INF5061:
Multimedia data communication using network processors

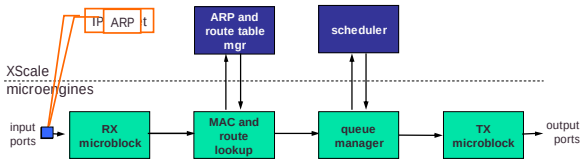
A First Example: The Bump in the Wire

9/9 - 2005

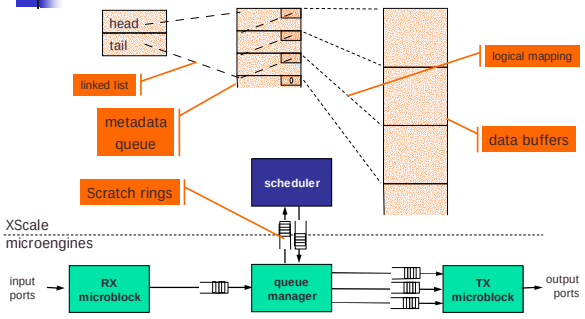
Using IXP2400

Programming Model

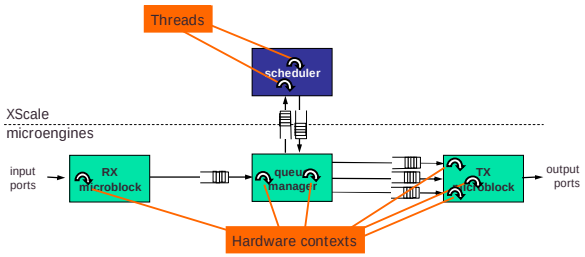
- Packet flow illustration for IP forwarding



Programming Model



Programming Model

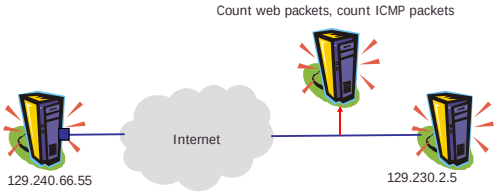


Framework

- uclo
  - Microengine loader
  - Necessary to load your microengine code into the microengines at runtime
- hal
  - Hardware abstraction layer
  - Mapping of physical memory into XScale processes' virtual address space
  - Functions starting with hal
- ossl
  - Operating system service layer
  - Limited abstraction from hardware specifics
  - Functions starting with ix_
- rm
  - Resource manager
  - Layered on top of uclo and ossl
  - Memory and resource management
    - all memory types and their features
    - IPC, counters, hash
  - Functions starting with ix_

