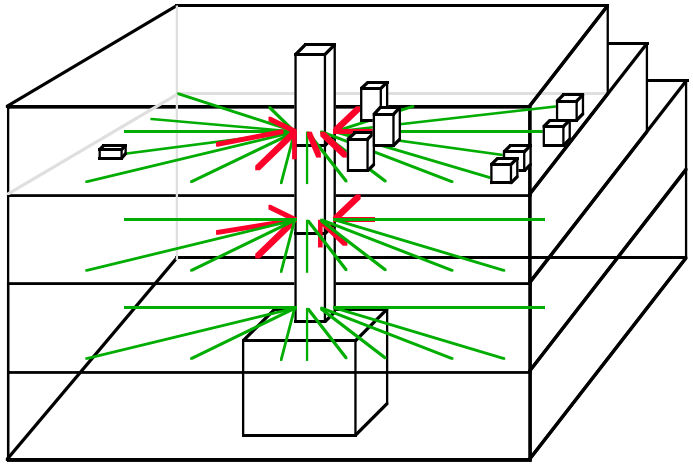
Berkeley NOW:
System Design in the era of the "killer" network

David Culler,

Tom Anderson,

Dave Patterson,

et al.

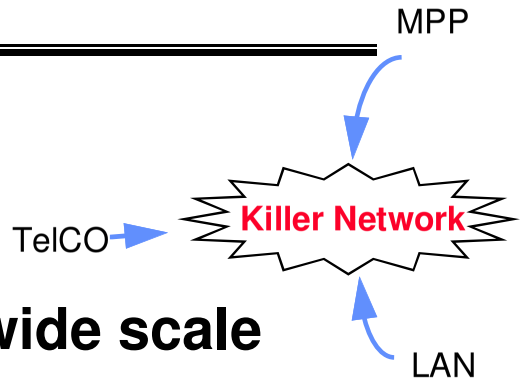


Computer Science Division

University of California, Berkeley

http://now.cs.berkeley.edu/

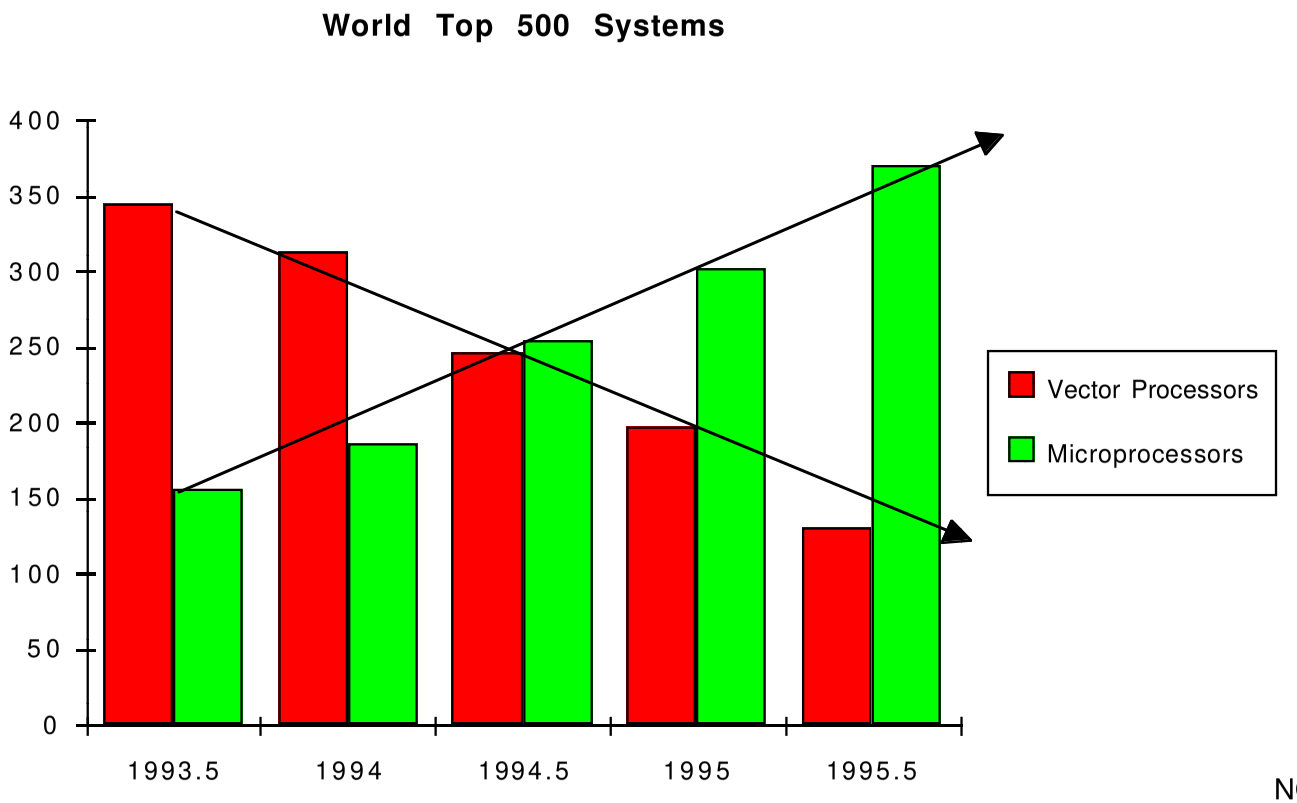
Why NOW?



- New "Killer Network" technology
  - make it "real"
  - push its effect throughout the system
- Integrated system on a building-wide scale
  - pool of resources
    - » processor, disks, memory
  - remote processor and memory closer than local disk
  - federation of systems with local and global role
- New methodology for building large systems
  - just in time commodity components
  - snap together in simple, scalable fashion
  - available system out of unreliable parts
  - auto-configuring, self-tuning
    - » anyone can do anything, so who does what
- The right way to build internet services

Remember

the “attack of the killer micro”



Rethinking Large System Design

Myth: Better integration yields efficiency and performance

=> Engineering time costs performance

- 1 year lag costs 50% in performance
- 2 year lag costs a factor of two

- Complete desktop system leads the tech. curve
  - SMP lags 1-2 years
  - MPP lags 2-3 year

Rethinking (cont)

Myth: Integration yields economies of scale

Volume yields economy of scale

- costs drop by 10% as volume doubles (G. Bell)
- small systems are cheap
- cheap system have large volume

=> 1,000x in volume reduces costs by 3x

- plus engineering costs are amortized over more units

Myth: Specialization yields cost reduction

- leave out any component of the desktop system and it costs you 2-5x more

Networks of Workstations

The Basis for building large-scale systems is:

