SC 2000: High-Performance Networking & Computing
Dallas, Texas, USA, November 4-10, 2000

Is TCP an Adequate Protocol for *High-Performance Computing* Needs?

Wu-chun Feng
feng@lanl.gov
http://home.lanl.gov/feng

Computer & Computational Science (CCS) Division
Los Alamos National Laboratory
and
School of Electrical & Computer Engineering
Purdue University

Q & A

Q: Is TCP an adequate protocol for high-performance computing (HPC) needs?

A: No!

Q: Can TCP be made into an adequate protocol for high-performance computing needs?

A: Maybe.

Q: What is the networking environment for HPC?

A: System-area network (or LAN) for cluster computing.
Wide-area network for computational grid.

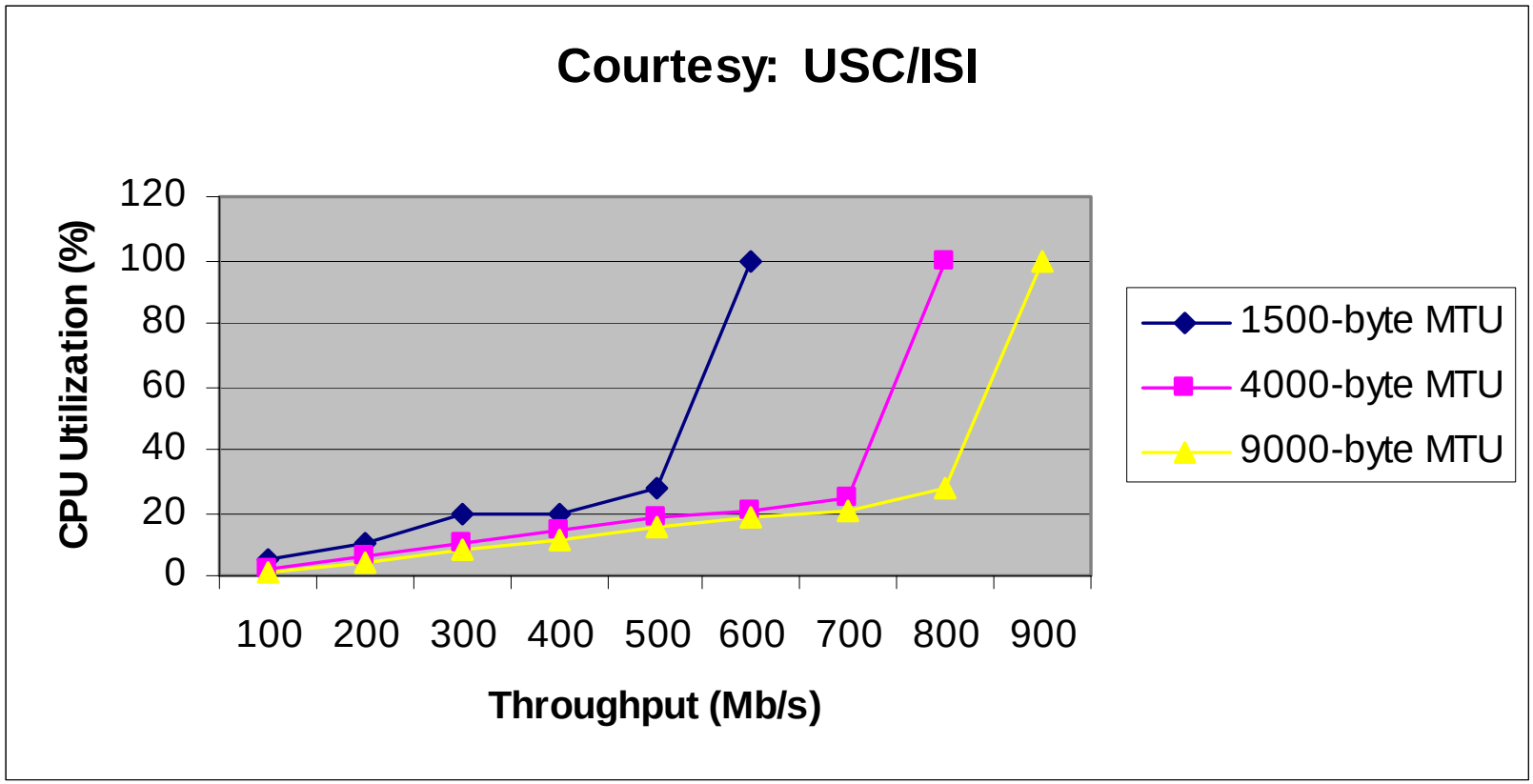
What's Wrong with TCP?

