



High bandwidth, Long distance....

Where is my throughput?

ARGONNE



STARLIGHTSM

UIC

Robin Tasker

CCLRC, Daresbury Laboratory, UK

[r.tasker@dl.ac.uk]



DataTAG is a project sponsored by the European Commission - EU Grant IST-2001-32459

RIPE-47, Amsterdam, 29 January 2004



Technical Directions

Objective

“to create a large scale intercontinental Grid testbed involving the EDG project, several national projects in Europe, and related Grid projects in the USA.”

To investigate

- *Advanced networking and*
- *Grid interoperability issues*

between these different Grid domains.

Achieved via Work Packages, including,

WP1: Establishment of a high performance intercontinental Grid testbed

WP2: High performance networking

WP3: Bulk data transfer validations and application performance monitoring

WP4: Interoperability between Grid domains



WP2: High Performance Networking

Task 2.1: Transport applications for high bandwidth-delay connections

Final report and demonstration of high performance data transport protocols suitable for long distance Grid data replication requirements

Task 2.2: End-to-end inter-domain QoS

Final report and demonstration of intercontinental QoS services in Grid environment

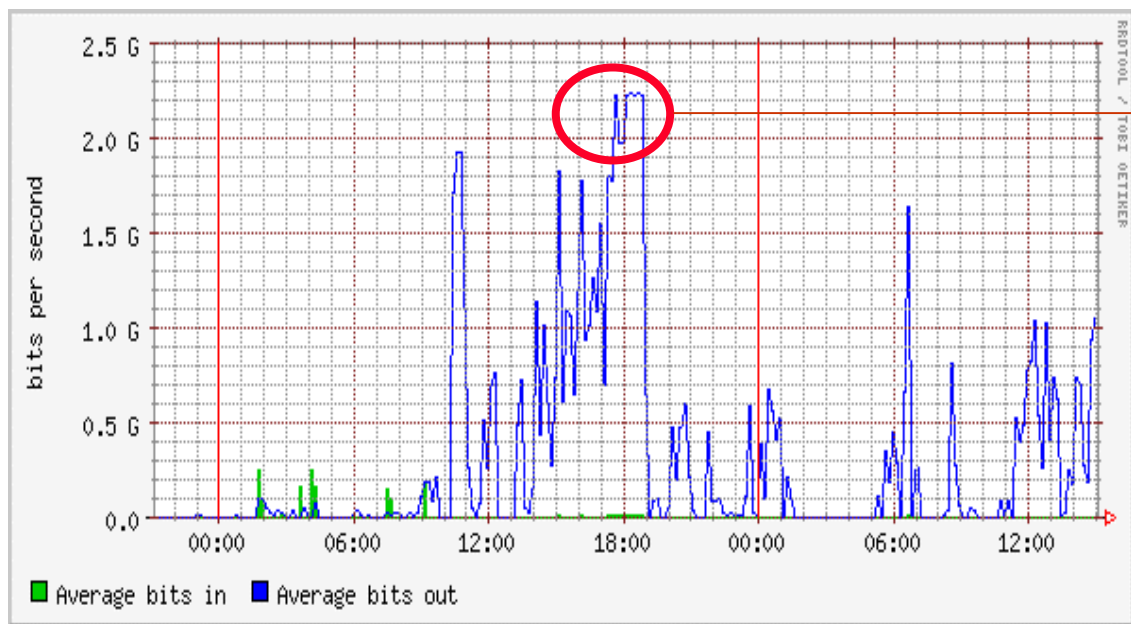
Task 2.3: Advanced reservation

Deployment of network services and advance reservation in the intercontinental testbed





Throughput... What's the problem?



One Terabyte of data transferred in less than an hour

On February 27-28 2003, the transatlantic DataTAG network was extended, i.e. **CERN - Chicago - Sunnyvale** (>10000 km).

For the first time, a terabyte of data was transferred across the Atlantic in less than one hour using **a single TCP (Reno) stream**.

The transfer was accomplished from Sunnyvale to Geneva at **a rate of 2.38 Gbits/s**

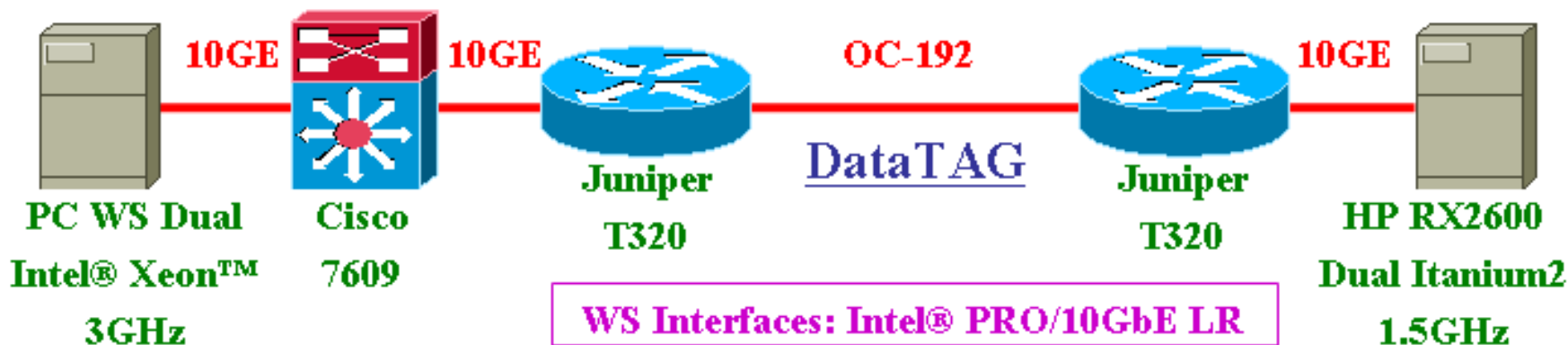


Internet2 Land Speed Record

On October 1 2003, DataTAG set a new **Internet2 Land Speed Record** by transferring **1.1 Terabytes of data in less than 30 minutes** from Geneva to Chicago across the DataTAG provision, corresponding to an average rate of **5.44 Gbits/s** using a **single TCP (Reno) stream**