- **Complete mass-produced platforms**
  - **"killer" micro, memory system, disk, O/S,**

    plus SOFTWARE

    => workstations (or PCs)

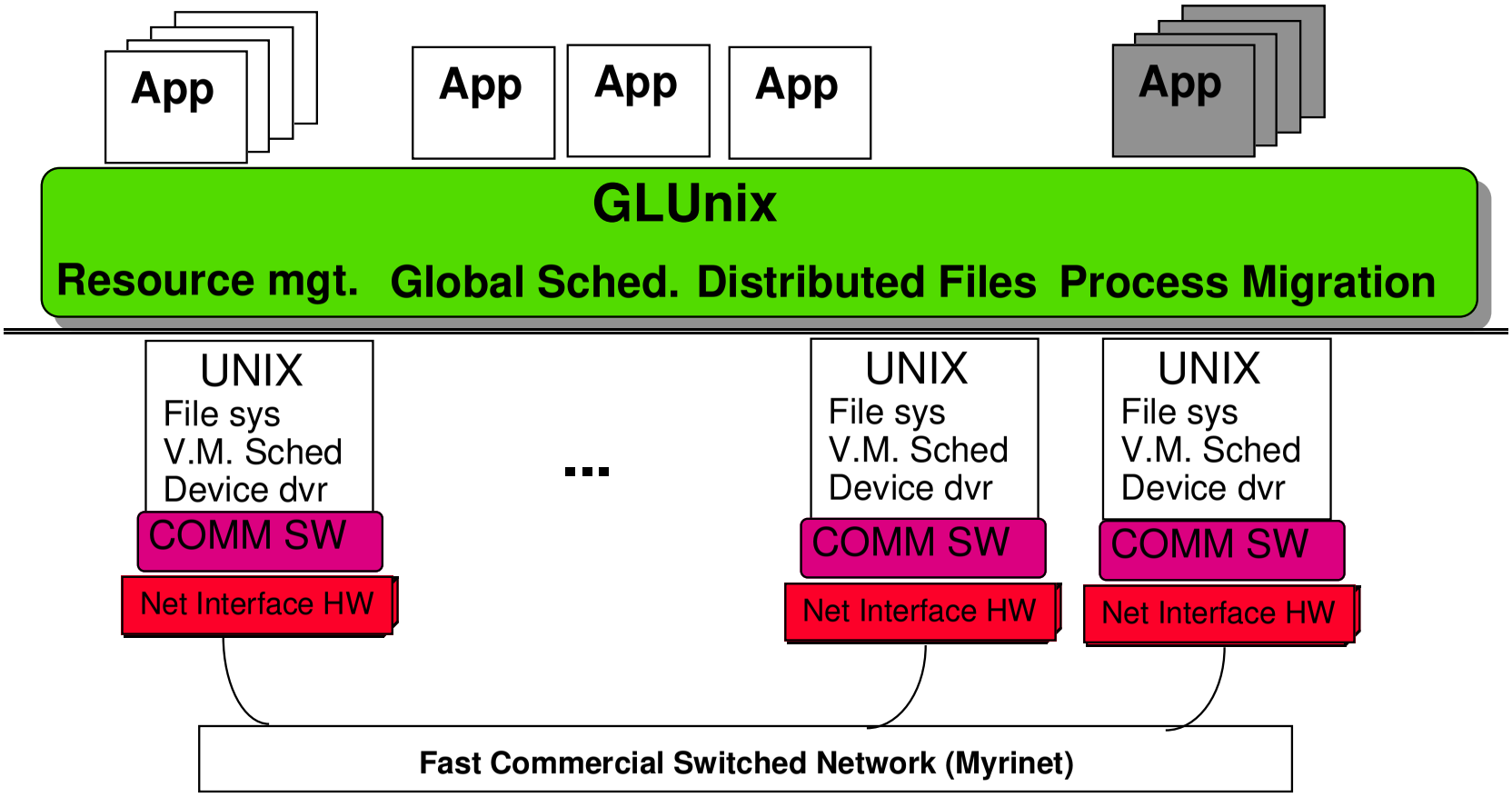
integrated via

- **Cheap, Fast, Scalable (switched) Networks**
  - **high bandwidth, low latency, low error rate**
  - **ATM, repackaged MPP, or whoever wins**
- **and they need to be self-configuring**

