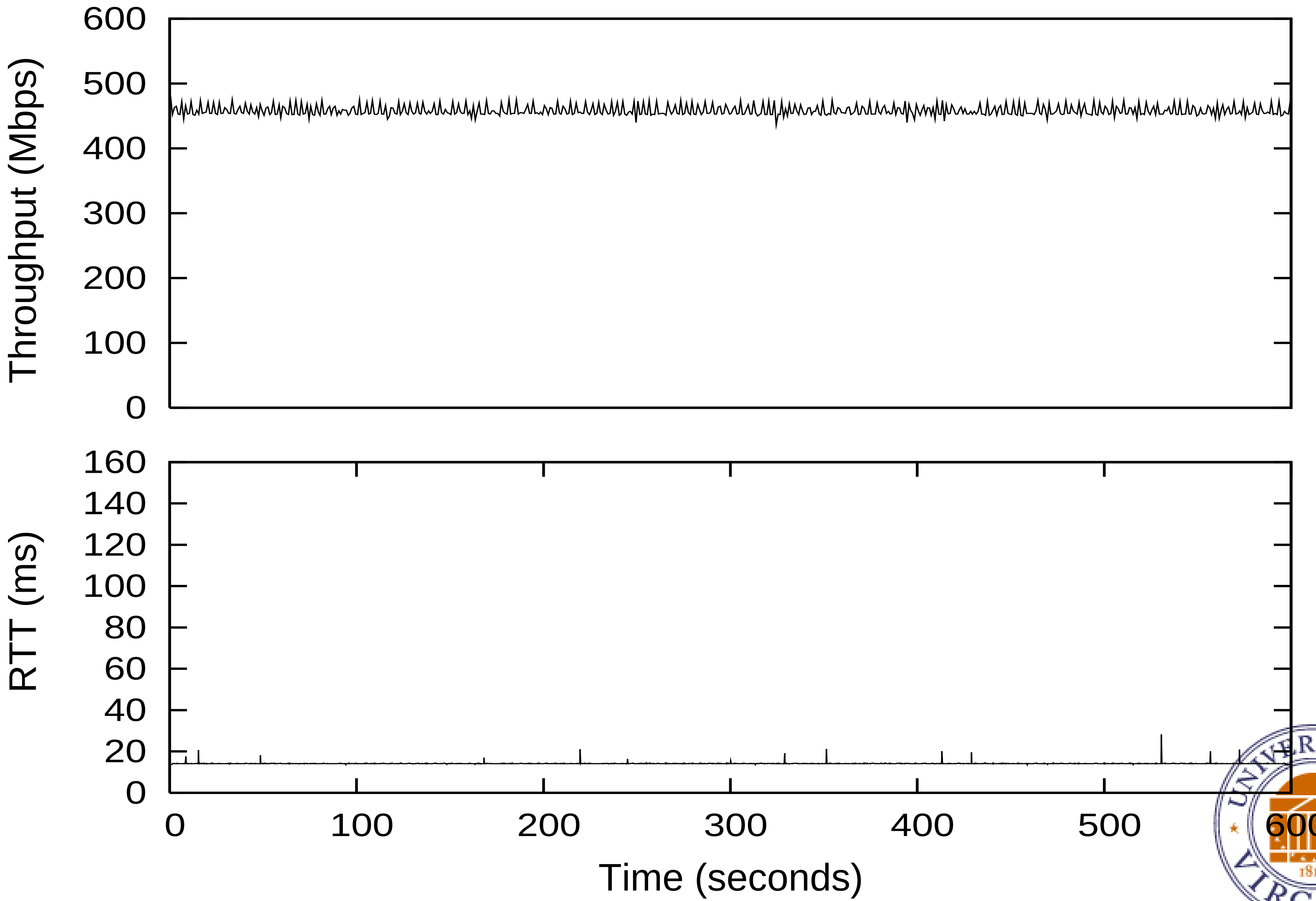
RTT variation: Reno TCP



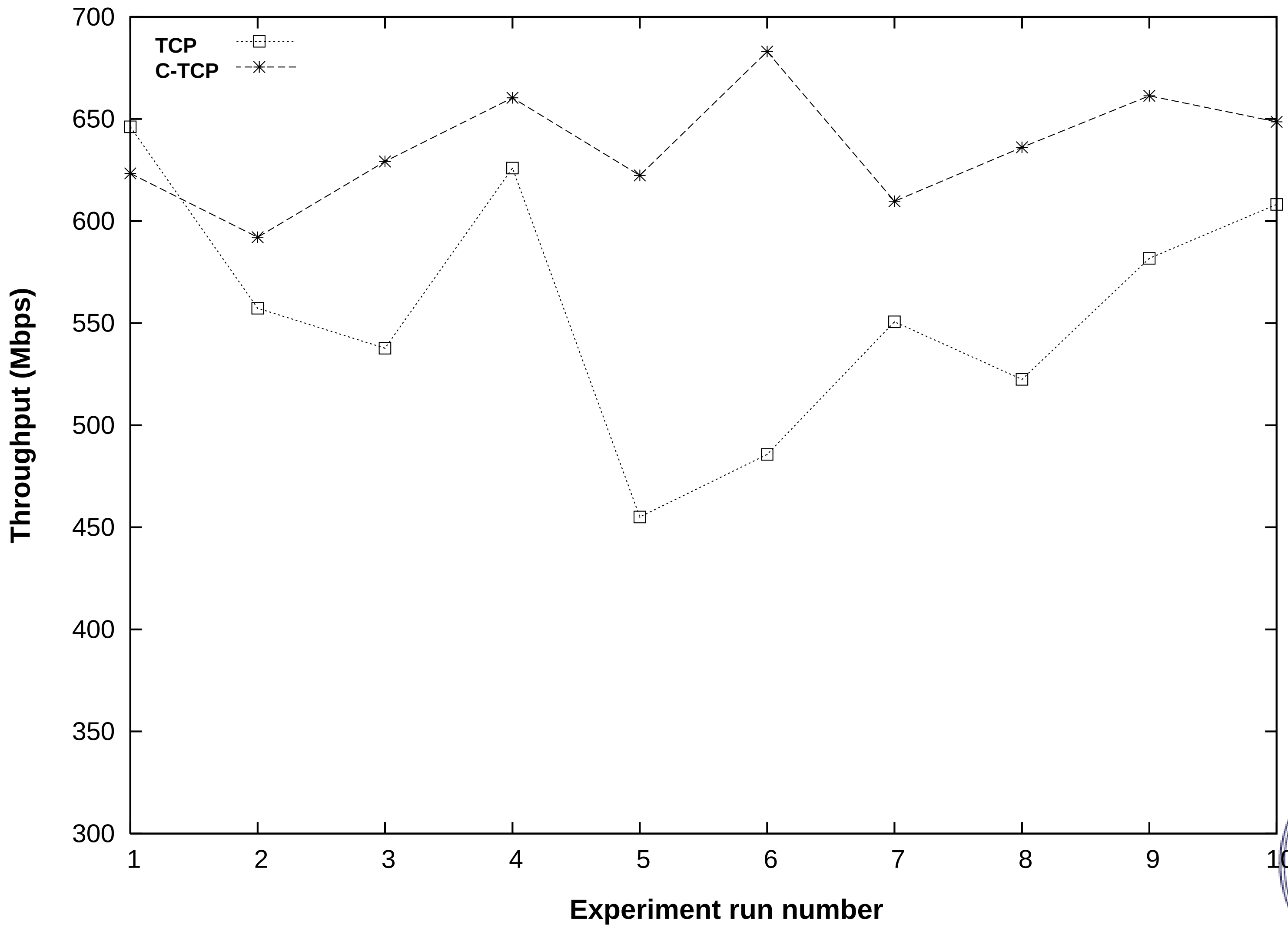
RTT variation: BIC-TCP



RTT variation: C-TCP



Disk-to-disk: 1.6GB; 1Gbps circuit



UVA work items

- Provisioning across CHEETAH and UltraScience networks
- Transport protocol for dedicated circuits: Fixed-Rate Transport Protocol (FRTP)
- Extend CHEETAH concept to enable heterogeneous connections – "connection-oriented internet"

Connection-oriented internet

- Cisco and Juniper routers implement
  - MPLS (user-plane)
  - RSVP-TE (control-plane)
- Many vendors' Ethernet switches
  - Untagged (port-mapped) VLANs
  - Tagged VLANs with priority queueing
  - Add external GMPLS controller

Design of signaling procedures

- Key idea
  - Signaling message encapsulation
    - Decreases delay - multiple RTTs avoided
  - Combine with Interface Adaptation Capability Descriptor (IACD)
- CSPF not very useful
  - TE-LSAs only intra-area within OSPF
  - Inter-domain - topology hiding

Recently completed papers

- A. P. Mudambi, X. Zheng, and M. Veeraraghavan, “A transport protocol for dedicated circuits, submitted to IEEE ICC 2006.
- X. Zhu, X. Zheng, M. Veeraraghavan, Z. Li, Q. Song, I. Habib N. S. V. Rao, “Implementation of a GMPLS-based Network with End Host Initiated Signaling,” submitted to IEEE ICC 2006.
- M. Veeraraghavan, X. Fang, X. Zheng, “On the suitability of applications for GMPLS networks,” submitted to IEEE ICC 2006.
- M. Veeraraghavan, X. Zheng, Z. Huang, “On the use of GMPLS networks to support Grid Computing,” submitted to IEEE Communications Magazine.