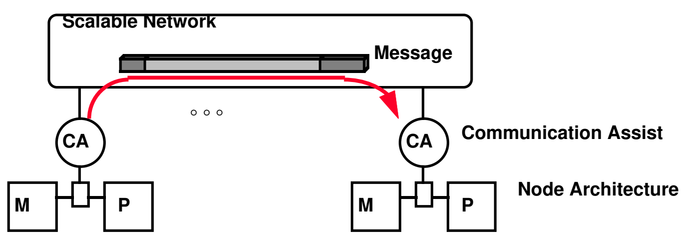
Design Space of Large-Scale Distributed Memory Machines: Part I

- Recap
- Key Issues
- Physical DMA
- User-level Ports
- User-level Handlers

Key Issue: degree of message interpretation



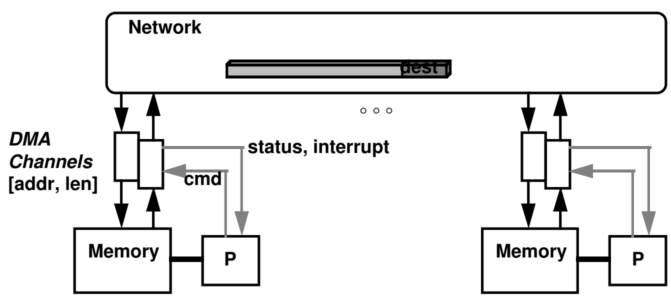
- How much interpretation of the message?
  - **format**
  - **buffering**
  - **action**
- How much dedicated processing in the Comm. Assist?

Spectrum of Designs

- None: Physical bit stream
  - **blind, physical DMA** — **nCUBE, iPSC, . . .**
- User/System
  - **User-level port** — **CM-5, *T**
  - **User-level handler** — **J-Machine, Monsoon, . . .**
- Remote virtual address
  - **Processing, translation** — **Paragon, Meiko CS-2**
- Global physical address
  - **Proc + Memory controller** — **RP3, BBN, T3D**
- Cache-to-cache
  - **Cache controller** — **Dash, KSR, Flash**

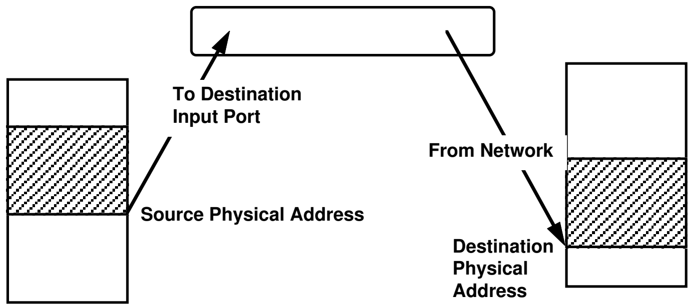
Increasing HW Support, Specialization, Intrusiveness, Performance (???)

Physical DMA



- Processor can cause a transfer into the network to a particular destination
- Processor can accept transfer from network into memory
  - **cannot look at the data to decide where it should go**

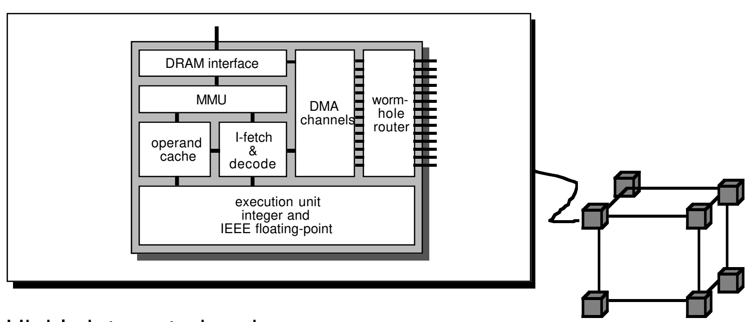
Implications of Physical DMA



- Transmit at system level
  - **virtual to physical translation**
  - **arbitration for media and destination**
- Undirected input and interrupt to system level