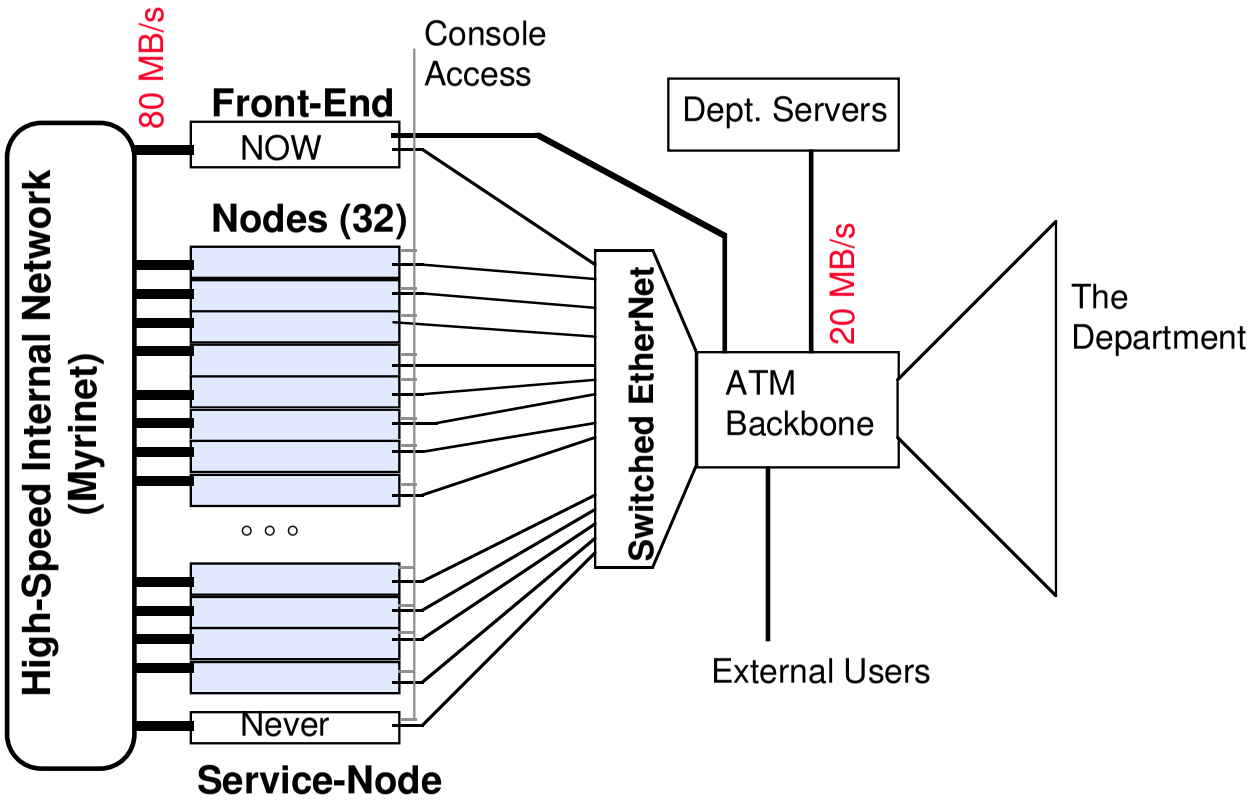
Outline

- **UCB NOW System**
- **Fast Communication in a NOW**
- **Implicit Coordinated Scheduling**
- **Scalable Servers**

