Frame Shared Memory: Line-Rate Networking on Commodity Hardware

John Giacomoni

John K. Bennett, Douglas C. Sicker, and Manish Vachharajani
University of Colorado at Boulder

Alexander L. Wolf - Imperial College London
Antonio Carzaniga - University of Lugano

2007.12.03
Problem Description

How do we route?
How do we protect?
How do we correlate?

<table>
<thead>
<tr>
<th>Link</th>
<th>Mbps</th>
<th>fps</th>
<th>ns/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-1</td>
<td>1.5</td>
<td>2,941</td>
<td>340,000</td>
</tr>
<tr>
<td>T-3</td>
<td>45.0</td>
<td>90,909</td>
<td>11,000</td>
</tr>
<tr>
<td>OC-3</td>
<td>155.0</td>
<td>333,333</td>
<td>3,000</td>
</tr>
<tr>
<td>OC-12</td>
<td>622.0</td>
<td>1,219,512</td>
<td>820</td>
</tr>
<tr>
<td>GigE</td>
<td>1,000.0</td>
<td>1,488,095</td>
<td>672</td>
</tr>
<tr>
<td>OC-48</td>
<td>2,500.0</td>
<td>5,000,000</td>
<td>200</td>
</tr>
<tr>
<td>10 GigE</td>
<td>10,000.0</td>
<td>14,925,373</td>
<td>67</td>
</tr>
<tr>
<td>OC-192</td>
<td>9,500.0</td>
<td>19,697,843</td>
<td>51</td>
</tr>
</tbody>
</table>
ASIC Solutions

How do we route?
How do we protect?
How do we correlate?

<table>
<thead>
<tr>
<th>Link</th>
<th>Mbps</th>
<th>fps</th>
<th>ns/frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-1</td>
<td>1.5</td>
<td>2,941</td>
<td>340,000</td>
</tr>
<tr>
<td>T-3</td>
<td>45.0</td>
<td>90,909</td>
<td>11,000</td>
</tr>
<tr>
<td>OC-3</td>
<td>155.0</td>
<td>333,333</td>
<td>3,000</td>
</tr>
<tr>
<td>OC-12</td>
<td>622.0</td>
<td>1,219,512</td>
<td>820</td>
</tr>
<tr>
<td>GigE</td>
<td>1,000.0</td>
<td>1,488,095</td>
<td>672</td>
</tr>
<tr>
<td>OC-48</td>
<td>2,500.0</td>
<td>5,000,000</td>
<td>200</td>
</tr>
<tr>
<td>10 GigE</td>
<td>10,000.0</td>
<td>14,925,373</td>
<td>67</td>
</tr>
<tr>
<td>OC-192</td>
<td>9,500.0</td>
<td>19,697,843</td>
<td>51</td>
</tr>
</tbody>
</table>
Programmable Network Processors

Intel® IXP2855
Multicore Systems

- **GPP Multicore systems**
  - Individual cores less powerful than UP
  - 10s-100s-1000s of cores
  - Full OS & Library Support
  - Asymmetric (Alpha)
  - Heterogeneous (AMD, Intel)
Moore’s Corollary vs. Moore’s Law

CPU Performance (SPEC)

- SPEC Benchmark Suite Performance
  - Predicted vs. actual

Graph Courtesy Tipp Moseley

University of Colorado at Boulder
Core Research Lab
Soft Network Processing
(Soft-NP)
Soft-NP Technique
Frame Generation

Execution Stages for Single Frame

<table>
<thead>
<tr>
<th>Gen</th>
<th>App</th>
<th>OP</th>
</tr>
</thead>
</table>

Generate 1 (Gen)
Application (App)
Output (OP)

Frame 1
Gen (P1)  App (P2)  OP (P3)

Frame 2
Gen (P1)  App (P2)  OP (P3)

Frame 3
Gen (P1)  App (P2)  OP (P3)

Frame 4
Gen (P1)  App (P2)

Frame 5
Gen (P1)

P1 (Gen)  P2 (App)  P3 (OP)

Time

Pipelined Execution
AMD Opteron System Overview

Diagram showing the system overview with cores, caches, and interconnect network.
Data Flow
Frame Generation
Communication Overhead

Available Work Time (%) vs. Stage Length in nanoseconds (including comm)
Communication Overhead

![Graph showing communication overhead over stage length in nanoseconds. The x-axis represents stage length in nanoseconds (including communication), and the y-axis represents available work time as a percentage. The graph shows that as stage length increases, the available work time decreases. There is a vertical dotted line indicating the point where locks begin, approximately at 200 nanoseconds.]
Communication Overhead

- **Hardware**: ≈ 10ns
- **Locks**: ≈ 200ns

### Graph

- **Y-axis**: Available Work Time (%)
- **X-axis**: Stage Length in nanoseconds (including comm)

The graph shows the relationship between stage length and available work time, with key points indicating the performance impact of hardware and locks.
Communication Overhead

Hardware \( \approx 10\text{ns} \)

Lamport \( \approx 160\text{ns} \)

Locks \( \approx 200\text{ns} \)

Available Work Time (%)

Stage Length in nanoseconds (including comm)
Communication Overhead

Hardware ≈ 10ns
FastForward ≈ 28ns
Lamport ≈ 160ns
Locks ≈ 200ns
enqueue(data) {
    lock(queue);
    if (NEXT(head) == tail) {
        unlock(queue);
        return EWOULDBLOCK;
    }
    buffer[head] = data;
    head = NEXT(head);
    unlock(queue);
    return 0;
}

enqueue_fastforward(data) {
    if (NULL != buffer[head]) {
        return EWOULDBLOCK;
    }
    buffer[head] = data;
    head = NEXT(head);
    return 0;
}

• Cache-optimized CLF queues
• Works with strong to weak consistency models
• Hides die-die communication
Frame Shared Memory (FShm)

- Pure software stack communicating via shared memory
  - Abstracted at the driver/NIC boundary
  - Cross-Domain modules (Kernel/Process, T/T, P/P, K/K)
  - Compatible with existing OS/library/language services
  - Can communicate with any device on the memory interconnect
struct ifdirect {
    void          (*if_direct_tick) (void *softc);
    void          (*if_direct_attach) (struct ifnet *, void *);
    void          (*if_direct_detach) (struct ifnet *, void *);
    int           (*if_direct_tx) (void *softc, struct mbuf *txbuf);
    void          (*if_direct_tx_post) (void *softc);
    void          (*if_direct_tx_clean_pre) (void *softc);
    struct mbuf*  (*if_direct_tx_clean) (void *softc);
    void          (*if_direct_tx_clean_post) (void *softc);
    void          (*if_direct_rx_pre) (void *softc);
    struct mbuf*  (*if_direct_rx) (void *, struct mbuf *new_rxbuf);
    void          (*if_direct_rx_post) (void *softc);
};
Evaluation Methodology

- AMD Opteron 2.0 GHz
  - Dual-Processor & Dual-Core
- Compute average time per call
  - TSC
FShm Generate
(linux pktgen)

\[ 64B^* \approx 1.36 \text{ Mfps} \]
FShm Capture (IDS)

64B* \approx 1.36 \text{ Mfps}

Frame Size (Bytes)
FShm Forward (Bridge)

64B* \approx 1.36 \text{ Mfps}

Frame Size (Bytes)

nanoseconds

64B* 74B 96B 128B 256B 512B 1024B 1518B
FShm’s Future

- Hardware $≈ 10\text{ns}$
- FastForward $≈ 28\text{ns}$
- Lamport $≈ 160\text{ns}$
- Locks $≈ 200\text{ns}$
Questions?

john.giacomoni@colorado.edu