


TCP BYPASS

STAC Research Performance Summit – London June 2011

Richard.Croucher@informatix-sol.com

www.informatix-sol.com

- 1969 - BBN implement first packet switching network control programs on custom hardware to support ARPANET which were to evolve into the first IP routers
- 1973 - TCP/IP defined V. Cerf , E. Cain and R. Kahn as part of ARPANET program funded by the US DoD to enable interconnection of different networks
- 1983 - University of California, Berkeley add TCP/IP networking , sockets and rcommands to UNIX 4.2 BSD
 - Basis for most host implementations and defines standard for network programming
 - TCP/IP one of several protocols alongside UNIX domain sockets, XNS, X.25 and HYPERchannel
- 1983 - Sun Microsystems ships Sun-2, first commercial system with TCP/IP and Ethernet built in as standard . They were to later add NFS , Yellowpages (NIS) and RPC's.
- 1985 - Dennis Ritchie, Bell Labs, creates kernel streams to allow easy implementation of protocol stacks as part of UNIX Eighth Edition
- 1989 - Sun and AT&T jointly implement streams based TCP/IP stack as part of SVR4 development and for use in Solaris 2. (They also implemented a full ISO stack).
- 1992 – Linus Torvolds posts Linux version 0.98 which included TCP/IP networking
- 1994 - Microsoft release it's own Winsocket TCP/IP add-on implementation for Windows 3.11 joining several 3rd party packages already available. It would not become standard until Windows 95 was released the following year
- 2001 Linux community re-implement TCP/IP stack to improve performance with version 2.4
- 2004 - Sun replaces Steams TCP/IP stack with FireEngine to improve performance as part of Solaris 10
- 2007 - Microsoft re-implement TCP/IP stack as part of win6 (Svr 2008) to improve performance

- Zero copy
 - Avoid copy from User space to kernel space, DMA from/to User space. Issues with holding onto buffers to support TCP retransmission requests, no current API/signalling to tell User space code when it can discard buffer. Implementations available but not in mainstream distributions.
 - Only the webserver Use case of Filecopy write to socket available in mainstream since this can DMA from the file buffer cache and can be called directly from the TCP code to release the buffer. Note syntax differences between UNIX implementations cause portability issues.
 - NIC optimizations
 - Interrupt coalescing. Reduces CPU load and can therefore increase throughput where this is a constraint but adversely impacts latency
 - NAPI support lowers CPU load by disabling RX interrupts and only polling under high load. Adaptive; resorts back to RX interrupts when load is lowered
 - Scatter gather I/O transfers from multiple blocks in a single DMA operation avoiding a kernel memory copy up to the 64K allowed for an IP packet
 - Offloads for TCP Segmentation, checksums, Large Receive
 - Receive Side Scaling (RSS) spreads RX load across multiple CPUs
 - TCP Offload engines
 - User Space TCP/IP implementations
 - Bypass TCP altogether
- 