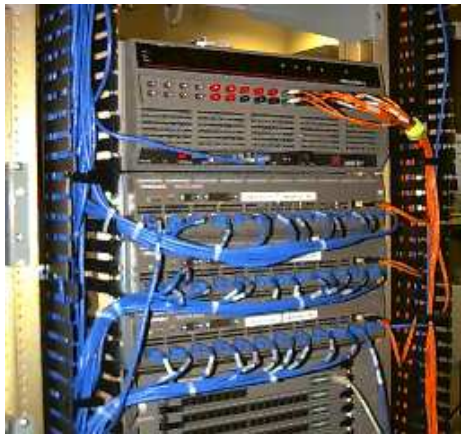
UCB NOW System Architecture



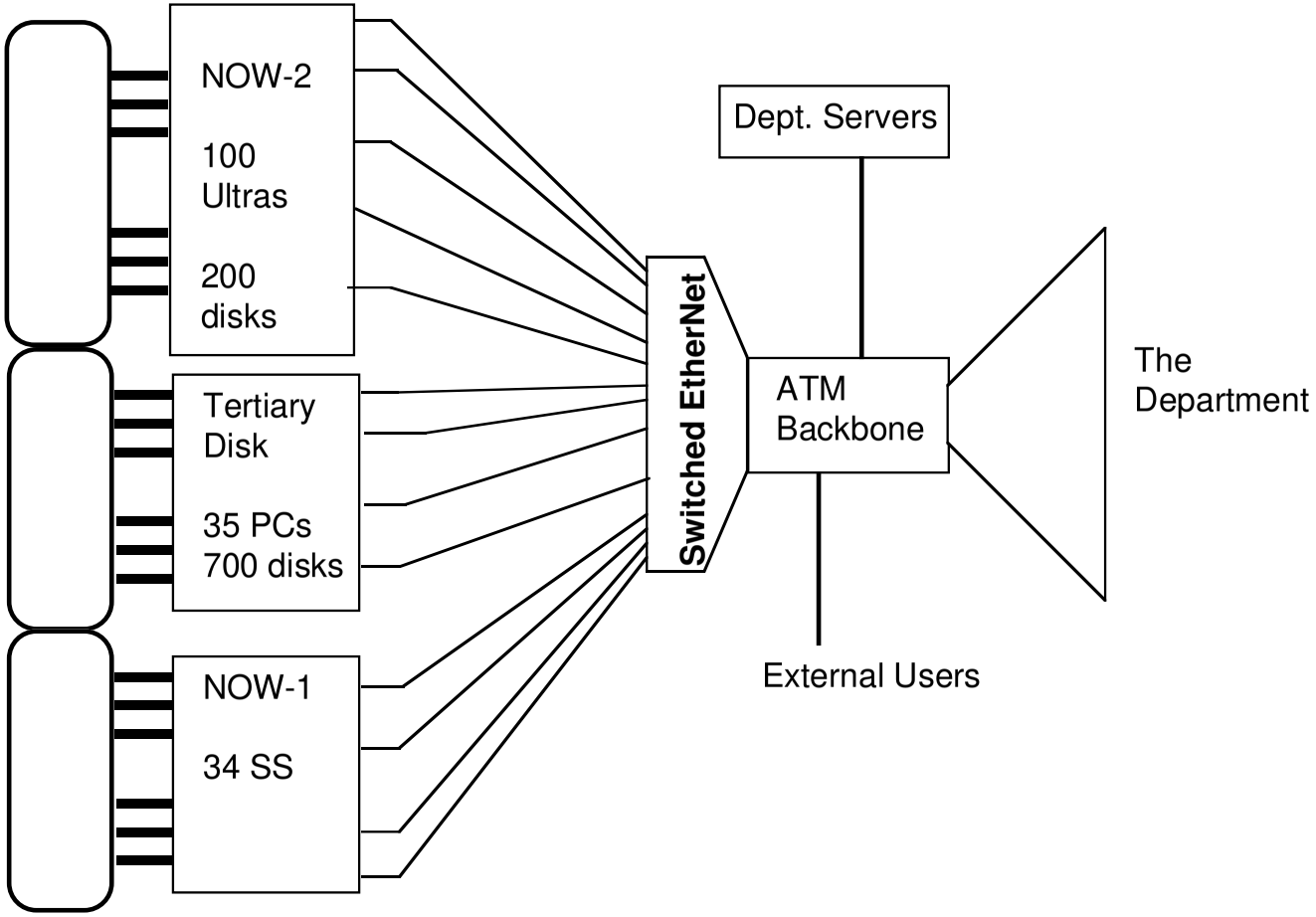
Logical Organization: NOW-1



Physical System: NOW-1



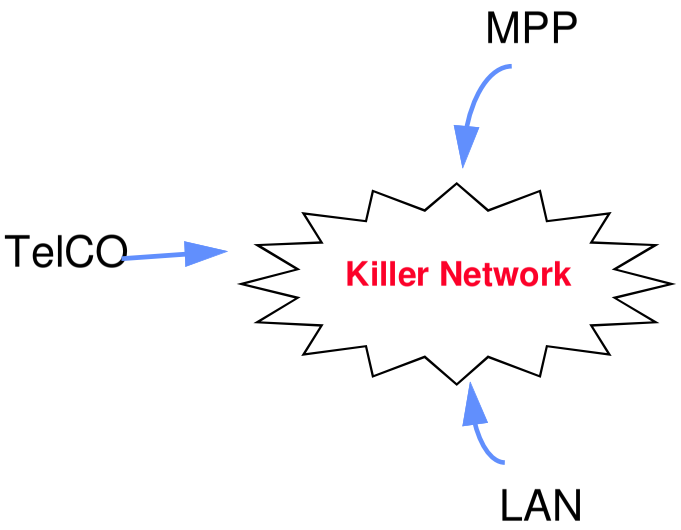
Logical Organization: NOW + T. Disk



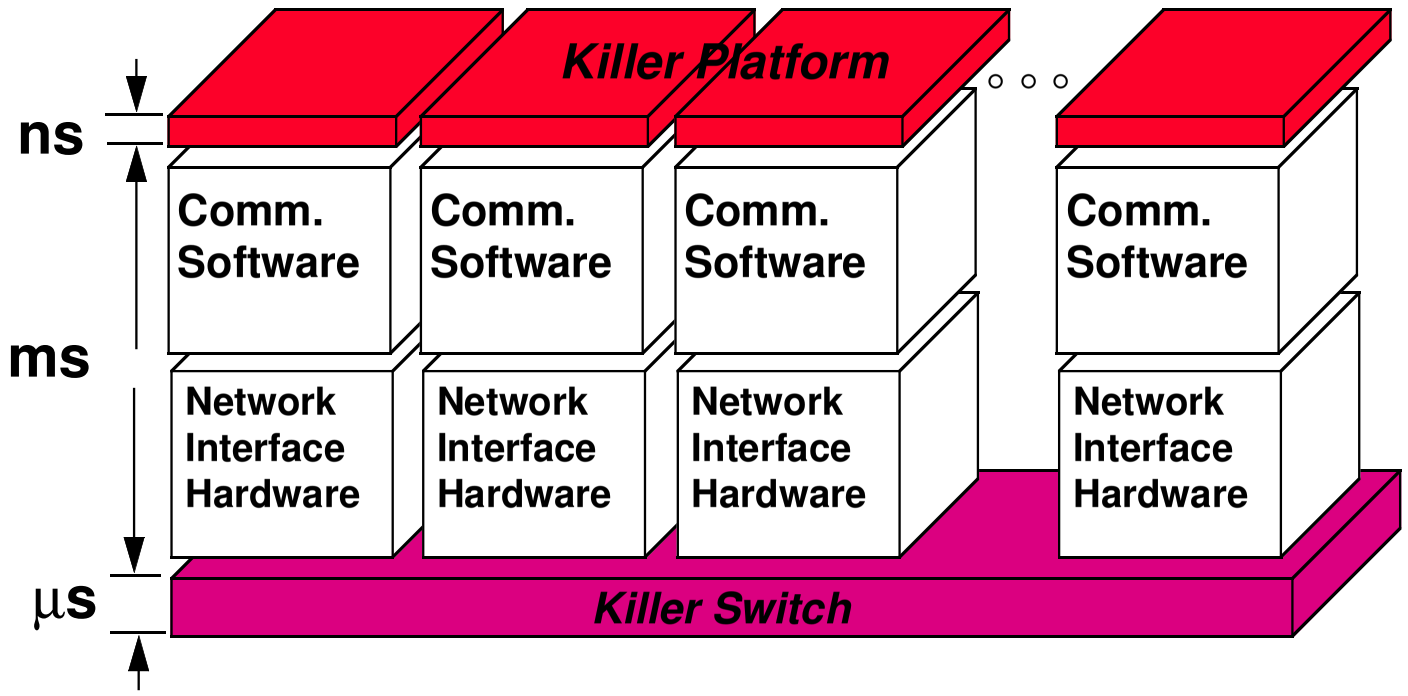
Physical System: NOW-2

Tertiary Disk “Wall”

Fast Communication in a NOW



Where does the time go?



