Engineering a User Level TCP

Kieran Mansley - kjm25@cam.ac.uk

Laboratory for Communication Engineering,

University of Cambridge.

Research Goal:

Reduce the CPU load of network protocol processing.

The Problems

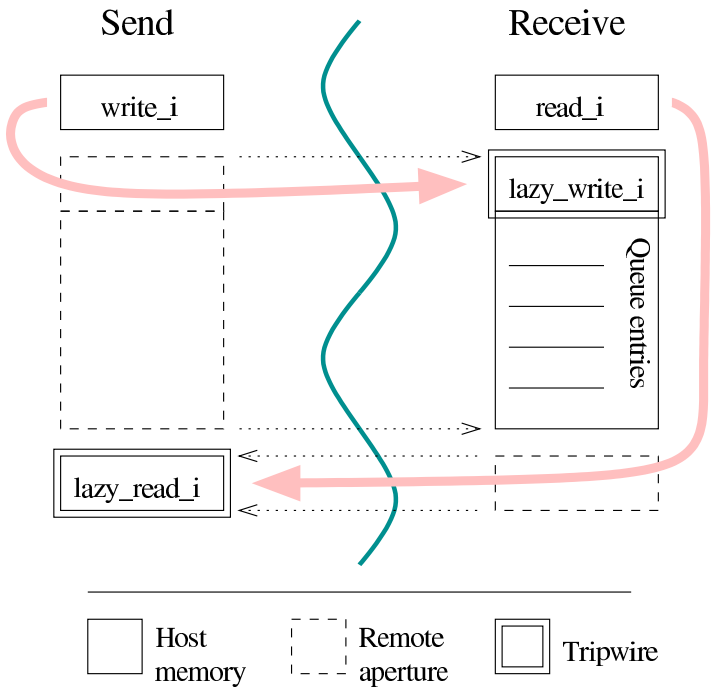
- ✗ Faster networks deliver higher packet rates:
  - 100 byte packets at Gbps ⇒ 1.25 million packets per second.
  - 0.8us to deal with each one.
- ✗ Overhead of dealing with packets isn't decreasing.
- ✗ The BSD sockets API mandates copying the data.
- ✗ CPU cycles required to service network → 100%.

Possible Solutions

- ✓ Increase MTU to reduce the number of packets.
  - ✗ OK for some traffic patterns, but no use for many.
- ✓ Interrupt coallescing to avoid receive livelock.
  - ✗ Adds latency.
- ✓ Coprocessors on network interface.
  - ✗ Expensive, adds latency.
- ✓ Change API to a zero copy compatible one.
  - ✗ Loose compatibility. Hard to get adopted.
- ✓ User Level networking.

The CLAN Network

- Distributed Shared Memory based cluster network.
- Data transfer interface looks like a circular queue.
- Supports both DMA and PIO.
- Prototype hardware.
- Results in:
  - ✓ Low overhead, low latency transfers.
  - ✓ Simple ⇒ cheap & scalable.