- Host-Interface Bottleneck
  - Software
    - A host can only send and receive packets as fast as the OS can process the packets.
  - [Hardware (PC) *Not anything wrong with TCP per se.*
    - PCI I/O bus. 64 bit, 66 MHz = 4.2 Gb/s. Solution: InfiniBand?]
- Adaptation Bottlenecks
  - Flow Control
    - No adaptation currently being done in any standard TCP.
    - Static-sized window/buffer is supposed to work for both the LAN and WAN.
  - Congestion Control
    - Adaptation mechanisms will *not* scale, particularly TCP Reno.

Host-Interface Bottleneck (Software)

- First-Order Approximation
  - deliverable bandwidth = maximum-sized packet / interrupt latency
  - e.g., 1500-byte MTU / 50 μs = 30 MB/s = 240 Mb/s
- Problems
  - Maximum-sized packet (or MTU) is only 1500 bytes for Ethernet.
  - Interrupt latency to process a packet is quite high.
  - CPU utilization for network tasks is too high.
- Solutions (Non-TCP, Non-Standard)
  - Reduce frequency of interrupts, *e.g., interrupt coalescing or OS-bypass*
  - Increase effective MTU size, *e.g., interrupt coalescing or jumbograms.*
  - Reduce interrupt latency, *e.g., push checksums into hardware.*
  - Reduce CPU utilization, *e.g., offload protocol processing to NIC.*

666-MHz Single CPU Alpha Linux



Note: The congestion-control mechanism does *not* get activated in these tests.

Non-TCP, Non-Standard Solutions

- Interrupt Coalescing
  - Increases bandwidth (BW) at the expense of even higher latency.
- Jumbograms
  - Increases BW with minimal increase in latency, but at the expense of more blocking in switches/routers.
- OS-Bypass Protocol
  - Increases BW & decreases latency by an order of magnitude or more.
  - Integrate OS-bypass into TCP?
    VIA over TCP (IETF Internet Draft, GigaNet, July 2000).
- Interrupt Latency Reduction (possible remedy for TCP)
  - Provide "zero-copy" TCP (*a la* OS-bypass) but OS still middleman.
  - Push protocol processing into hardware, e.g., checksums.