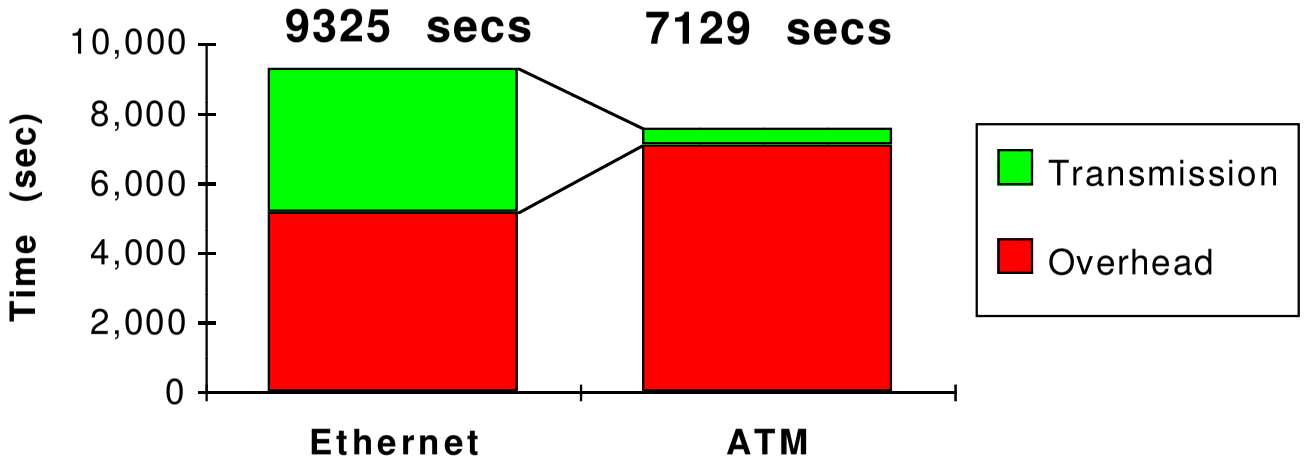
- **NOW Goal: 10 µs end-to-end**

Overhead and Latency

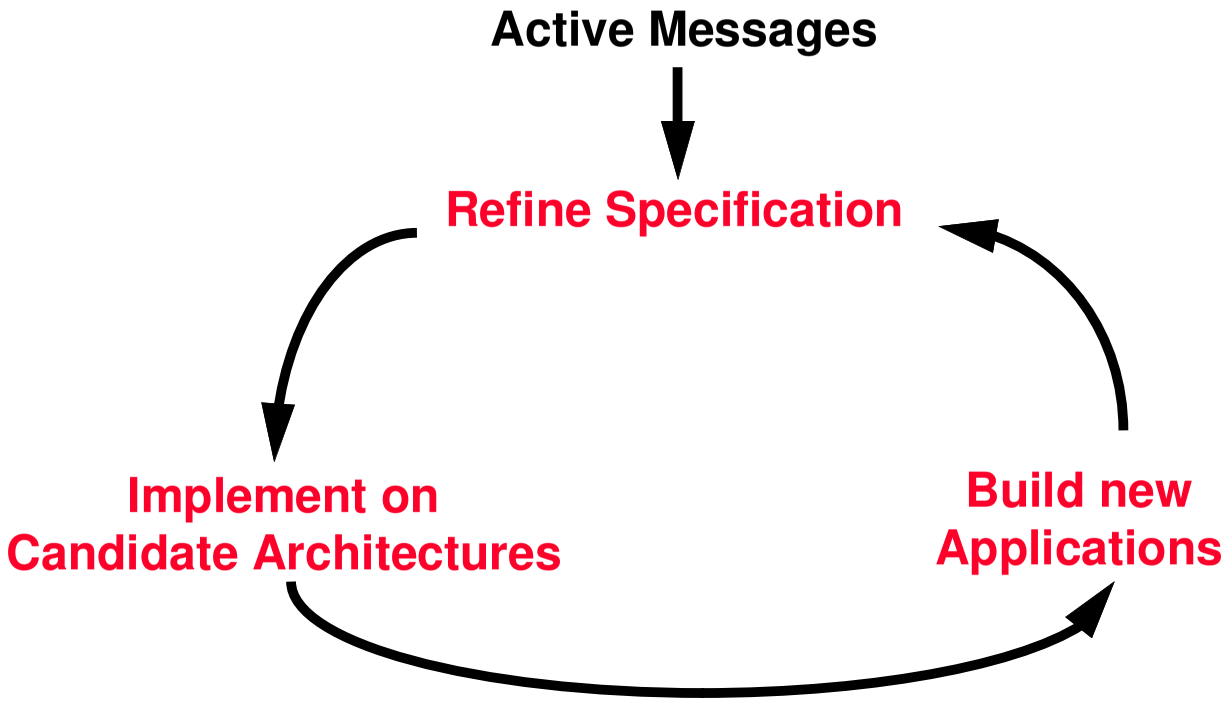
UDP on

- **Ethernet / SS10: 9 Mb/s BW, 900 μsecs ovhd**
- **ATM Synoptics: 78 Mb/s BW, 1,250 μsecs ovhd**

- **NFS trace over 1 week: 95% msgs < 200 bytes**



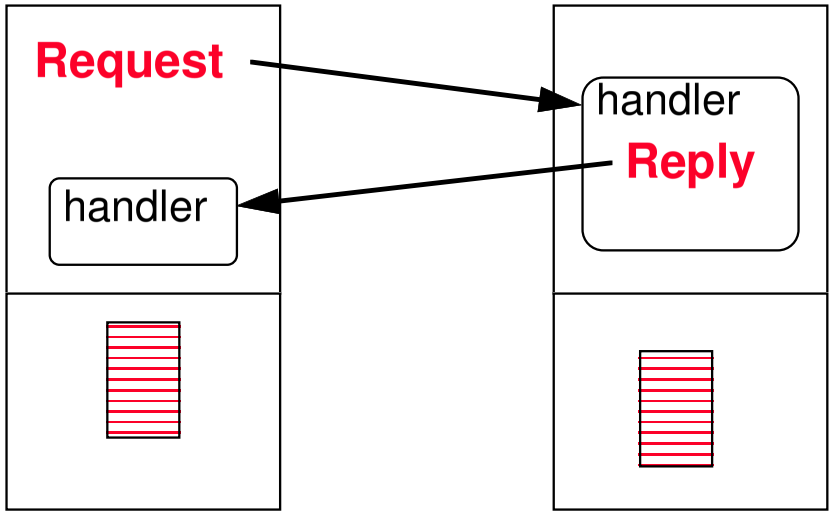
Towards fast, general purpose comm.



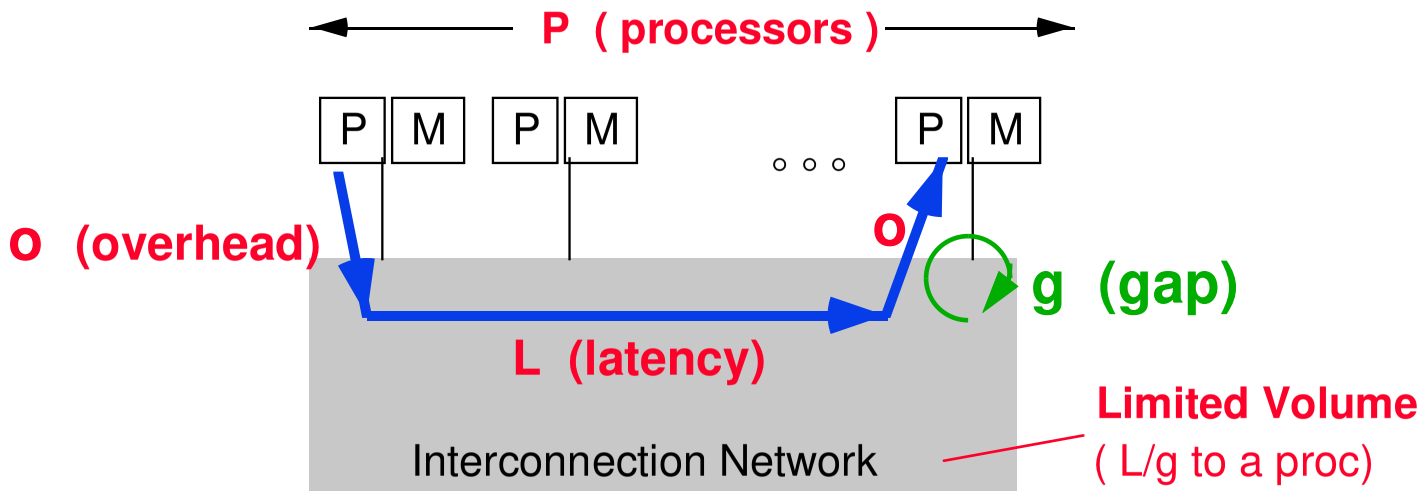
- Send by writing directly into the (virtual) network
- Read and handler directly from same

Generic Active Messages (GAM)

- **Highly optimized communication layer on a range of HW**
- **Request / Reply small active messages (RPC)**
- **Bulk-Transfer (store & get)**