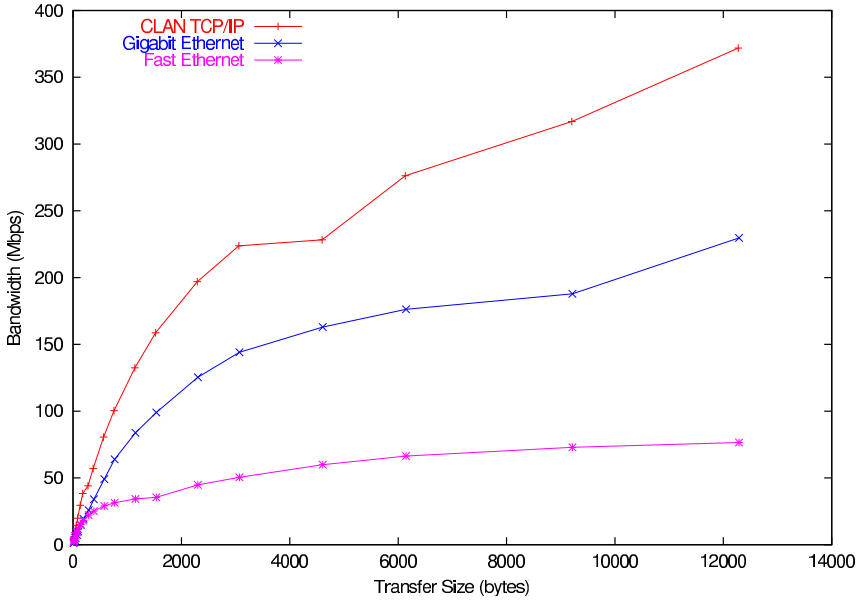


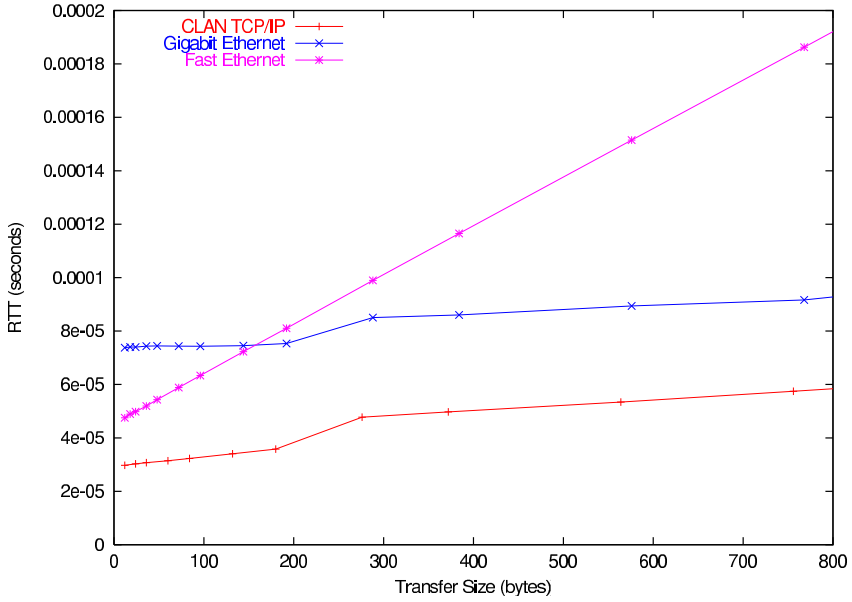
User Level Stack Architecture

- Connection oriented network.
  - ✓ Very simple demultiplex.
  - ✗ Must maintain mapping between connections and sockets.
  - ✗ Connection setup overhead.
- Application thread does protocol processing.
  - ✓ No thread switches on the data path.
  - ✗ Thought needed to ensure everything gets the time it needs.
- Change blocking semantics of sockets.
  - ✓ No application changes.
  - ✗ Doesn't solve everything (in particular, ***`recv()`***).

Performance

- ✓ Bandwidth Increased.
  - Standard NetPIPE test.
- ✓ Latency Reduced.
  - Less than half Gig Ethernet.



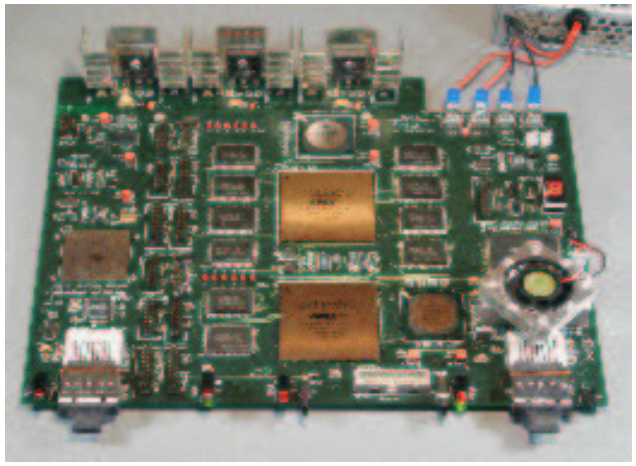


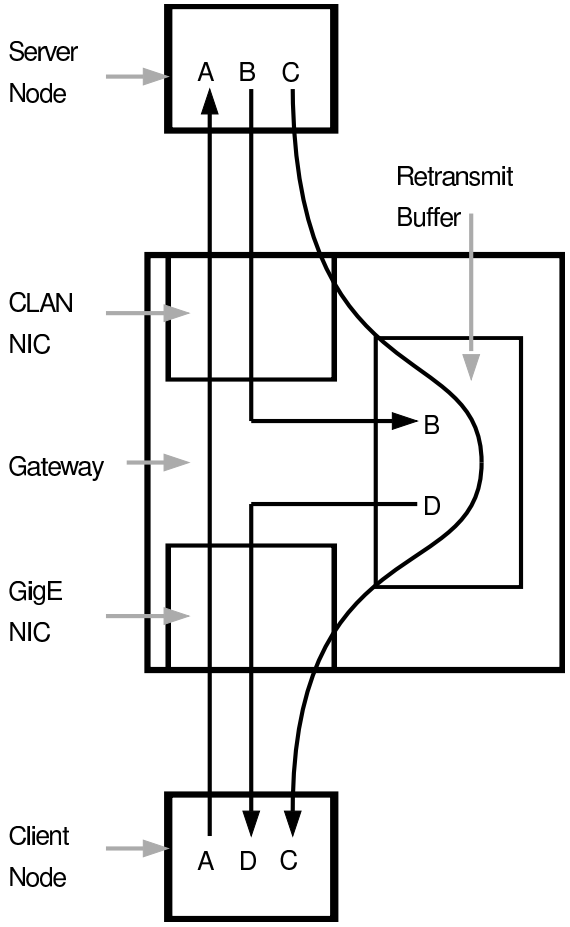
True Zero-Copy Transmission

- Sockets API requires a copy.
  - Otherwise application could overwrite packet before it is sent.
- To achieve zero-copy:
  - Modify blocking semantics of the sockets API.
  - Don't return until stack has finished with the data.
  - ✓ Doesn't impact greatly on performance in most cases.
  - ✓ Copy is now unnecessary.
  - ✗ But, still need to preserve the data for retransmission...

Gateway Assisted Retransmission

- Gateway accesses all packets flowing through it.
  - ✓ Can copy at little extra cost.
  - ✗ But, problems too:
    - Single point of failure?
    - Scalability of memory?





Summary

- Move TCP/IP stack to user level:
  - ✓ Improved efficiency getting data to and from the NIC,
  - ✓ Allows changes to the threading model in the stack which in turn:
  - ✓ Enables zero copy transmission by moving retx to the gateway.
  - ✗ Requires unusual (but simple, potentially cheap) hardware.
- Retransmission by the gateway:
  - Compromise between host based TCP/IP and full offload engine.
  - ✗ Requires additional hardware (i.e. RAM on the gateway).
  - ✓ But... releases resources (i.e. RAM on nodes) elsewhere in cluster.

Future Work

- Full performance measurement and comparison.
- Deal with the copy required for ***`recv()`***.
- TCP Timers:
  - ✗ Many TCPs at user level means many timers.
  - ✓ Improve efficiency by preventing timer firing when there is nothing for it to do.
  - ✓ Allows for more accurate timers (when needed) without extra overhead.
- Load balancing:
  - ✓ Implicit knowledge of the load on cluster nodes from the circular queue pointers.