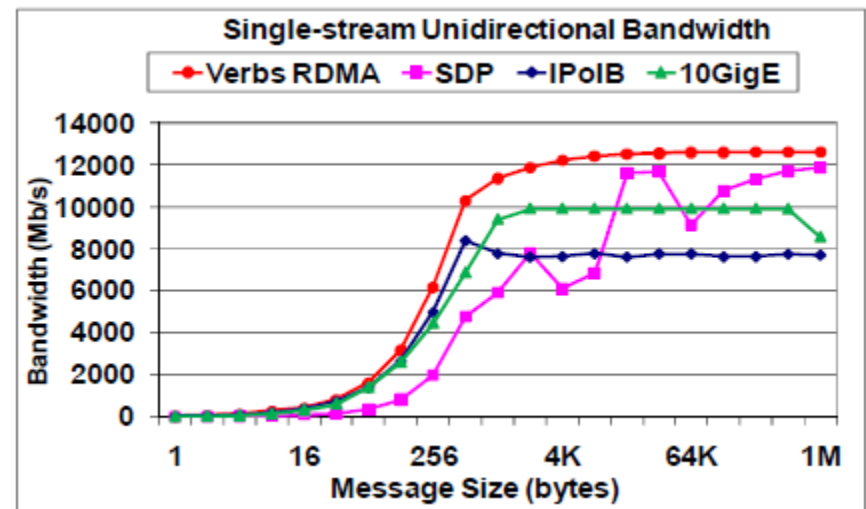
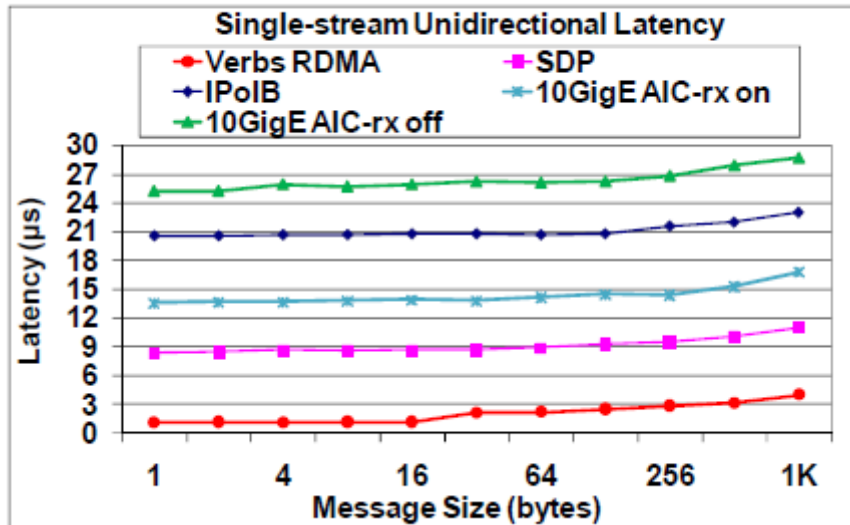
- Leveraging NIC processing power to reduce host CPU demand
- Original Patents filed by Larry Boucher (Auspex later Alacritech) in 1990
- Typically no performance improvement since NIC CPU is not as powerful as host, although latest 10G engines look promising
- Issues with how to integrate with existing host stack and host based firewalls – referred to as Parallel Offload
- Microsoft licensed Offload Patents and supports it with the “Chimney” implementation in Windows Svr 2008
 - Fully integrated stack
 - Dependencies on NIC driver quality
- Linux community decided against integrated stack and only support presenting the offload engine as an iSCSI initiator device or for iWARP
 - Avoids host stack integration issues
 - Limits offload to block storage devices and RDMA
- Independent TCP/IP stacks in FPGA or co-processors could terminate session locally but still need to communicate across PCIe to host

- Developed initially to allow easier experimentation with TCP/IP protocols – e.g. Daytona
- Later adapted to leverage zero copy and maintain CPU context
 - DMA's directly to/from User memory
- Challenge is integration with kernel stack and state management
- www.Openonload.org developed by David Riddoch, now of Solarflare
 - Comfortably achieves single threaded wire speed on 10Ge
 - Linux only, and currently limited to Solarflare HW
 - Measured median latency of 8.5µS RTT across crossover cables
 - Constraints (ver 20101111-u1):
 - Trades CPU for lowest latency, RX spin wait for lowest latency burns a thread for each session
 - No Broadcast. No Multicast within same host, single (offload receiver) for each MC group (unless sharing same offload User stack) –transparently moves to kernel stack
 - Bonding only between (offloaded) Solarflare ports
 - No ipfilters, netstat, tcpdump, Wireshark