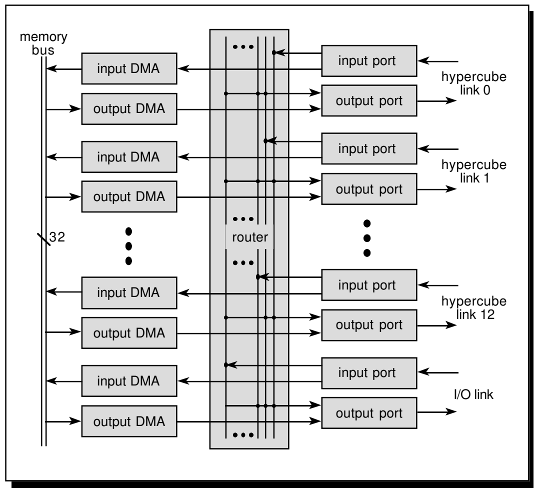
(Can play games if channel is memory mapped and network is dedicated)

Example: nCUBE/2

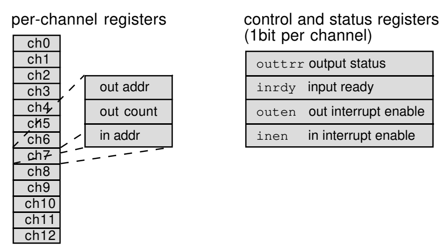


- Highly integrated node
  - **20 MHz, 64-bit int and FP, "reduced VAX", 5 MIPS**
  - **direct 32-bit + ECC DMA interface**
  - **28 DMA channels**
- Up to 13-D (8096-node) hypercube + 1 I/O per node
- Primitive MMU and Cache
- Priviledged channel operations (add, len, status, mask)

nCUBE Network Interface Organization

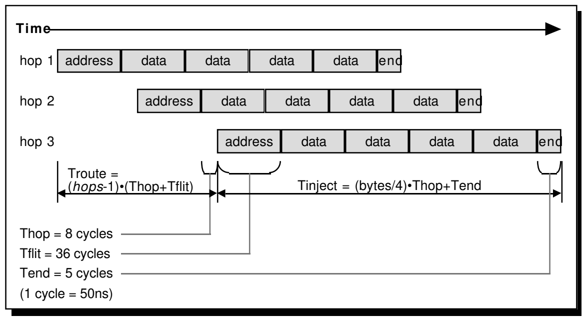


nCUBE/2 Channel Operations



- Protection: only priviledge level channel operations
- No programmed I/O
- First Level or routing is port selection
- Format: destination node + { 32-bit word }*
- Status bit vectors
- Interupt enable and vector per channel

nCUBE/2 Network



- 32-bit + 4 parity flit
- e-cube routing (correct bits in order)
- Reliable (blocking)
- Deadlock free if always receive
- sub-cube broadcast
- message “train” option

SEND:

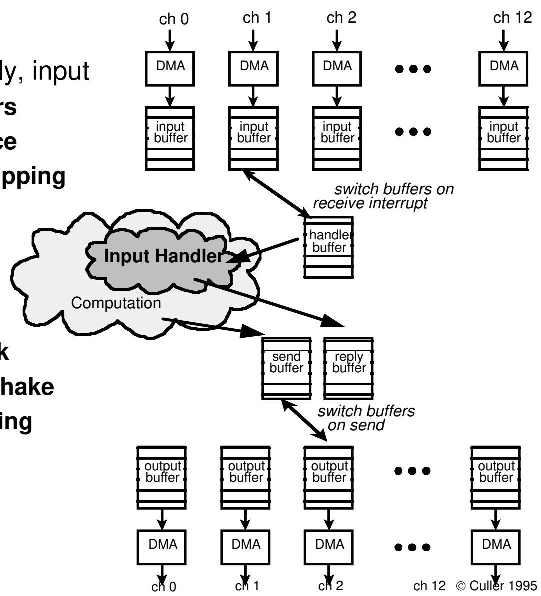
```
while(!ready(sendDMA))
    while(ready(recvDMA))
        receive();
send();
```