Caltech (DoE) PoP - Starlight - Chicago

CERN - DataTAG - Geneva





So how did we do that?

Management of the End-to-End Connection

Memory-to-Memory transfer; no disk system involved

Processor speed and system bus characteristics

TCP Configuration – window size and frame size (MTU)

Network Interface Card and associated driver and their configuration

End-to-End “no loss” environment from CERN to Sunnyvale!

At least a 2.5 Gbits/s capacity pipe on the end-to-end path

A single TCP connection on the end-to-end path

No real user application

That’s to say - not the usual User experience!



Realistically – what's the problem?

End System Issues

Network Interface Card and Driver and their configuration

TCP and its configuration

Operating System and its configuration

Disk System

Processor speed

Bus speed and capability

Network Infrastructure Issues

Obsolete network equipment;

Configured bandwidth restrictions;

Security restrictions (e.g., firewalls)

Sub-optimal routing

Transport Protocols

Network Capacity and the influence of Others!

Many, many TCP connections

Mice and Elephants on the path

Congestion



Buses, NICs and Drivers [1]

A Methodology to Characterise End-System Capability

For each combination of **motherboard, NIC and Linux Kernel**, three sets of measurements were made using two PCs with the NICs directly connected, namely,

Latency *Round trip times were measured using Request-Response UDP frames*

UDP Throughput *transmission of streams of UDP packets at regular, carefully controlled intervals; throughput was measured at receiver.*

Bus Activity *measured on the PCI Bus*

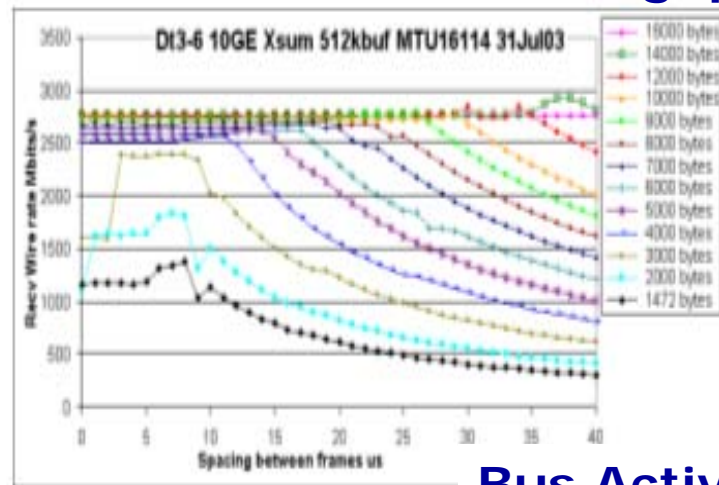
UDP/IP frames were chosen for the tests as they are processed in a **similar manner to TCP/IP frames**, but are **not subject to the flow control and congestion avoidance algorithms** defined in the TCP protocol, i.e. they do not distort the base-level performance. Methodology uses a train of UDP frames of differing MTU and differing frame spacing sent between end-systems.



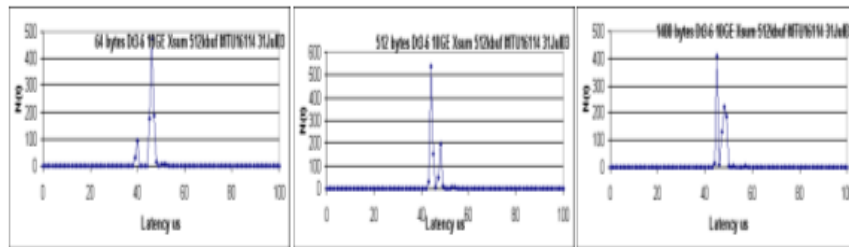
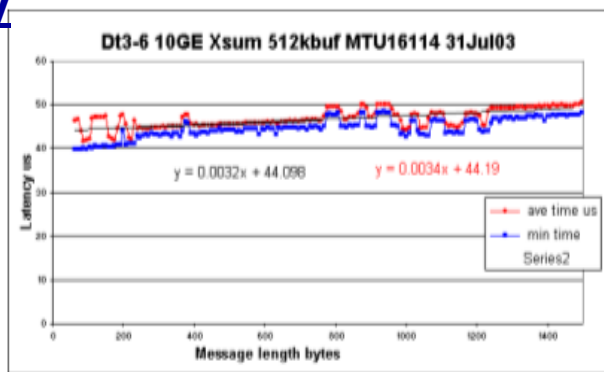
Buses, NICs and Drivers [2]