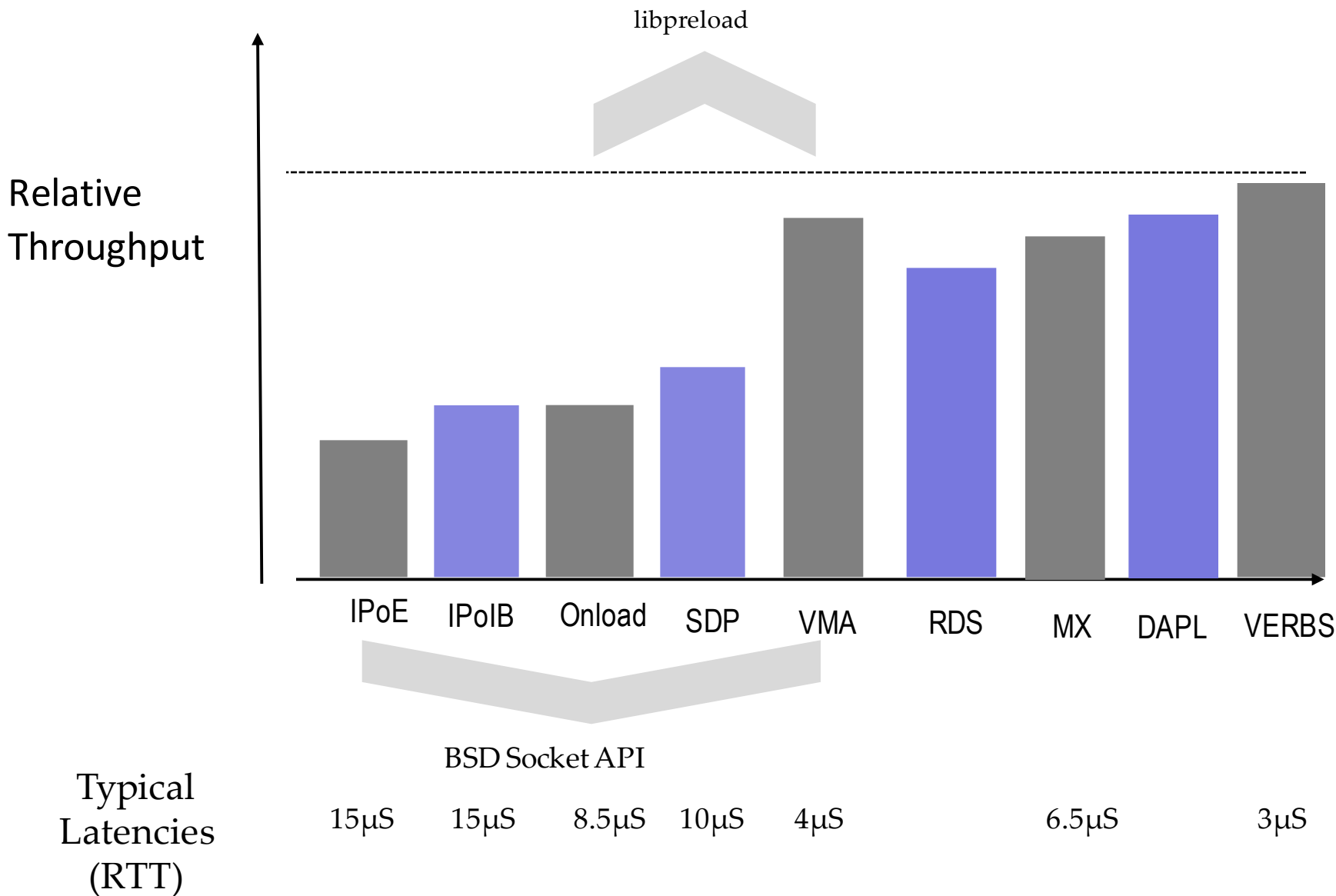
- TCP provides error detection and correction, in-order delivery of data, flow control and congestion management
- IP provides routing, packet chopping and aggregation, error detection and a address namespace
- IP allows us to scale by routing across multiple L2 subnets
- We can only replace TCP/IP by finding alternatives to these
- By limiting to Layer2 networks we can drop IP but need to provide a namespace and still need to overcome any scalability issues
- Ethernet subnets are rarely extended beyond 1024 addresses due to broadcast storm issues
- Ethernet traditionally did not guaranteed delivery and is permitted to arbitrarily drop packets. Only the new DCB standard provides delivery guarantees (for specific Ethertypes e.g. FCoE mapped to no drop PFC)

- GAMMA – Genoa Active Message Machine
- Open-MX – Myricom API
- MPI – Message Passing Interface
- Open Fabric Alliance Enterprise Distribution (OFED)
 - Transports:
 - InfiniBand – Converged interconnect
 - RoCEE – RDMA over Converged Ethernet
 - iWARP – RDMA over TCP/IP/Ethernet
 - API's:
 - IPoIB, RDS, SDP, SRP, iSER, VERBS
 - Used to support external API's and subsystems:
 - DAPL, NFS/RDMA, MPI, Lustre, IBM GPFS, Mellanox VMA
 - Used by various middleware components including:
 - TIBCO FTL, IBM Websphere LLM, RNA networks, Oracle RAC, IBM DB2
 - Integrated into Linux kernel since 2.6.14.
 - Only RDMA subsystem included.