nCUBE/2 Operation Summary

- Messages
  - **Arbitrary number of 32-bit words, first word is dest node address**
  - **All messages originate in a memory buffer on the source node**
  - **Are injected into the network by a DMA engine**
  - **Are routed through the network to the destination node**
  - **Are transferred into a memory buffer by another DMA engine**
- DMA access reserved to the operating system
  - **operates in physical address space (ok simple segmented memory)**
- Send:
  - **set-up output DMA engine with the message start addr and length**
  - **when done DMA sets a status bit and optionally generates an interrupt.**
- Receive:
  - **input DMA engine must be blindly set-up with base address**
  - **no message length limit is enforced by the hardware**
  - **when done the DMA sets status bit and optionally generates interrupt**
- Messages can be subdivided into segments
  - **allows separate envelop and body**
  - **each segment is a separate DMA operation**

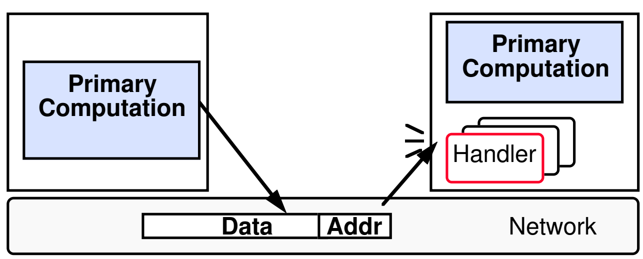
Buffer Management

- Buffer per channel +
  Process request, reply, input
  - pre-allocate buffers
  - xlate and map once
  - alloc/dealloc by flipping pointers
- Send stall policy ?
- Interrupt on input
  - system/user check
  - kernel/user handshake
  - mapping vs. copying
  - security holes



Network Transactions

- System Level
  - Kernel supported message passing
  - Remote Virtual Memory
    - » get / put regions
- User Level



•Sender injects the message directly into the network

•Associates a small user-level handler with each message

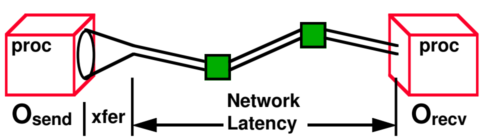
•Handler executes immediately upon arrival

–pulls the message out of the network and integrates it into the ongoing computation, or replies (as determined by higher level programming model)

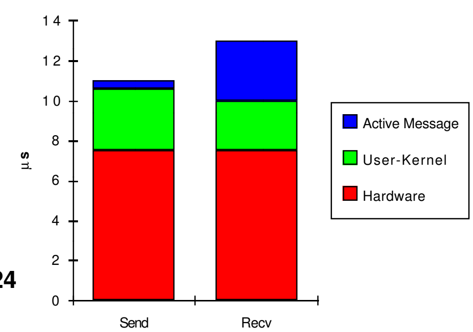
nCUBE/2 AM Timing (von Eicken 93)

	Inst Category	Count		Memory Accesses	
AM Send	Kernel Trap	2		6	
	Check Dest. and Length	4		2	
	Route to port	1		4	
	Check Port Ready	3		0	
	Set up DMA	2		0	
	Buffer Mgmt	4		6	
	Total	*16*		*18*	
		Kernel	*User*	*Kernel*	*User*
AM Recv	Interrupt	2		7	
	Check System/User	1		1	
	Buffer Mgmt	4		6	
	restart DMA	2		1	
	user-level int + handshake	3+3	2	5+4	2
	dispatch to handler		3		3
	return to computation		7		6
	save registers	2	6	2	6
	Total	*17*	*18*	*26*	*17*

