Outline

- Overview of Network Processors
- Network Processor Architectures
- Applications
- Case Studies
  - Wireless Mesh Network
  - a Content-Aware Switch
- Conclusion
Packet Processing in the Future Internet

Future Internet

More packets & Complex packet processing

ASIC

General-Purpose Processors

- High processing power
- Support wire speed
- Programmable
- Scalable
- Optimized for network applications
- ...

12/18/05 Yan Luo, CAR of UML
What is Network Processor?

- Programmable processors optimized for network applications and protocol processing
  - High performance
  - Programmable & Flexible
  - Optimized for packet processing
- Main players: AMCC, Intel, Hifn, Ezchip, Agere

## Commercial Network Processors

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Product</th>
<th>Line speed</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMCC</td>
<td>nP7510</td>
<td>OC-192/10 Gbps</td>
<td>Multi-core, customized ISA, multi-tasking</td>
</tr>
<tr>
<td>Intel</td>
<td>IXP2850</td>
<td>OC-192/10 Gbps</td>
<td>Multi-core, h/w multi-threaded, coprocessor, h/w accelerators</td>
</tr>
<tr>
<td>Hifn</td>
<td>5NP4G</td>
<td>OC-48/2.5 Gbps</td>
<td>Multi-threaded multiprocessor complex, h/w accelerators</td>
</tr>
<tr>
<td>EZchip</td>
<td>NP-2</td>
<td>OC-192/10 Gbps</td>
<td>Classification engines, traffic managers</td>
</tr>
<tr>
<td>Agere</td>
<td>PayloadPlus</td>
<td>OC-192/10 Gbps</td>
<td>Multi-threaded, on-chip traffic management</td>
</tr>
</tbody>
</table>
Typical Network Processor Architecture

SDRAM (e.g. packet buffer)

SRAM (e.g. routing table)

Network interfaces

Co-processor

H/w accelerator

Network Processor
Snapshots of IXP2xxx Based Systems

ADI Roadrunner Platform
- IPv4 Forwarding/NAT
- Forwarding w/ QoS / DiffServ
- ATM RAN
- IP RAN
- IPv6/v4 dual stack forwarding

Radisys ENP2611 PCI Packet Processing Engine
- multiservice switches,
- routers, broadband access devices,
- intrusion detection and prevention (IDS/IPS)
- Voice over IP (VoIP) gateway
- Virtual Private Network gateway
- Content-aware switch
Intel IXP425 Network Processor

- Media Independent Interface
- 66 MHz Advanced Peripheral Bus
- UART 921K baud
- Interrupt Controller
- Timers
- GPIO Controller
- USB Controller
- PMU (AHB)
- Intel XScale® Core
- 286/400/533 MHz
- 32-KB Data Cache
- 32-KB Instruction Cache
- 2-KB Mini-Data Cache
- 133 MHz Advanced High-Performance Bus
- Queue Manager 8-KB SRAM
- SDRAM Controller 8 MB-256 MB
- PCI Controller
- Exp. Bus Controller
- Test Logic Unit
- JTAG
- 16 Pins
- 32-bit
- 16-bit
- Queue Status Bus
- 130 MHz Advanced High-Performance Bus
- Arbiter
- Arbiter
- Bridge
- WAN/Voico NPE
- UTOPIA
- (Max 24 xDSL PHYs)
- AAL, HSS, HDLC
- Ethernet NPE A
- Ethernet MAC
- Ethernet NPE B
- Ethernet MAC
- SHA-1/MD5, DES, 3DES, AES
- HSS-0
- HSS-1
- UTOPIA 2
- 32-bit
- UART 921K baud
- 66 MHz Advanced Peripheral Bus
- 16 Pins
- JTAG
- 32-bit
- 16-bit
StarEast: IXP425 Based Multi-radio Platform
Applications of Network Processors

Core router

Edge router

Internet

DSL modem

Wireless router

VoIP terminal

VPN gateway

Printer server
Case Study 1: Wireless Mesh Network
Software Stack on StarEast

Customer Applications

Seamless Networking  Mesh  Cognitive Radio

ARA/RAL (Radio/MAC Abstraction Layer)

IXP425 Access Library  CardBus  miniPCI
NPE A  NPE B  USB Slave  Netgear WAG511  Intel PRO 100  Prism2

Kernel

 miniPCI
loctl()  HostAP  Driver  FW/decode  Card

User

Bootsloader (Redboot)

NPE A  NPE B  UART  Intel PRO100

Stareast Hardware Platform
Case Study 2: Content-aware Switch

- Front-end of a Web cluster, only one Virtual IP
- Route packets based on Layer 5 information
  - Examine application data in addition to IP & TCP
- Advantages over layer 4 switches
  - Better load balancing: distributed based on content type
  - Faster response: exploit cache affinity
  - Better resource utilization: partition database
Mechanisms to Build a Content-aware Switch

- **TCP gateway**
  - An application level proxy
  - Setup 1st connection w/ client, parses request → server, setup 2nd connection w/ server
  - Copy overhead

- **TCP splicing**
  - Reduce the copy overhead
  - Forward packet at network level between the network interface driver and the TCP/IP stack
  - Two connections are spliced together
  - Modify fields in IP and TCP header
Anatomy of TCP Splicing

Without TCP Splicing

With TCP Splicing

- SEQ # translation
- Checksum Recalculation
- Etc.