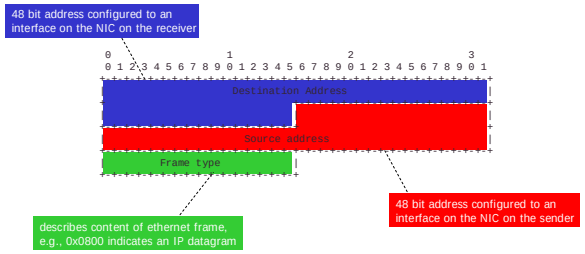
Bump in the Wire

Bump in the Wire

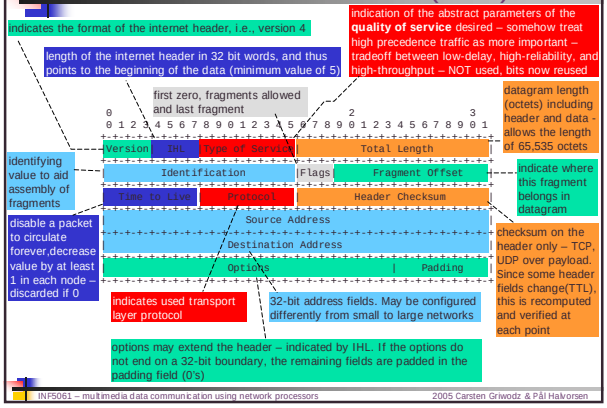


Packet Headers and Encapsulation

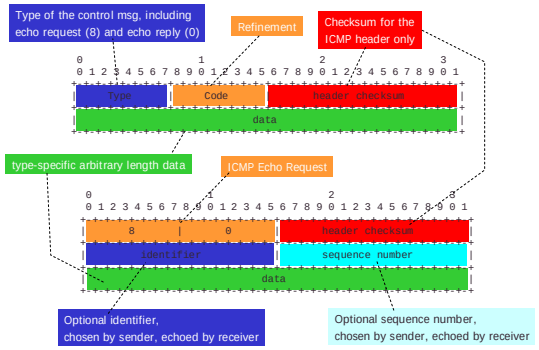
Ethernet



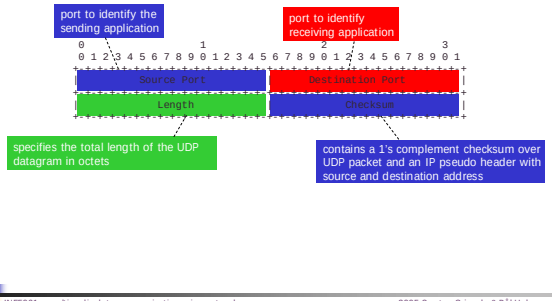
Internet Protocol version 4 (IPv4)



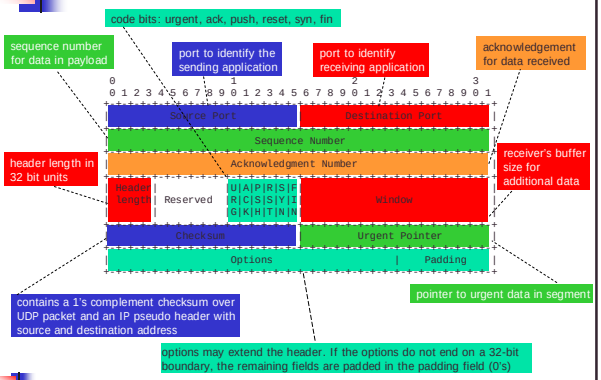
Internet Control Message Protocol (ICMPv4)