This **example** shows the characterisation of the **Intel PRO/10GbE LR 10 Gigabit Ethernet Server adaptor** with the **Supermicro P4DP8-G2 motherboard** in a system with a **Dual Xenon 2.2GHz 32 bit CPU** and **400Mhz System PCI-X bus**

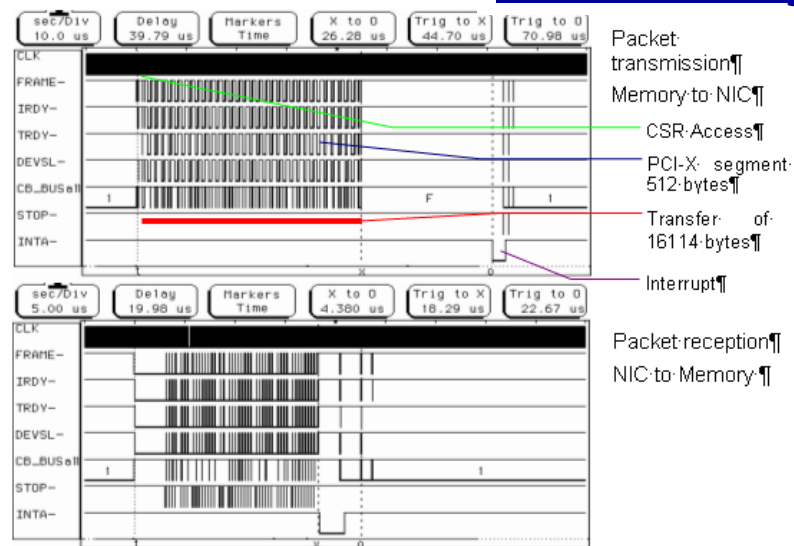
Throughput



Latency



Bus Activity





Understanding NIC Drivers [1]

Linux driver basics – TX

- Application system call
- Encapsulation in UDP/TCP and IP headers
- Enqueue on device send queue
- Driver places information in DMA descriptor ring
- NIC reads data from main memory via DMA and sends on wire
- NIC signals to processor that TX descriptor sent

Linux driver basics – RX

- NIC receives packet onto card
- NIC places data in main memory via DMA to a free RX descriptor
- NIC signals RX descriptor has data
- Driver passes frame to IP layer and cleans RX descriptor
- IP layer passes data to application

Linux NAPI driver model

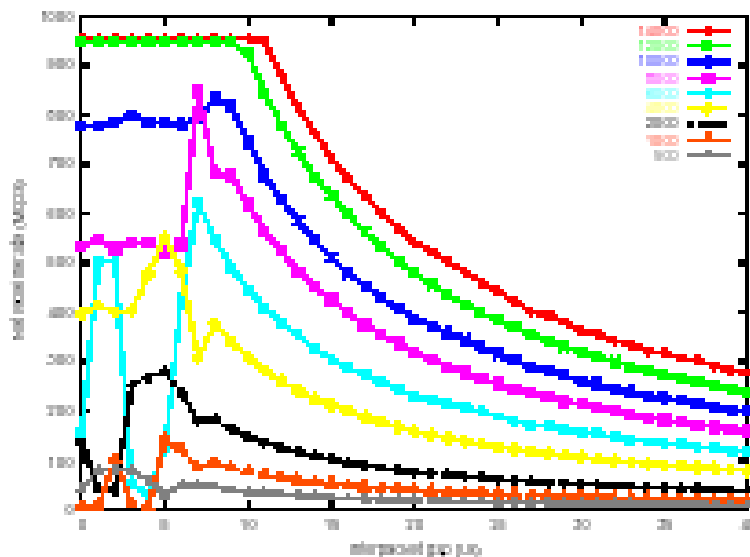
- On receiving a packet, NIC raises interrupt
- Driver switches off RX interrupts and schedules RX DMA ring poll
- Frames are pulled off DMA ring and is processed up to application
- When all frames are processed RX interrupts are re-enabled
- Dramatic reduction in RX interrupts under load
- Improving the performance of a Gigabit Ethernet driver under Linux



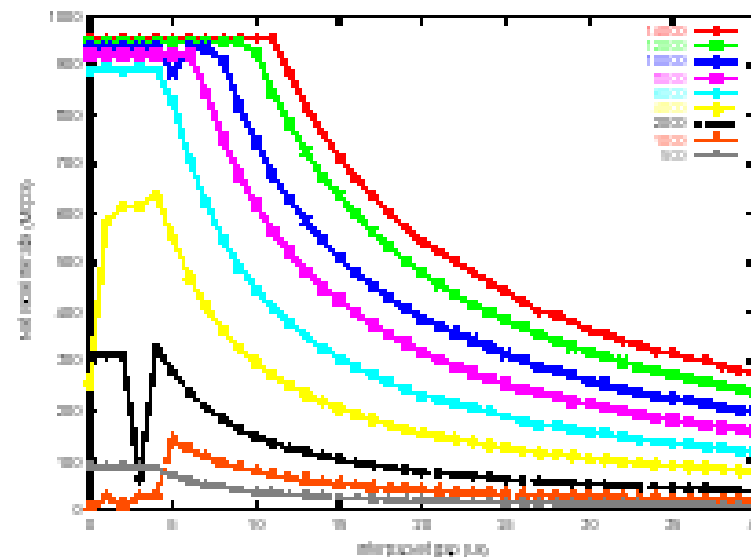
Understanding NIC Drivers [2]