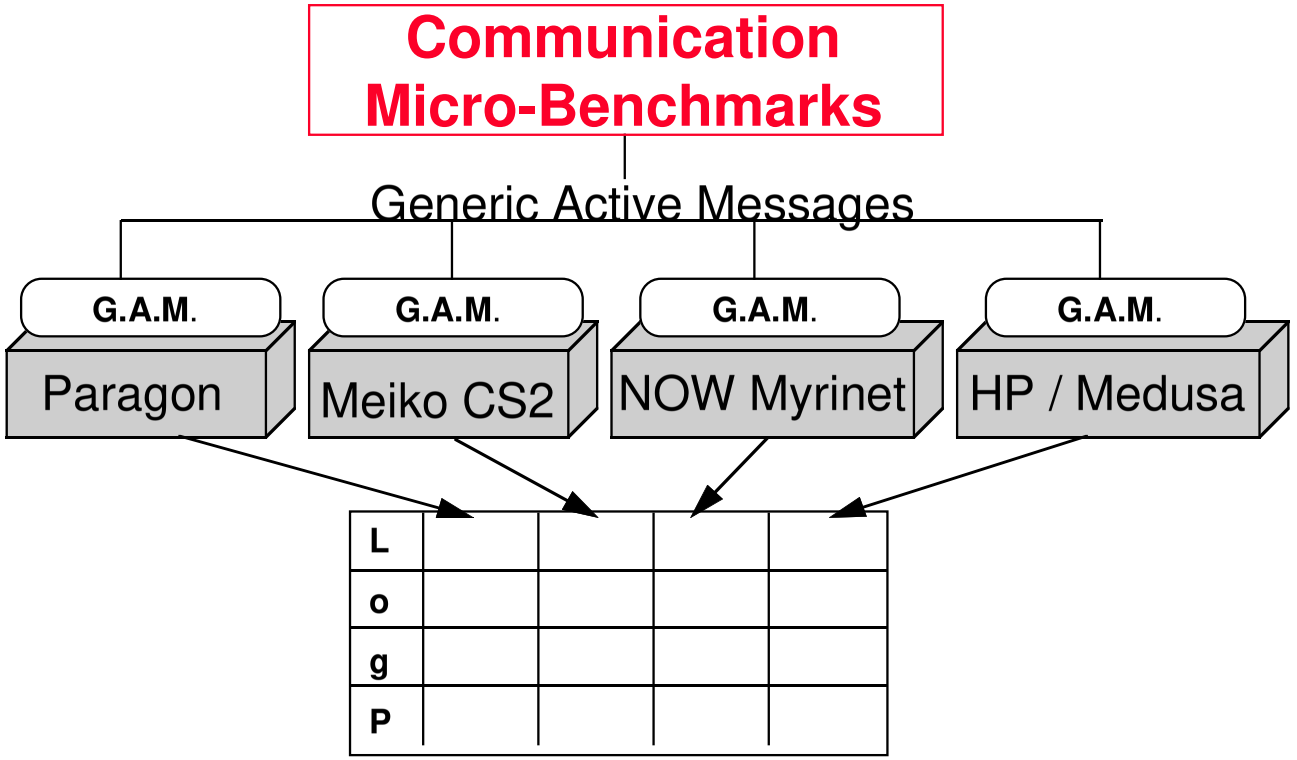


LogP Model of Comm. Costs



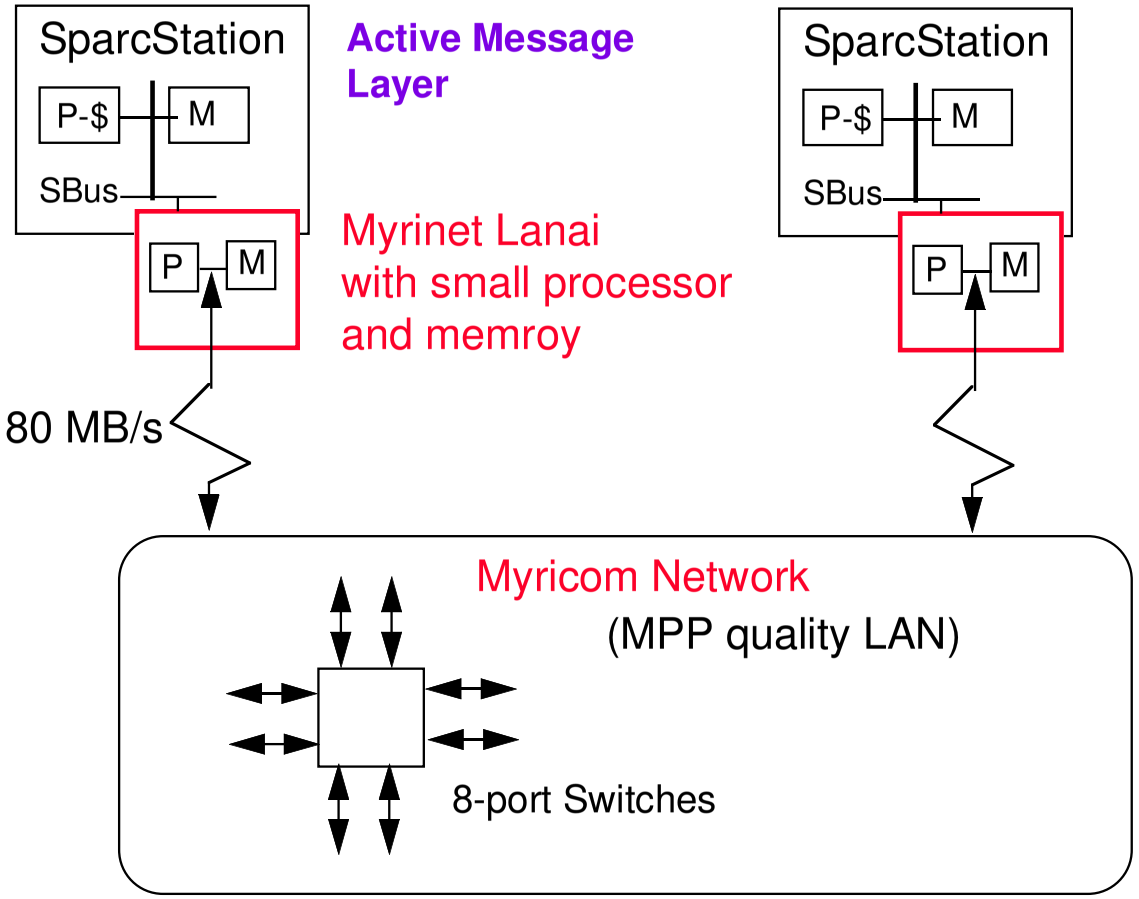
- **Latency in sending a (small) message between modules**
- **overhead felt by the processor on sending or receiving msg**
- **gap between successive sends or receives (1/BW)**
- **Processors**

