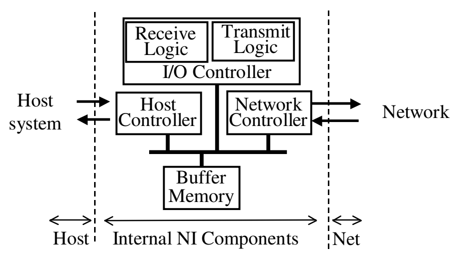
Programmable Network Interfaces

Marc E. Fiuczynski

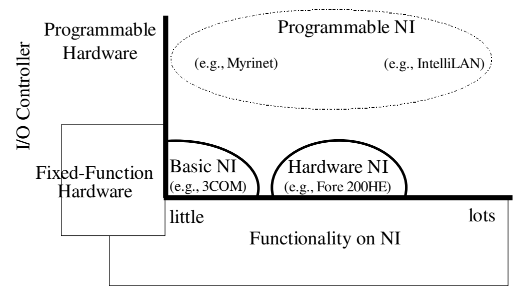
Outline

- Programmable NI Background
- Pros/Cons of Programmable NIs
  - Performance overview of Prog. NIs
- Compelling & Emerging Applications for Prog. NIs
- Future Directions
- Conclusion

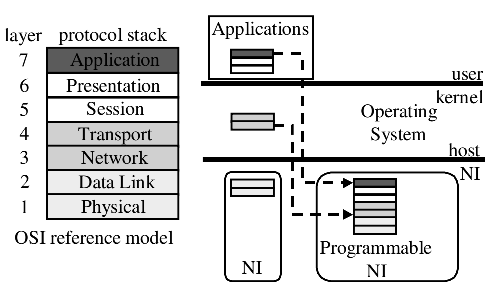
NI Architecture



NI Design Space



Offloading Functionality to NIs



Example Applications/Services offloaded to programmable NIs

- telnet/rlogin + TCP/IP
  - CMC & Excelan NIs (circa 1985-1988)
- Active Message, Fast messages, VMMC, etc.
  - Myrinet NI for NOWs (circa 1995-1998)
- NFS services
  - Auspex NI (circa 1989)

Pros of Programmable NIs

- Enables new applications/protocols
- Improves performance
  - eliminates software overhead
  - reduces CPU utilization on host system
  - increases scalability
- Improves development cycle & product lifetime
  - software development faster than ASIC development
  - bug-fix and feature upgrade after shipment

Cons of Programmable NIs

- NI CPUs are too slow
  - H&P book states:
    *“**Fallacy:** Adding a processor to the network interface card improves performance.”*
- NI CPUs do not improve at same rate as host CPUs
  - H&P book states:
    *“**Pitfall:** Moving functions from the host CPU to the NI CPU to improve performance.”*
- Inadequate memory subsystem (Kanakia & Cheriton ‘88)
  - insufficient bandwidth for all concurrent hardware components
- “Too expensive; will never be commodity”

Evaluating NI Processors

- Dhrystone benchmark (DMIPS)
  - short synthetic benchmark program intended to be representative for system programming
  - available performance data for processors ranging back to mid 1980s & embedded processors vendors still use it
- Dhrystone Instructions Per Word (DIPW)

$$\boxed{\text{DIPW} = \frac{\text{Dhrystone MIPS}}{\text{max. bandwidth through NI (words/second)}}}$$

  - conservative ratio between computing and communication capacity of a programmable NI
    - e.g., 40 DMIPS CPU goes further with 10Mbps Ethernet than with 100Mbps Ethernet.

NI Applications by Complexity

	functionality offloaded	system	stack layer
high (~50)	TCP/IP + telnet/rlogin	CMC, Excelan	layer 7
	NFS services	Auspex NI	layer 7
	TCP/IP, IPSec	IntelliLAN, 3Com R990	layer 3-4
med. (2-5)	VMTP, RMON	VMP-NAB, Nectar CAB	
	TCP/UDP csum	Alteon Tigon2	
	packet filtering	lazy receive processing	
low (~1)	Active Messages, VCM, …	Myrinet & NOWs	layer 2-4
	ATM adaption layer, UNET	Fore200, Osiris	layer 2

Programmable NIs Processor Performance

NIC	Year	I/O Processor	Link type	BW	DMIPS	DIPW	Functionality Offloaded
Meiko CS2 NIC	1994	Elan	SAN	320	1.50	0.2	Active Messages
CMC ENP	1987	MC68010	Ether	10	0.71	2.2	telnet/rlogin, TCP/IP
Excelan EXOS	1988	Intel 186	Ether	10	1.13	3.6	telnet/rlogin, TCP/IP
VMP-NAB	1988	MC68020	Point2Point	100	2.30	0.7	VMTP
Myrinet	1995	LANai	SAN	640	20.00	1.0	AM, GMS, SPINE
Osiris	1991	Intel 960-CA	ATM	622	23.90	1.5	AAL5 + App. Dev. Ch.
Alteon Tigon2	1997	MIPS R3000	Ether	1000	75.00	2.4	TCP/UDP checksum
Fore 200E	1993	Intel 960-JA	ATM	155	28.00	4.5	AAL5, UNET, VCM
StarT-JR	1996	Intel 960-CF	IEEE 1394	100	47.80	15.3	User-level networking
3Com-R990	1999	ARM9 TDMI	Ether	100	220.00	70.0	TCP Seg.,IPSec
IntelliLAN	1999	Intel StrongARM	Ether	100	268.00	85.7	TCP/IP