NAPI receiver results

2.4Ghz machines connected through router with Linux 2.4.20 sender using TX interrupt moderation



2.4.19 non-NAPI receiver



2.4.20 NAPI receiver

Better throughput for NAPI receiver under load
Some strange behaviour with 100byte and 50byte packets



TCP, Linux and their configuration

DataTAG has produced a **Technical Report** on the *Linux Kernel 2.4.20* which describes the **structure** and **organization** of the **networking code of the Linux kernel** including the **main data structures**, the **sub-IP layer**, the **IP layer**, and two **transport layers: TCP and UDP**. See

http://datatag.web.cern.ch/datatag/papers/drafts/linux_kernel_map/

“In this technical report, we describe the structure and organization of the networking code of Linux kernel 2.4.20. This release is the first of the 2.4 branch to support network interrupt mitigation via a mechanism known as NAPI. We describe the main data structures, the sub-IP layer, the IP layer, and two transport layers: TCP and UDP. This material is meant for people who are familiar with operating systems but are not Linux kernel experts.”



TCP (Reno) – Performance

AIMD and High Bandwidth – Long Distance networks

Poor performance of TCP in high bandwidth wide area networks is due in part to the TCP congestion control algorithm

For each ack in a RTT **without** loss:

$wnd \rightarrow wnd + a / wnd$

- **Additive Increase, $a=1$**

For each window **experiencing loss**:

$wnd \rightarrow wnd - b (wnd)$

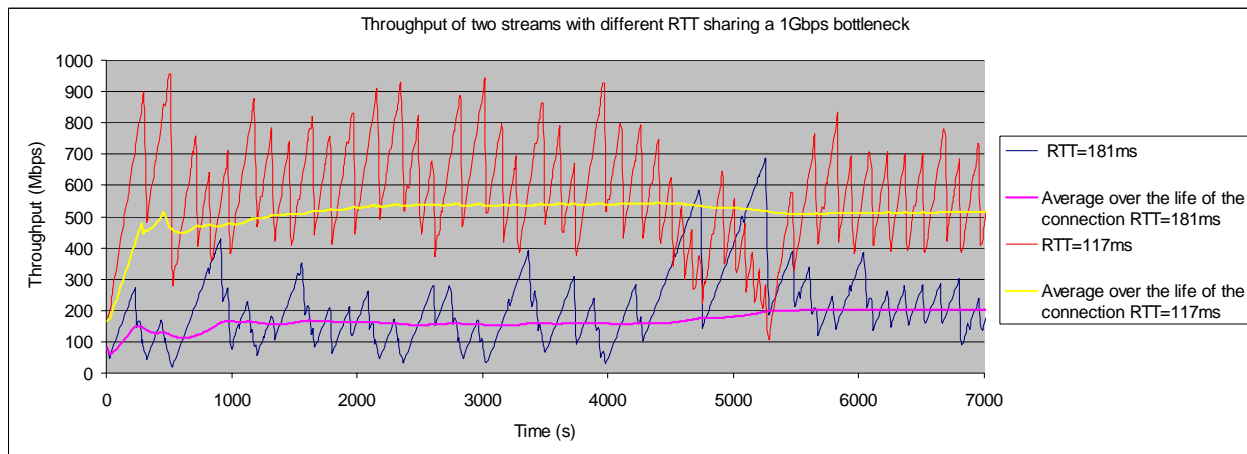
- **Multiplicative Decrease, $b=1/2$**

Throughput	Window	Loss recovery time	Supporting loss rate
10Mbps	170pkts	17s	5.4×10^{-5}
100Mbps	1700pkts	2mins 50s	5.4×10^{-7}
1Gbps	17000pkts	28mins	5.4×10^{-9}
10Gbps	170000pkts	4hrs 43mins	5.4×10^{-11}

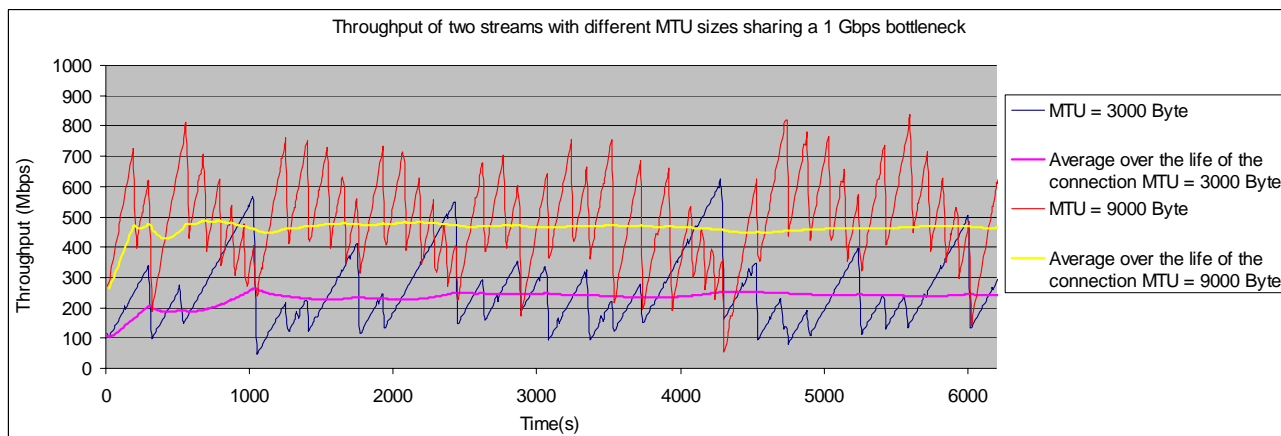


TCP (Reno) - Fairness

TCP throughput for two connections with **different RTT** sharing a 1 Gbits/s bottleneck