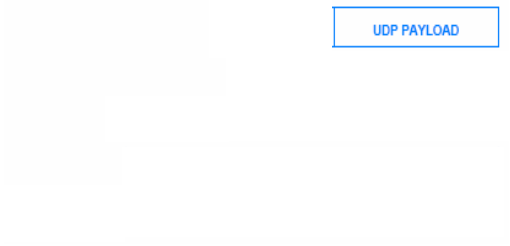
UDP



TCP

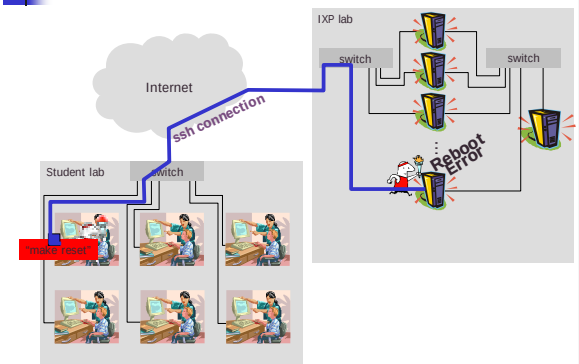


Encapsulation

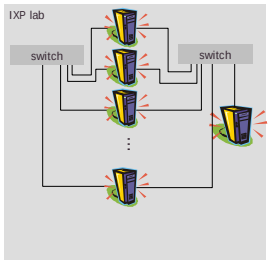


Lab Setup

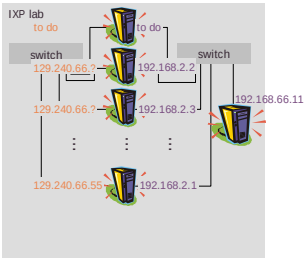
Lab Setup



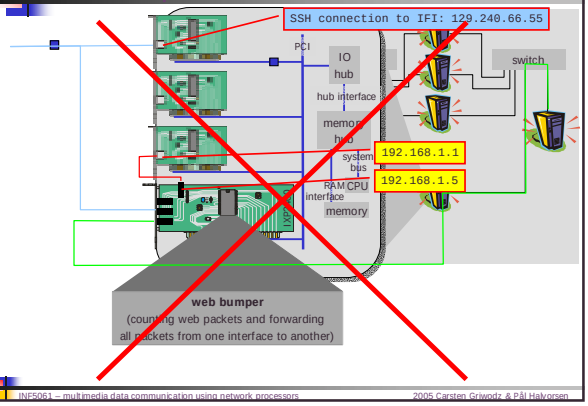
Lab Setup - Addresses



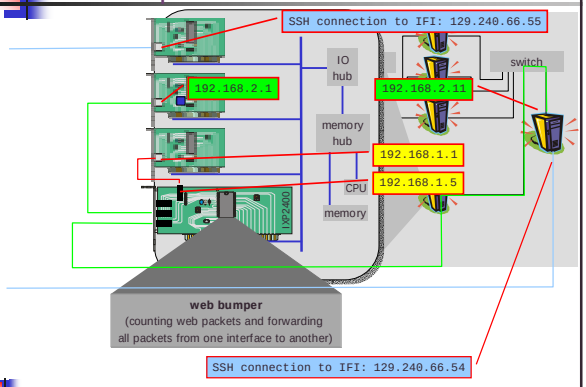
Lab Setup - Addresses



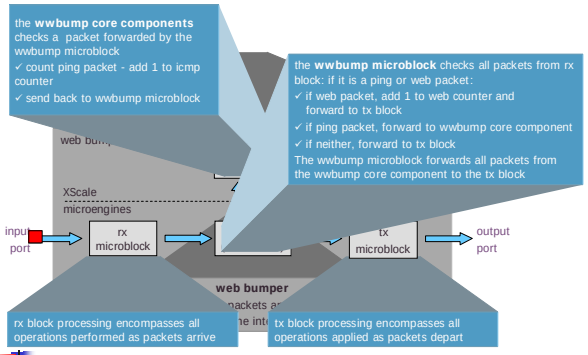
Lab Setup – Data Path



Lab Setup – Data Path



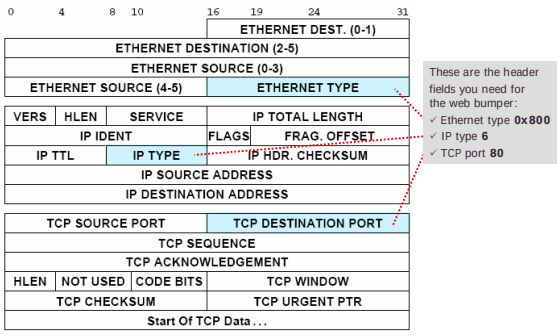
The Web Bumper



Starting and Stopping

- ✓ On the host machine
  - Location of the example: **`/root/ixa/wwpingbump`**
  - Rebooting the IXP card: **`make reset`**
  - Installing the example: **`make install`**
  - Telnet to the card: **`telnet 192.168.1.5`**
- ✓ On the card
  - To start the example: **`./wwbump`**
  - To stop the example: **`CTRL-C`**

Identifying Web Packets



Memory

Intel IXP2400 Hardware Reference Manual

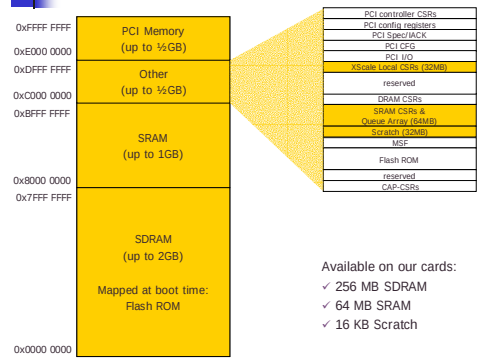
Intel® IXP2400 Network Processor

intel

This page intentionally left blank.