Basic Approach

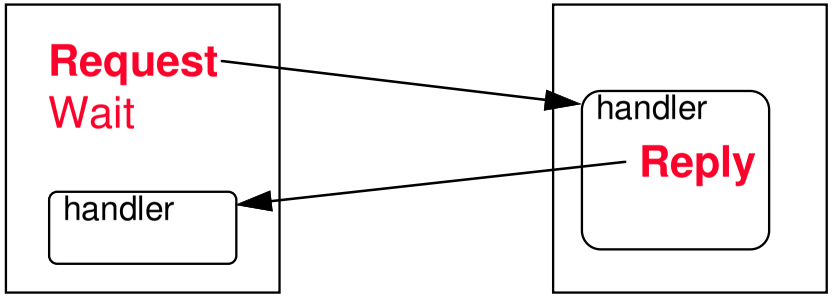


- **Assume the system behaves as LogP**
- **Conduct a set of GAM "gray box" experiments**
- **Analyze "signatures" to LogP deduce parameters**

NOW1 Prototype

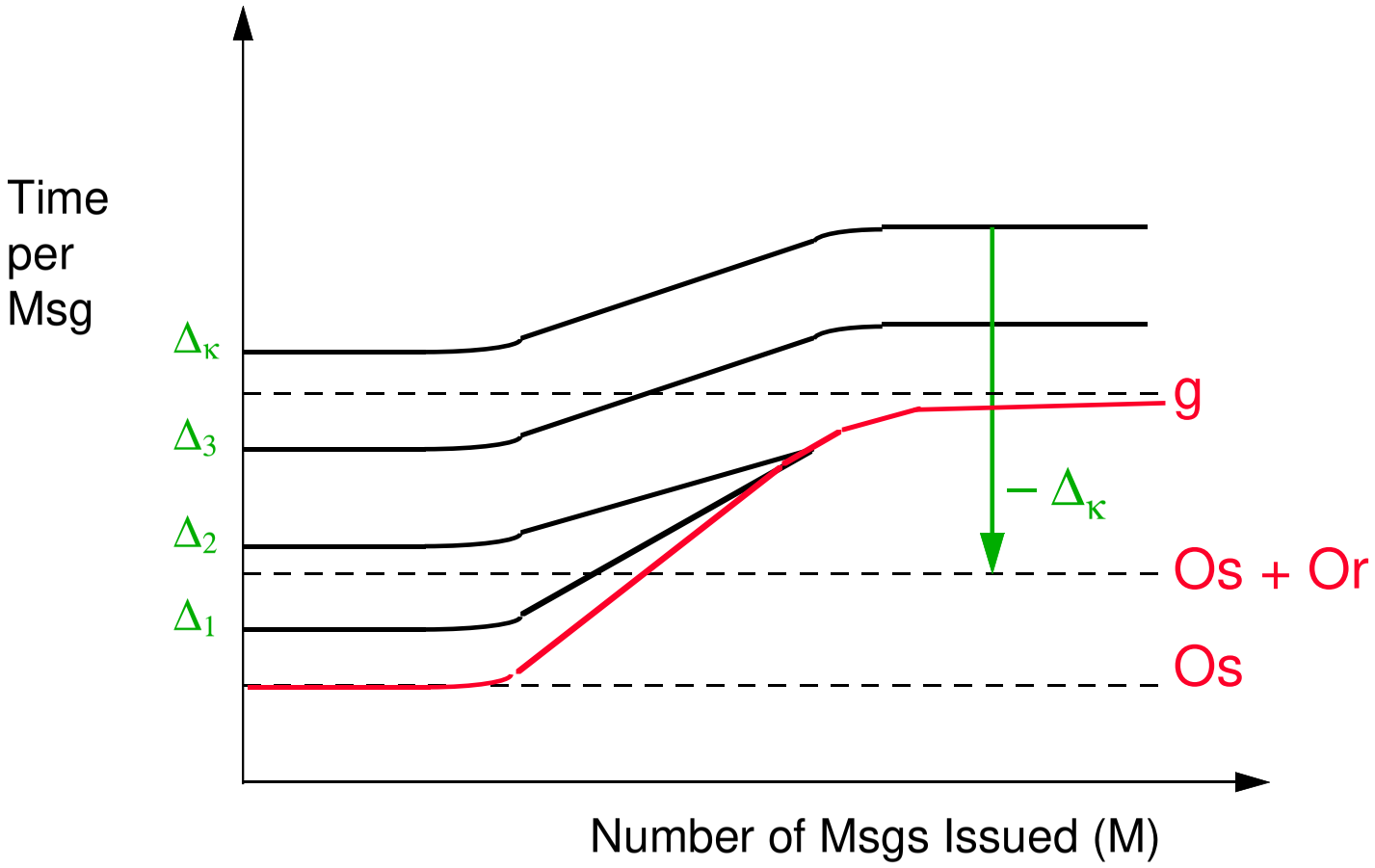


Time for a Complete Operation