TCP throughput for two connections with **different MTU** sharing a 1 Gbits/s bottleneck





Solution #1 High Speed TCP

Adjusting the AIMD Algorithm – TCP Reno

For each ack in a RTT **without** loss:

$wnd \rightarrow wnd + a / wnd$ - **Additive Increase, where $a = 1$**

For each window **experiencing** loss:

$wnd \rightarrow wnd - b * (wnd)$ - **Multiplicative Decrease, where $b = 1/2$**

High Speed TCP modifies **AIMD** such that

a and **b** vary depending on **current wnd** where

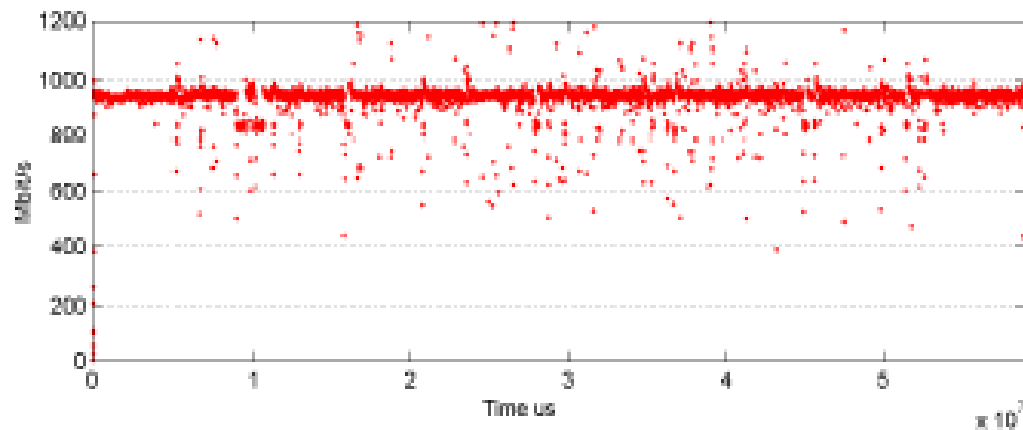
a increases more rapidly with larger wnd and as a consequence returns to the 'optimal' wnd size sooner for the network path; and

b decreases less aggressively and, as a consequence, so does the wnd .
The effect is that there is not such a decrease in throughput.

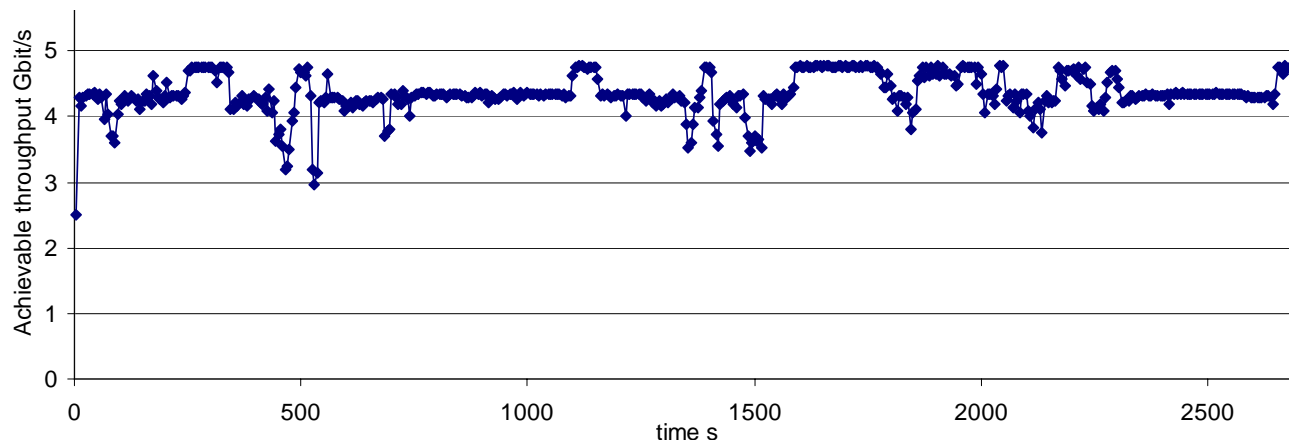


Solution #1 High Speed TCP

Single **High Speed TCP** connection over a **2.5 Gbit/s** link



Single High Speed TCP connection over a **10 Gbits/s** link





Solution #2 Scalable TCP

Adjusting the AIMD Algorithm – TCP Reno

For each ack in a RTT **without** loss:

$cwnd \rightarrow cwnd + a / cwnd$

- **Additive Increase, where $a = 1$**

For each window **experiencing loss**:

$cwnd \rightarrow cwnd - b * (cwnd)$

- **Multiplicative Decrease, where $b = 1/2$**

Scalable TCP modifies **AIMD** such that

a and **b** are **fixed** adjustments for the increase and decrease of $cwnd$ such that the increase is greater than TCP Reno, and the decrease on loss is less than TCP Reno