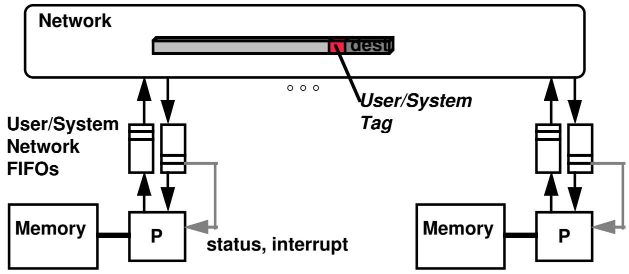
nCUBE/2 Timing Summary



- Assuming data is in output buffer
- O_{send} = 13.1 µs
- Xfer = 0.45 µs per byte
- Network latency = 2.3µs per hop
  – **average distance 5 for 1,024**
- O_{rcv} = 15.0 µs



User Level Port



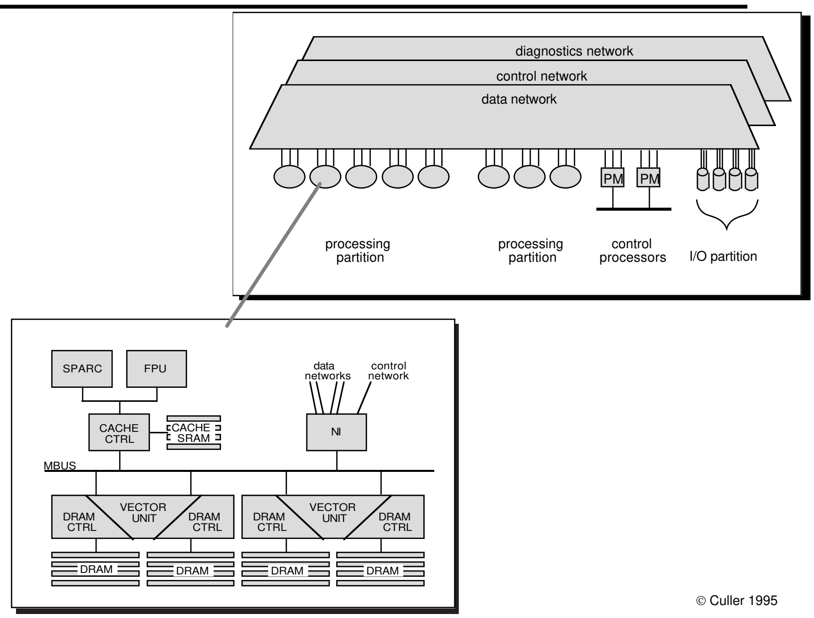
- Processor can initiate transaction at user level by writing the destination and information into the network
- Processor can read messages from the network
- Network contents is part of the "user process state"

=> Comm. assist checks destination, arbitrates for media, inserts message type

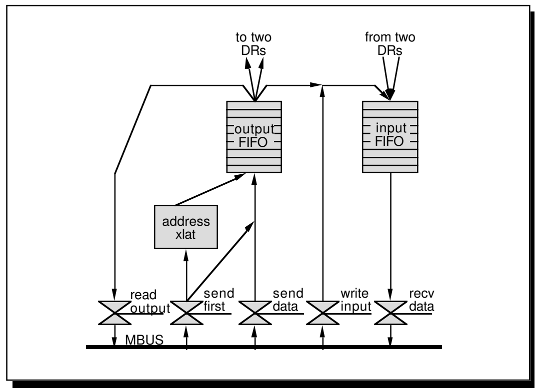
=> Message either waits in input queue or causes interrupt