Programmable NIs Memory Issues

NIC	Year	cache (KB)	scratch (KB)	Memory Size	Memory Type
Meiko CS2 NIC	1994	0	-	0	none
CMC ENP	1987	0	-	512KB	1port DRAM
Excelan EXOS	1988	0	-	512KB	1port DRAM
Myrinet	1995	0	-	1MB	2port SRAM
VMP-NAB	1988	0.25I 0D	-	128KB	2port VRAM
Osiris	1991	1I 0D	2	128KB	SRAM
Fore 200E	1993	2I 0D	1	256KB	SRAM
StarT-JR	1996	4I 1D	1	2MB	DRAM
Alteon Tigon2	1997	-	24	1MB	SRAM
3Com-R990	1999	16I 16D	-	2MB	SRAM
IntelliLAN	1999	16I 16D	-	16MB	SDRAM

Present “State of the Art” of Programmable NI Designs

- Cost/performance improvements in processor & memory technology enable high-performance programmable NIs
  - E.g., 3COM-R990 costs $120 retail
    - 25% premium over 3COM-905 “basic NI”
  - Fast embedded processors
    - 200+ Mhz, 200+ Dhrystone MIPS
    - sufficient processing capacity for 100Mbps Ethernet
  - high capacity, high bandwidth memory
    - SDRAM, EDORAM

10/14/99 13

Applications for Programmable NIs (*middle* of the network)

- Web Switching
  - load balance web requests, firewall, NAT
    - e.g., Alteon Websystems, F5Labs BIG/IP, ...

  challenge: *scaling to gigabit and faster network connections*
- Active Networking
  - deploy innovative network services/features rapidly
    - e.g., active reliable multicast, active congestion control, ...

  challenge: *introduce user-provided services into network, and execute them securely and efficiently on a per packet basis*

10/14/99 14

Applications for Programmable NIs (*edge* of the network)

- Server-specific NI Software
  - increase manageability, reliability, scalability, efficiency
    - e.g., Auspex, Intel SmartEther, Lazy Receive Processing

  challenge: *vendor determines what functions to offload to the NI that benefit many applications*
- Application-specific NI software
  - customize NI to system’s current application workload
    - e.g., GMS, Video Client, object specific HTTP get/put ops

  challenge: *application-specific message processing both efficiently and safely*

10/14/99 15

System Software for Programmable NIs

- Controller executives (e.g., Active Message, firmware NIs)
  - single threaded, polling loop
  - tightly integrated with device
- Embedded OS (e.g., IxWorks for I_2O, MMlite, Nectar)
  - threads & real-time scheduling
- “Slim” Unix (e.g., PicoBSD, ELKS, eVINO)
  - user processes

10/14/99 16

Future Directions

- Next generation NI hardware architecture
  - support 1Gbps Ethernet and faster
- System software that enable applications to customize the NI

10/14/99 17

Next Generation NI Hardware Architecture

Goal: Process message data @ wire rate

- Multiprocessor *(e.g., SMT, cMP)*
  - each message is an independent unit of work
  - 350MHz 4-processor NI *may have* sufficient DIPW to support TCP/IP @ 1Gbps Ethernet
- Vector or SIMD functional units *(e.g., v-IRAM, MMX)*
  - perform the same operation across message data
- Memory hierarchy for streaming message data
  - avoid D-cache pollution with message data

10/14/99 18

System Software that Enables Applications to Customize the NI

Goal: Application-specific message processing @ wire rate

- How are applications partitioned?
- How is application code loaded?
- How does one protect against bugs?
- How are resources managed?
- How are principals specified?
- How are they scheduled?

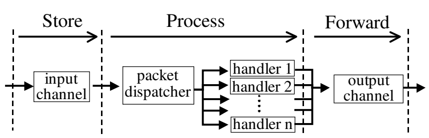
SPINE

Conclusion

- Programmable NIs are enabled by cost/performance improvements in embedded processor & memory technology
- Programmable NIs are being used to meet scalability, reliability, manageability, and efficiency requirements
- More work is needed in both software & hardware architecture to address challenges of emerging applications

"General-Purpose" Message-driven Scheduling

Goal: Efficiently and fairly schedule message movement and associated processing



- Optimal Store-and-Forward Pipeline Communication (Wang et al.)
- *Scout Paths*: communication oriented scheduling (Mosberger)
- Language-supported ephemeral message processing (Fiuczynski et al.)

Dhrystone Performance of Workstation Processors

Machine	Year	Host Processor	DMIPS	host/pNI
Sun 3x	1989	MC68030 (20Mhz)	7	3.0
DECstation 5000	1991	MIPS R3000 (33Mhz)	29	1.2
DECstation 3000/400	1993	Alpha 21064 (133Mhz)	149	5.4
Intel PC	1996	Intel PPro (200Mhz)	213	4.2
Intel PC	1998	Intel PII (450Mhz)	400	2.1
Alpha Personal Workstation	1998	Alpha 21164 (500Mhz)	517	1.9
Alpha NG Workstation	2000	Alpha 21264 (1Ghz)	~ 1040	2.0