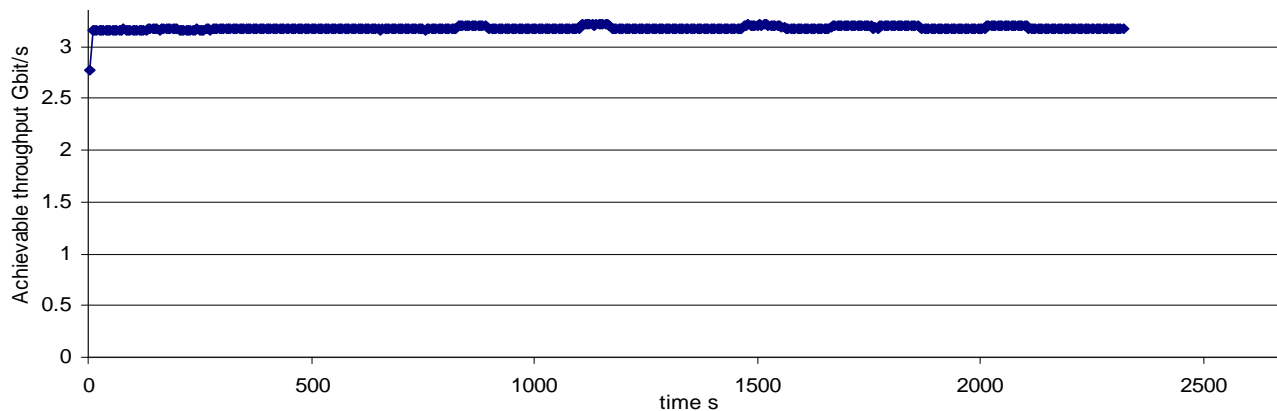


Solution #2 Scalable TCP

Number of 2 gigabytes transferred achieved in 1200s

Number of flows	2.4.19 TCP	2.4.19 TCP & giga-bit device buffer	Scalable TCP
1	7	16	44
2	14	39	93
4	27	60	135
8	47	86	140
16	66	106	142

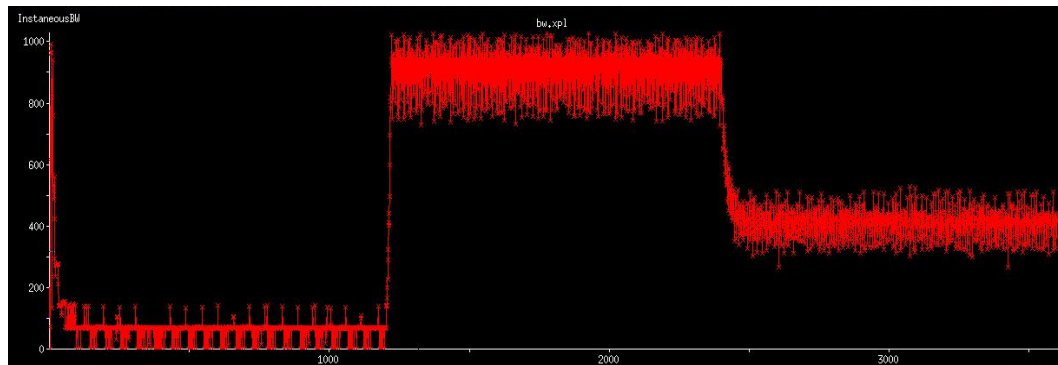
Single **Scalable TCP** connection over a **10 Gbits/s** link



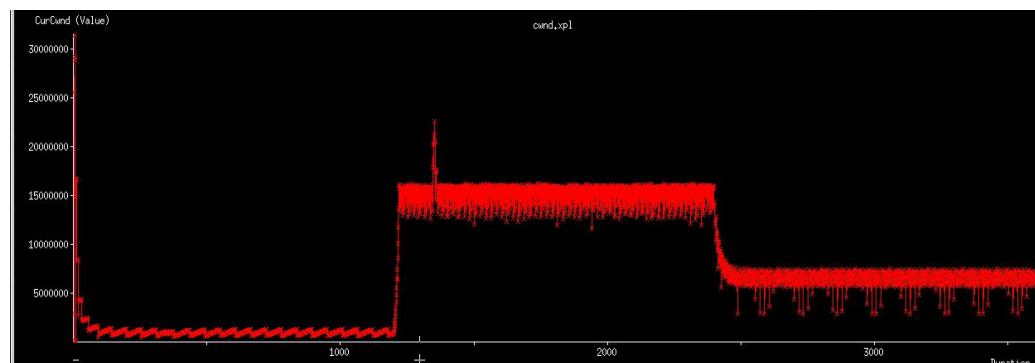


TCP – Comparing Proposals [1]

A **single TCP flow** over the DataTAG testbed over a one hour period. For the **first 20 minutes** the kernel was run **with the standard TCP** stack, for the **second 20 minute** period the kernel was switched to **Scalable TCP**, and for the **final 20 minutes** the kernel ran **High Speed TCP**. The test was set such that **continuous loss** was induced in the network at a rate of **1 packet every 30,000 packets**