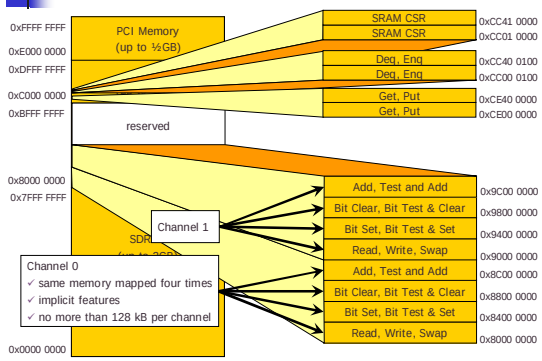
Hardware Reference Manual 24

XScale Memory Mapping

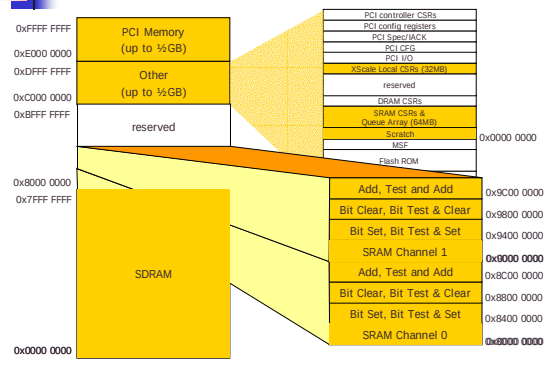


Available on our cards:
- ✓ 256 MB SDRAM
- ✓ 64 MB SRAM
- ✓ 16 KB Scratch

XScale Memory Mapping



Microengine Memory Mapping



XScale Memory

- ✓ A general purpose processor
  - With MMU
    - In use!
  - 32 Kbytes instruction cache
    - Round robin replacement
  - 32 Kbytes data cache
    - Round robin replacement
    - Write-back cache, cache replacement on read, not on write
  - 2 Kbytes mini-cache for data that is used once and then discarded
    - To reduce flushing of the main data cache
  - Instruction code stored in SDRAM

Microengine Memory

- 256 general purpose registers
  - Arranged in two banks
- 512 transfer registers
  - Transfer registers are not general purpose registers
  - DRAM transfer registers
    - Transfer in
    - Transfer out
  - SRAM transfer registers
    - Transfer in
    - Transfer out
  - Push and pull on transfer registers usually by external units
- 128 next neighbor registers
  - New in ME V2
  - Dedicated data path to neighboring ME
  - Also usable inside a ME
  - SDK use: message forwarding using rings
- 2560 bytes local memory
  - New in ME V2
  - RAM
  - Quad-aligned
  - Shared by all contexts
  - SDK use: register spill in code generated from MicroC

SDRAM

- Recommended use
  - XScale instruction code
  - Large data structures
  - Packets during processing
- 64-bit addressed (8 byte aligned, quadword aligned)
- Up to 2GB
  - Our cards have 256 MB
  - Unused higher addresses map onto lower addresses!
- 2.4 Gbps peak bandwidth
  - Higher bandwidth than SRAM
  - Higher latency than SRAM
- Access
  - Instruction from external devices are queued and scheduled
  - Accessed by
    - XScale
    - Microengines
    - PCI

SRAM

- Recommended use
  - Lookup tables
  - Free buffer lists
  - Data buffer queue lists
- 32-bit addressed (4 byte aligned, word aligned)
- Up to 16 MB
  - Distributed over 4 channels
  - Our cards have 8 MB, use 2 channels
- 1.6 Gbps peak bandwidth
  - Lower bandwidth than SDRAM
  - Lower latency than SDRAM
- Access
  - XScale
  - Microengines
- Accessing SRAM
  - XScale access
    - Byte, word and longword access
  - Microengine access
    - Bit and longword access only

SRAM Special Features

- Atomic bit set and clear with/without test
- Atomic increment/decrement
- Atomic add and swap
- Atomic enqueue, enqueue_tail, dequeue
  - Hardware support for maintaining queues
  - Combination enqueue/enqueue_tail allows merging of queues
  - Several modes
    - Queue mode: data structures at discontiguous addresses
    - Ring mode: data structures in a fixed-size array
    - Journaling mode: keep previous values in a fixed-size array

SRAM Special Features

Name	Longword #	Bit #[a]	Definition
EOP	0	31	End of Packet—decrement Q_count on dequeue
SOP	0	30	Start of Packet
Segment Count	0	29:24	Number of segments in the buffer
Head	0	23:0	Head pointer
Tail	1	23:0	Tail pointer
Q_count	2	23:0	Number of packets on the queue or number of buffers on the queue
SW_Private	2	31:24	Ignored by hardware, returned to ME
Head Valid	N/A		Cached head pointer valid—maintained by hardware
Tail Valid	N/A		Cached tail pointer valid—maintained by hardware

Name	Longword #	Bit #	Definition
Ring Size	0	31:29	See Table 129 for size encoding.
Head	0	23:0	*Get* pointer
Tail	1	23:0	*Put* pointer
Ring Count	2	23:0	Number of longwords on the ring

Scratch Memory

- Recommended use
  - Passing messages between processors and between threads
  - Semaphores, mailboxes, other IPC
- 32-bit addressed (4 byte aligned, word aligned)
- 4 Kbytes
- Has an atomic autoincrement instruction
  - Only usable by microengines

Scratchpad Special Features

- ✓ Atomic bit set and clear with/without test
- ✓ Atomic increment/decrement
- ✓ Atomic add and swap
- ✓ Atomic get/put for rings
  - Hardware support for rings links SRAM
  - Signaling when ring is full