   – **system must get its messages**

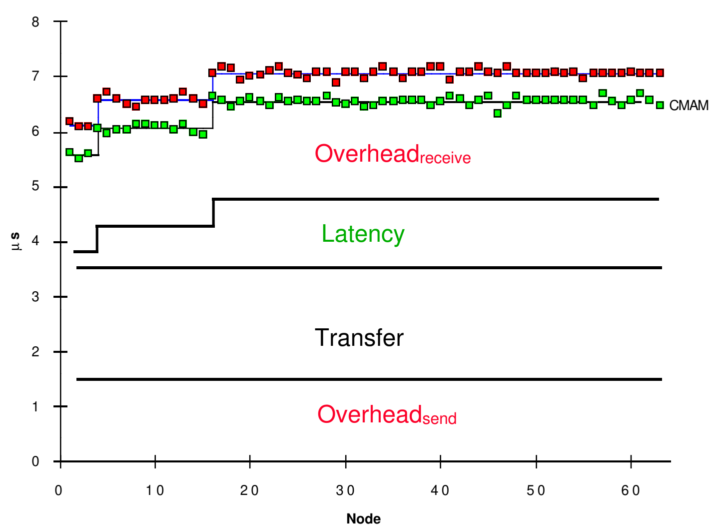
Example: CM-5



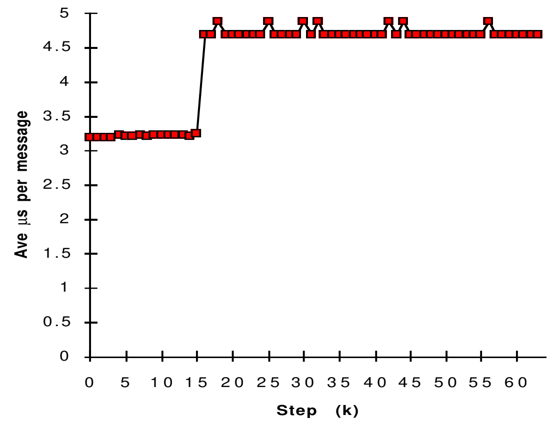
CM-5 Network Interface



CM-5 One-way Message Timing

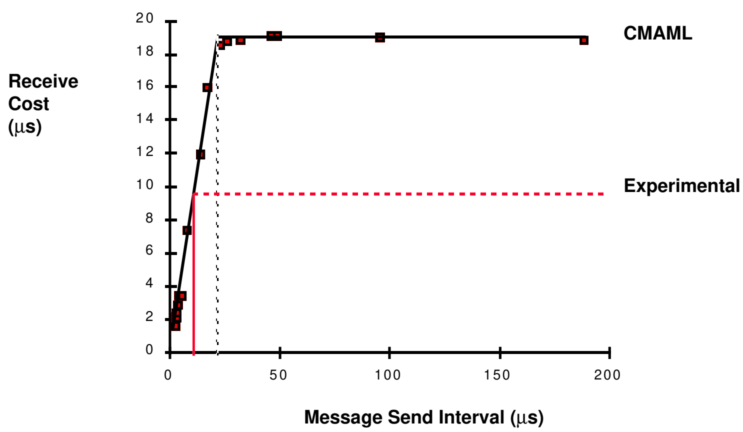


CMAM Performance under load



In step *k*, each processor *p* send several messages to processor *p* xor *k*.

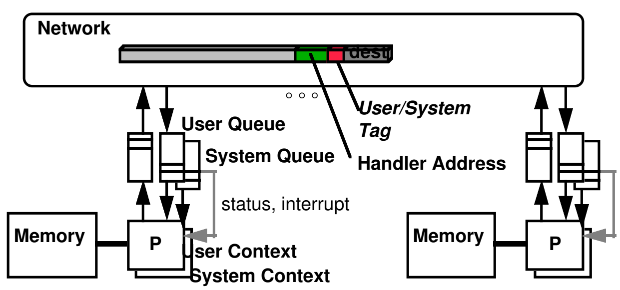
Cost of the interrupt



Deadlock Avoidance and Virtual Networks

- Check output buffer status on send
  - **must drain receive while waiting**
- Distinct request / response network
  - **drain both on request stall**
  - **drain responses on reply stall**
  - **or build a queue in memory and manage it**
- System must be able to get its messages out from behind user
  - **two queues**
  - **Drain queue and stuff them back in**
- System must be able to send
  - **User may have partially formed message**
  - **Pull it back out are put it back or make it fail**
- Drain user network partition on time-slice
  - **Restore on resume**
  - **System pushes msgs in, but user needs to pull them out**
    - » **may take several tries to restart**

User-level Handler



- Examples: J-Machine, Monsoon
- Processor dispatches directly to address of highest priority message
  - **interaction of thread state and handlers**
  - **handler cannot suspend; it queues its continuation**
  - **input queue overflow**