Throughput against time

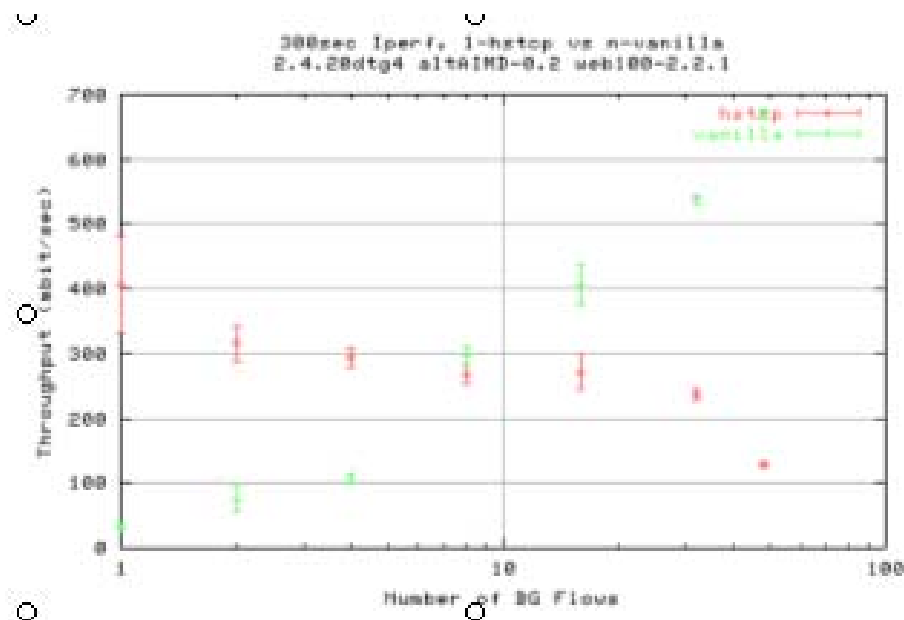
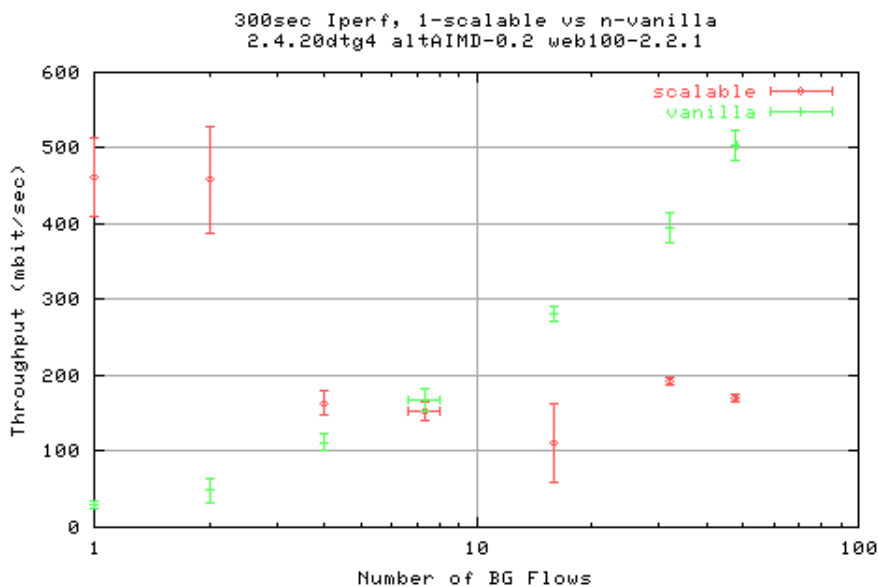


Measured congestion window against time



TCP – Comparing Proposals [2]

Comparison of throughput between a single Scalable TCP flow (left, red) and a single High Speed TCP flow (right, red) and an increasing number of standard TCP flows (green). This suggests that both Scalable and High Speed TCP are broadly equivalent to eight standard TCP flows

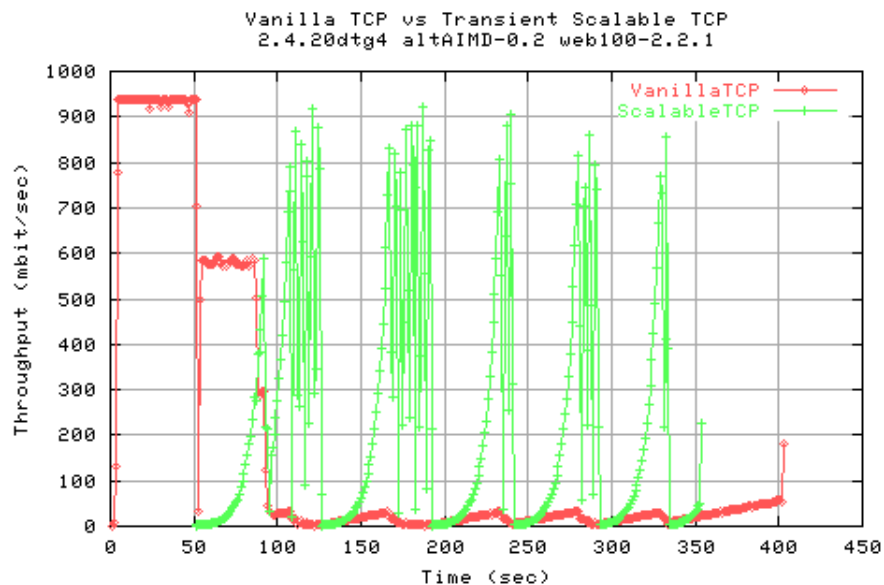
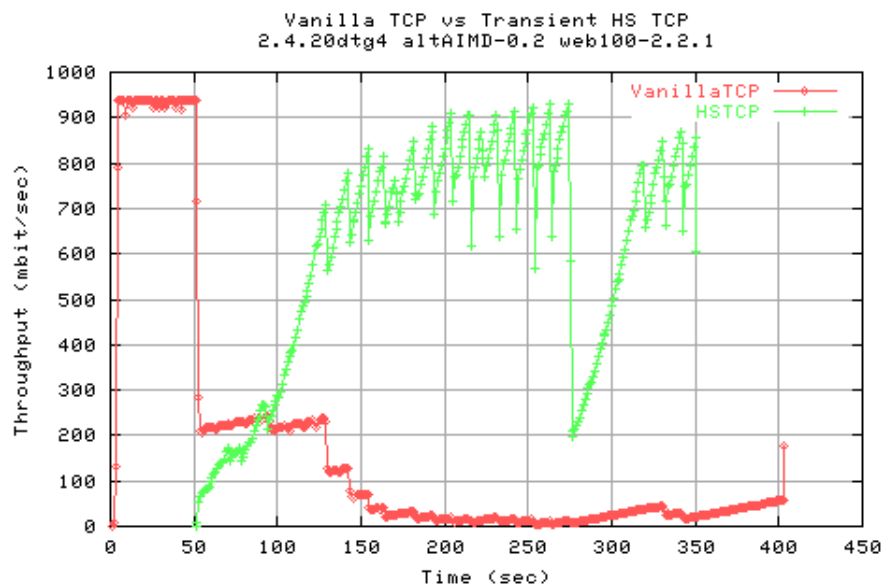