Latency and throughput by protocol

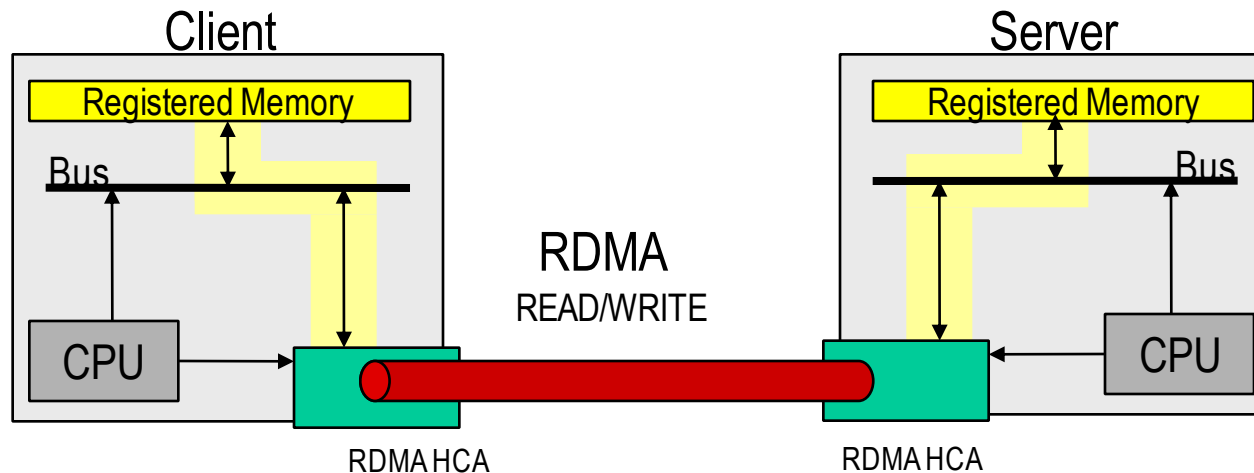


“Evaluation of ConnectX Virtual Protocol Interconnect for Data Centers”;
 R. Grant, A. Afsahi, P. Balaji; Ontario University and Argonne National Laboratory
 Testing same ConnectX card in both 40g InfiniBand and 10g CNEE modes with Jumbo frames for throughput



- VERBS initially defined in the InfiniBand standard (IBA) but also supported in RoCEE as a result of its InfiniBand encapsulation across Ethernet.
Subset of VERBs supported across iWARP
- Libibverbs API (Linux, Windows, some UNIX)
 - Wrapping of native VERBS into 'C' library functions
 - Requires native addressing LID/GID/GUID which can be cumbersome
- RDMA_cm API (Linux only)
 - Connection management library for VERBs programs
 - Leverages TCP/IP namespace to identify endpoints
 - Alternate Path Management to enable host failover between interfaces
- Initial code complexity comparison with TCP and UDP sockets

Program name	Description	NOM	LOC	MVG	L_C	rel-LOM	rel-MVG	Avg
ping-sr-4	UNH IOL VERB level ping-pong	17	1951	265	2.331	2.240	1.828	2.034
ping-rdma-4	UNH IOL RDMA ping-pong	14	2420	311	2.334	2.778	2.145	2.462
mping	multicast socket ping-pong	2	701	121	3.403	0.805	0.834	0.820
tping	TCP socket ping-pong	2	871	145	2.943	1	1	1



- Removes CPU from being bottleneck
- User space to User space remote copy - after memory registration
- HCA is responsible for virtual-physical → physical-virtual address mapping
- Shared keys created and exchanged for access rights and current ownership
- Memory has to be registered to lock into RAM and initialize HCA's TLB
- Demands explicit buffer management which creates obstacles for Java and C#
- RDMA read uses no CPU cycles after registration on donor side

- Model is Command Queue based with completion events, running on reliable rather than buffered network
 - Send/Receive
 - Send data to remote queue. Receiver has to have previously posted a buffer to receive data (otherwise send fails). Typically used when data can fit inside a single packet.
 - RDMA Read
 - Registered memory region is read from remote host. Completion event signals to reader that data transfer is finished. Donor server is not interrupted. Nothing explicitly prevents donor from using or changing memory
 - RDMA Write
 - Similar to Read, but data is written to remote host. Completion events are sent to Sender to notify that Write has completed and to receiver so that (local) copy may be safely accessed
 - Multicast Send/Receive
 - 1 to many. 16K reserved addresses. IGMP like subscription behaviour. Unreliable – no guaranteed delivery
 - Atomic Compare/Swap and Fetch/Add
 - Used for distributed lock primitives

- Sources of help:
 - Mellanox RDMA Aware programmer Manual
 - http://www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf
 - OFED code examples and documents
 - http://www.openfabrics.org/index.php?option=com_content&view=article&id=19&Itemid=113
 - University New Hampshire, Interoperability lab – RDMA programming class, including 20 example programs
 - http://www.openfabrics.org/index.php?option=com_content&view=article&id=12&Itemid=105
- STAC is conducting API comparisons, due in the vault next quarter
 - Code complexity and detailed performance analysis
 - Source code for API bindings to be available to premium subscribers
- These slides posted to STACresearch.com and www.informatix-sol.com