Bookkeeping of connection states, selection of servers, state migration
Design Options

• Option 0: GP-based (Linux-based) switch
• Option 1: CP setup & and splices connections, DPs process packets sent after splicing
  Connection setup & splicing is more complex than data forwarding
  Packets before splicing need to be passed through DRAM queues
• Option 2: DPs handle connection setup, splicing & forwarding
IXP 2400 Block Diagram

- XScale core
- Microengines (MEs)
  - 2 clusters of 4 microengines each
- Each ME
  - run up to 8 threads
  - 16KB instruction store
  - Local memory
- Scratchpad memory, SRAM & DRAM controllers
Resource Allocation

**SRAM (8MB)**
- Client side CB list
- Server side CB list
- Server selection table
- Locks

**DRAM (256MB)**
- Packet buffer

**Scratchpad (16KB)**
- Packet queues

**Client-side control block list**
- Record states for connections between clients and SpliceNP, states after splicing

**Server-side control block list**
- Record states for connections between server and SpliceNP

**Microengines**

- Rx ME
- Client ME
- Server ME
- Tx ME
### Comparison of Functionality

A lite version of TCP due to the limited instruction size of microengines.

#### Processing a SYN packet

<table>
<thead>
<tr>
<th>Step</th>
<th>Functionality</th>
<th>TCP</th>
<th>Linux Splicer</th>
<th>SpliceNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dequeue packet</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>IP header verification</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>IP option processing</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>TCP header verification</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>Control block lookup</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>Create new socket and set state to LISTEN</td>
<td>Y</td>
<td>Y</td>
<td>No socket, only control block</td>
</tr>
<tr>
<td>7</td>
<td>Initialize TCP and IP header template</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>Reset idle time and keep-alive timer</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>9</td>
<td>Process TCP option</td>
<td>Y</td>
<td>Y</td>
<td>Only MSS option</td>
</tr>
<tr>
<td>10</td>
<td>Send ACK packet, change state to SYN_RCVD</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Processing a SYN packet
Experimental Setup

- Radisys ENP2611 containing an IXP2400
  - XScale & ME: 600MHz
  - 8MB SRAM and 128MB DRAM
  - Three 1Gbps Ethernet ports: 1 for Client port and 2 for Server ports
- Server: Apache web server on an Intel 3.0GHz Xeon processor
- Client: Httperf on a 2.5GHz Intel P4 processor
- Linux-based switch
  - Loadable kernel module
  - 2.5GHz P4, two 1Gbps Ethernet NICs
Latency on a Linux-based TCP Splicer

Latency is reduced by TCP splicing
Latency vs Request File Size

- Latency reduced significantly
  - 83.3% (0.6ms → 0.1ms) @ 1KB
- The larger the file size, the higher the reduction
  - 89.5% @ 1MB file
# Comparison of Packet Processing Latency

## Table 5: Processing latency for control and data packets

<table>
<thead>
<tr>
<th>Packet Type</th>
<th>IXP2400 Microengine</th>
<th>Latency (us)</th>
<th>Linux Latency (us)</th>
<th>Latency reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>SYN</td>
<td>clientME</td>
<td>7.2</td>
<td>48</td>
</tr>
<tr>
<td>Packet</td>
<td>ACK/Request</td>
<td>clientME</td>
<td>8.8</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>SYN/ACK</td>
<td>serverME</td>
<td>8.5</td>
<td>42</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
<td>serverME</td>
<td>6.5</td>
<td>13.6</td>
</tr>
<tr>
<td>Packet</td>
<td>ACK</td>
<td>clientME</td>
<td>6.5</td>
<td>13.6</td>
</tr>
</tbody>
</table>
## Analysis of Latency Reduction

<table>
<thead>
<tr>
<th>Linux-based</th>
<th>NP-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt: NIC raises an interrupt once a packet comes</td>
<td>polling</td>
</tr>
<tr>
<td>NIC-to-mem copy</td>
<td>No copy: Packets are processed inside without two copies</td>
</tr>
<tr>
<td>Xeon 3.0Ghz Dual processor w/ 1Gbps Intel Pro 1000 (88544GC) NIC, 3 us to copy a 64-byte packet by DMA</td>
<td></td>
</tr>
<tr>
<td>Linux processing: OS overheads</td>
<td>IXP processing: Optimized ISA</td>
</tr>
<tr>
<td>Processing a data packet in splicing state: 13.6 us</td>
<td>6.5 us</td>
</tr>
</tbody>
</table>
Throughput vs Request File Size

- Throughput is increased significantly
  - 5.7x for small file size @ 1KB, 2.2x for large file @ 1MB
- Higher improvement for small files
  - Latency reduction for control packets > data packets
  - Control packets take a larger portion for small files
Conclusion

- Network Processor combines high-performance packet processing and programmability
- A large variety of NP applications
- Efficient resource utilization is challenging
Thank you!