TCP – Comparing Proposals [3]

A notional measurement of **TCP fairness** for High Speed TCP (left) and Scalable TCP (right) against standard TCP.

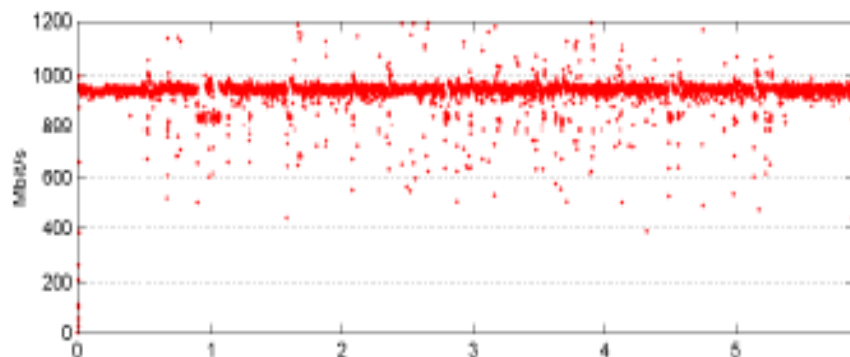
This show the differences between the AIMD approaches of High Speed TCP (left) and Scalable TCP (right). Scalable TCP not only causes the standard TCP flow to decrease its throughput more quickly, but is also a lot more variable in its throughput for the duration of the connection.



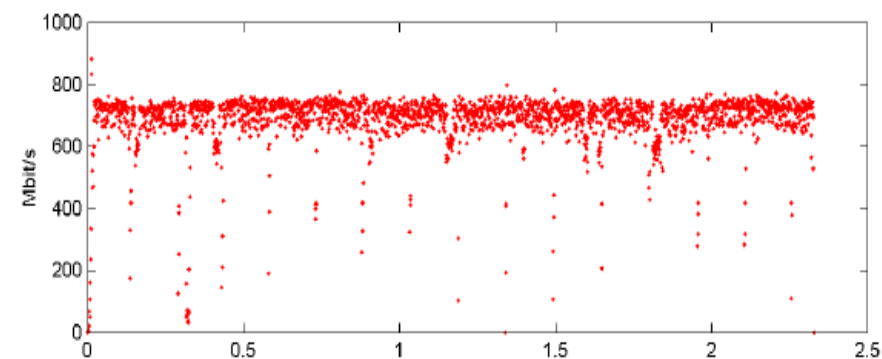


User Applications!!

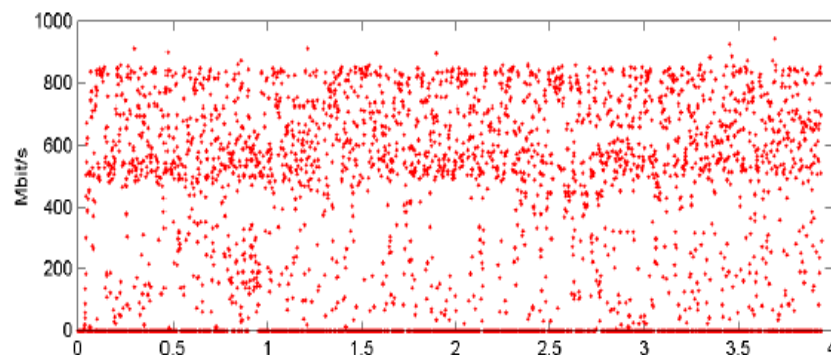
High Speed TCP transfer
using Iperf, i.e. null
application



Web100 records of High
Speed TCP during a http-
Get data transfer



Web100 records of High
Speed TCP during a
GridFTP data transfer





Questions?

ARGONNE



STARLIGHTSM

UIC

<http://www.datatag.org>



DataTAG is a project sponsored by the European Commission - EU Grant IST-2001-32459

RIPE-47, Amsterdam, 29 January 2004