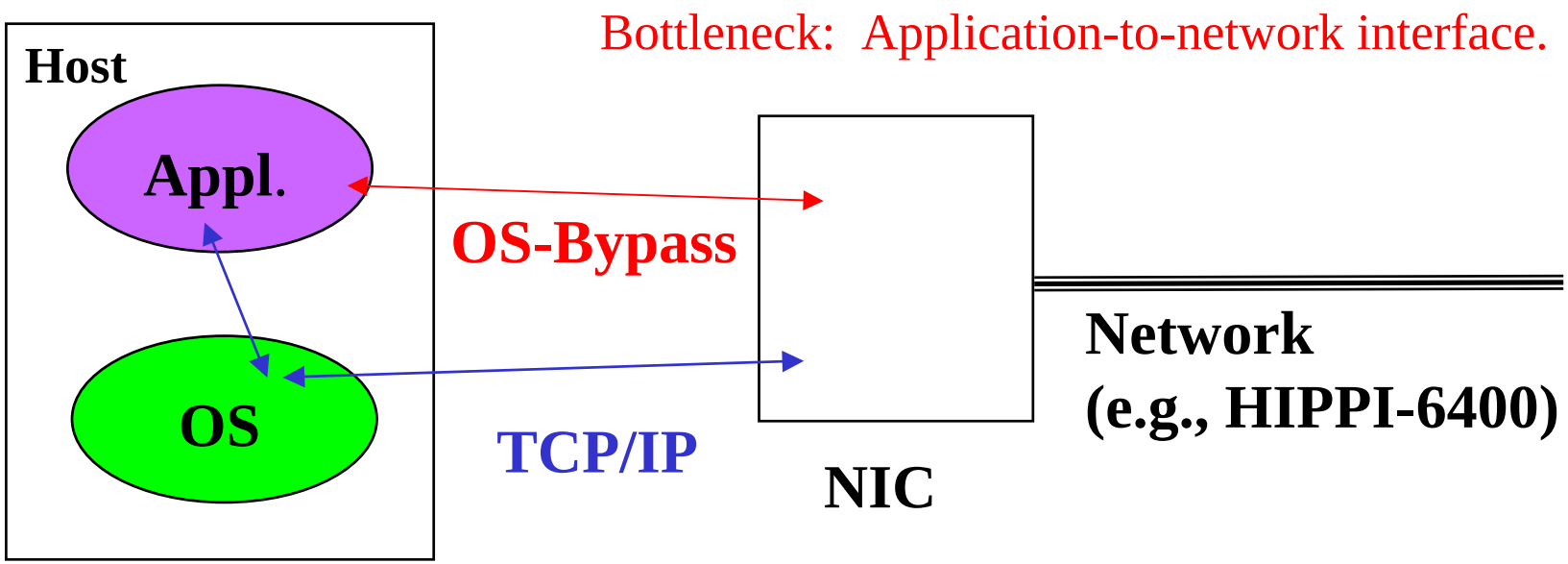
Benchmarks: TCP

- TCP over Gigabit Ethernet (via loopback interface)
  - Theoretical Upper-Bound: 750 Mb/s due to the nature of TCP Reno.
  - Environment: Red Hat Linux 6.2 OS on 400-MHz & 733-MHz Intel PCs; Alteon AceNIC GigE cards; 32-bit, 33-MHz PCI bus.
  - Test: Latency & bandwidth over loopback interface.
    - Latency: O(50 μs).
    - Peak BW w/ default set-up: 335 Mb/s (400) & 420 Mb/s (733).
    - Peak BW w/ *manual* tweaks by network gurus at both ends: 625 Mb/s.
      - Change default send/receive buffer size from 64 KB to 512 KB.
      - Enable interrupt coalescing. (2 packets per interrupt.)
      - Jumbograms. *Theor. BW: 18000 / 50 = 360 MB/s = 2880 Mb/s.*
  - Problem: OS is the middleman. Faster CPUs provide slightly less latency and slightly more BW. 10GigE BW for a high-speed connection wasted.

Solution? OS-bypass

Problem? Data copies across mem. bus. (Cong. ctrl.)

OS-Bypass Protocol



- Over the WAN? How would it compare to HP-TCP?
- Problems with OS-Bypass: Routing & congestion control.
  Hence, the proposal for VIA over TCP.

Benchmarks: OS-Bypass

Two orders of magnitude faster wrt latency and one wrt BW (when compared to TCP).

- GM over Myrinet 2000 Interconnect
  - Peak Bandwidth: 2.0 Gb/s.
  - User-Level *(Reference: Myrinet web site & brochure.)*
    - Latency: 9 μs.
    - Bandwidth: 225 MB/s = 1.8 Gb/s.
- Elan OS-Bypass Library over Quadrics Interconnect
  - Peak Bandwidth: 3.2 Gb/s.
  - User-Level *(Reference: Petrini, Hoisie, Feng, & Graham.)*
    - Latency: 1.9 μs.
    - Bandwidth (unidirectional): 307 MB/s = 2.5 Gb/s.

All is not rosy. Flow control but no congestion control. Manually configured routing tables.

Adaptation Bottleneck

- Flow Control
  - Issues
    - No adaptation currently being done in any standard TCP.
    - 32-KB static-sized window/buffer that is supposed to work for both the LAN and WAN.
  - Problem: Large bandwidth-delay products require flow-control windows as large as 512-KB or 1024-KB to fill the network pipe.
  - Consequence: As little as 3% of network pipe is filled.
  - Solutions
    - *Manual* tuning of buffers at send and receive end-hosts.
    - *Automatic* tuning of buffers.
      - PSC: Auto-tuning but does not abide by TCP semantics, 1998.
      - LANL: Dynamic right-sizing, 2000.
    - *Network striping & pipelining* w/ default buffers. UIC, 2000.

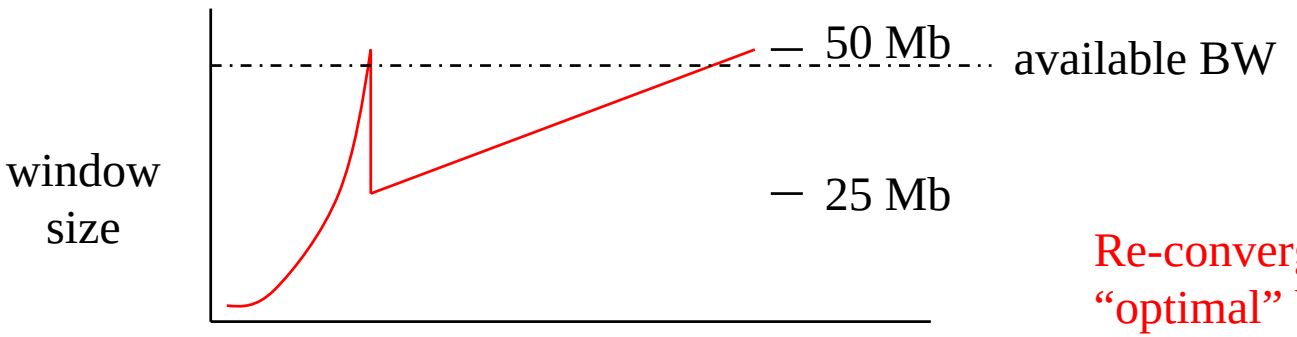
Adaptation Bottlenecks

- Congestion Control
  - Adaptation mechanisms will *not* scale due to
    - Additive increase / multiplicative decrease algorithm
    - TCP Reno congestion control
      - Bad: Allow/induce congestion.
        Detect & recover from congestion. (Synch prob.)
        Analogy: "Deadlock detection & recovery" in OS.
      - Result: At best, 75% utilization in steady state.
    - TCP Vegas congestion control
      - Better: Approach congestion but try to *avoid* it.
        Usually results in better network utilization.
        Analogy: "Deadlock avoidance" in OS.

“Optimal” Bandwidth

- The future performance of computational grids looks bad if we continue to rely on the widely-deployed TCP Reno.

  Example: High BW-delay product: 1 Gb/s WAN * 100 ms RTT = 100 Mb
- Additive increase
  - when window size is 1 ⟶ 100% increase in window size.
  - when window size is 1000 ⟶ 0.1% increase in window size.



Re-convergence to “optimal” bandwidth takes over 3 minutes!

What's Wrong with TCP?

- Host-Interface Bottleneck
  - Software
    - A host can only send and receive packets as fast as the OS can process the packets.
  - Hardware (PC)
    - PCI I/O bus. 64 bit, 66 MHz = 4.2 Gb/s. Solution: InfiniBand?

BW problems potentially solvable. Latency?

Based on past trends, the I/O bus will continue to be a bottleneck.

- Adaptation Bottlenecks
  - Flow Control
    - No adaptation currently being done in any standard TCP.
    - Static-sized window/buffer is supposed to work for both the LAN and WAN.
  - Congestion Control
    - Adaptation mechanisms will *not* scale, particularly TCP Reno.

Solutions exist but are not widely deployed.

TCP Vegas for high-performance TCP?

That’s all folks!