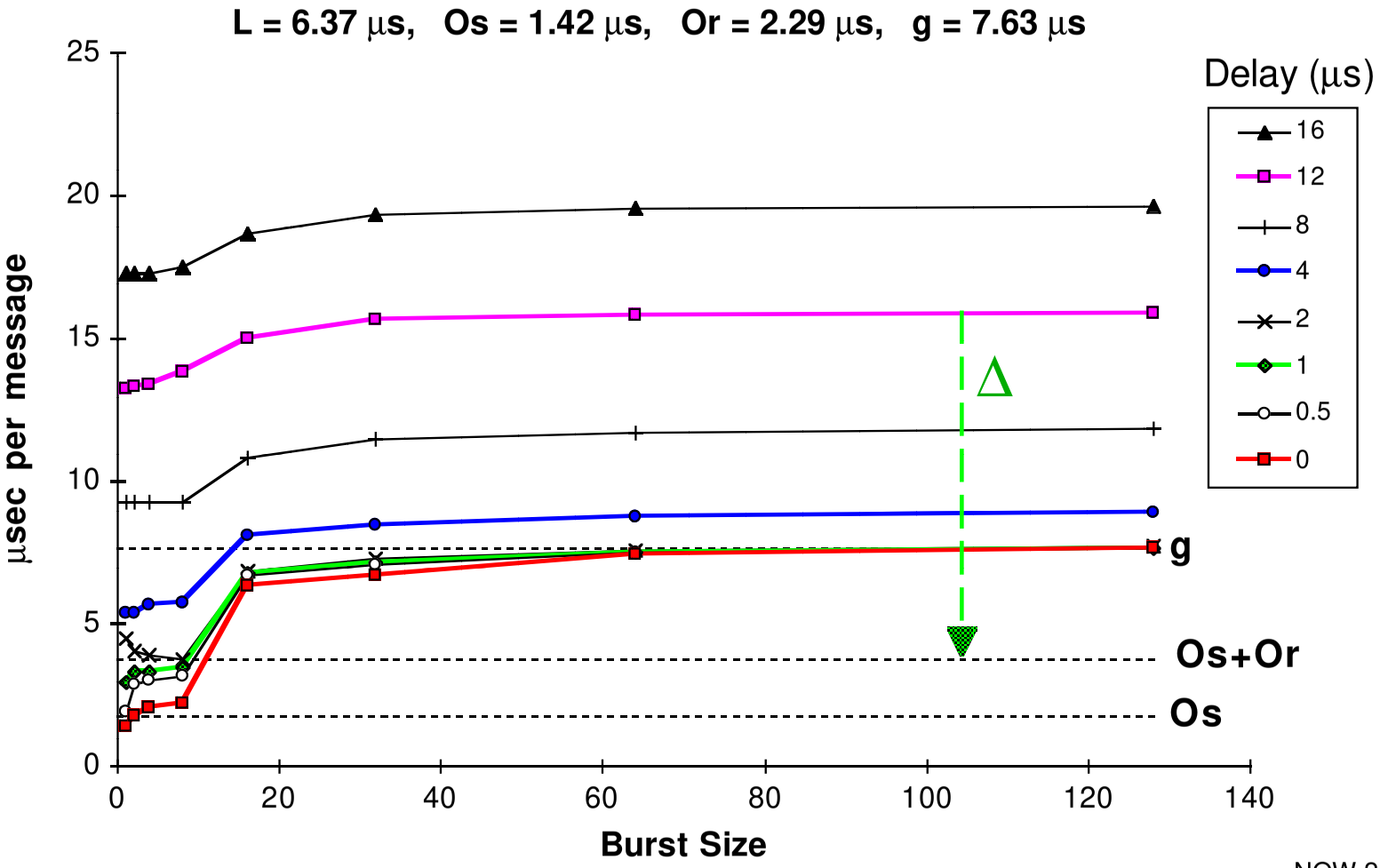


	Intel Paragon	Meiko CS-2	SS20 Myrinet	HP 735 Medusa
Round Trip (μs)	19.9 - 20.1	20.3 - 21.6	30.6 - 31.5	33.8 -
$2(O_s + L + O_r)$				

Extracting O and Gap from Issue Profile

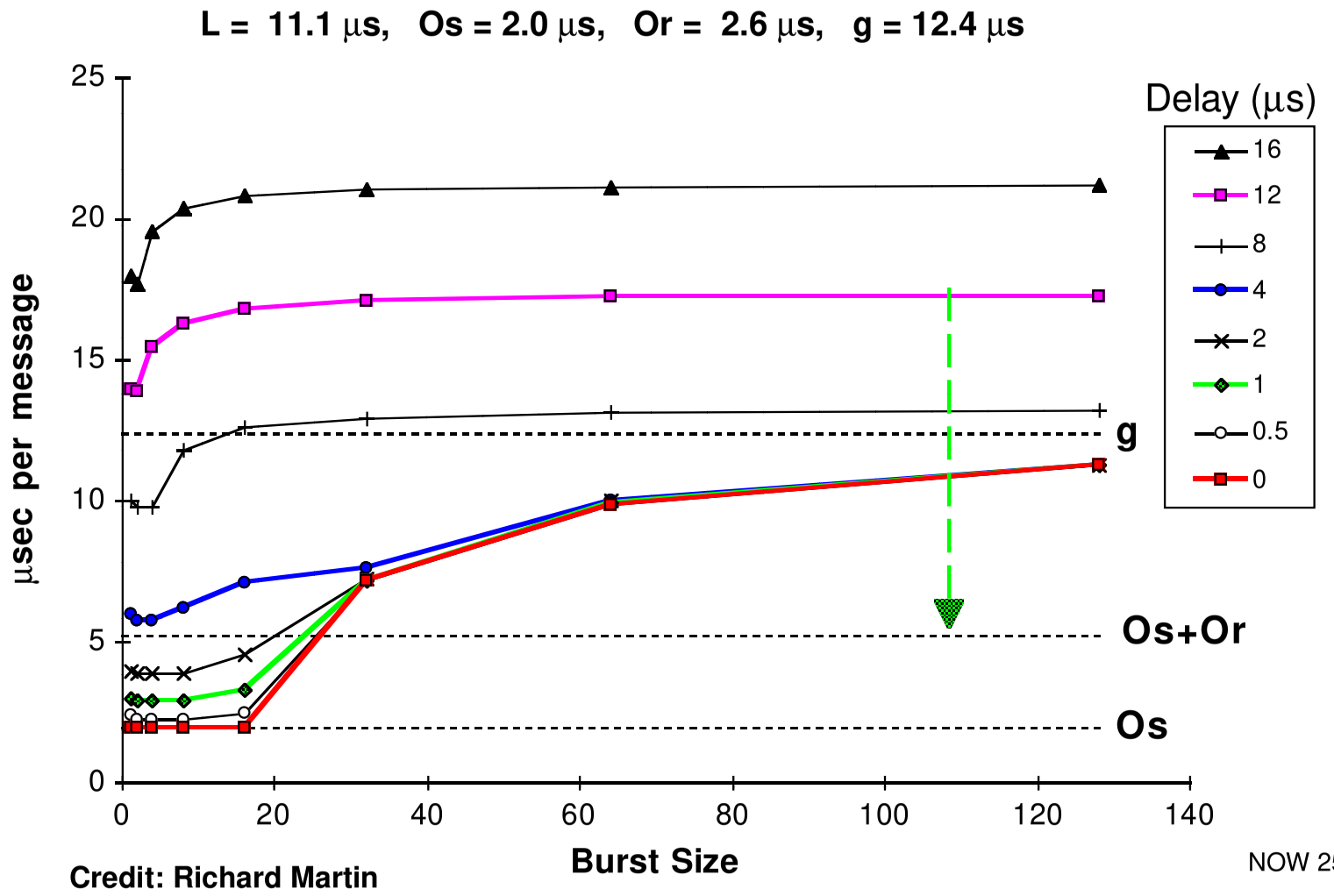


Paragon Microbenchmark Signature



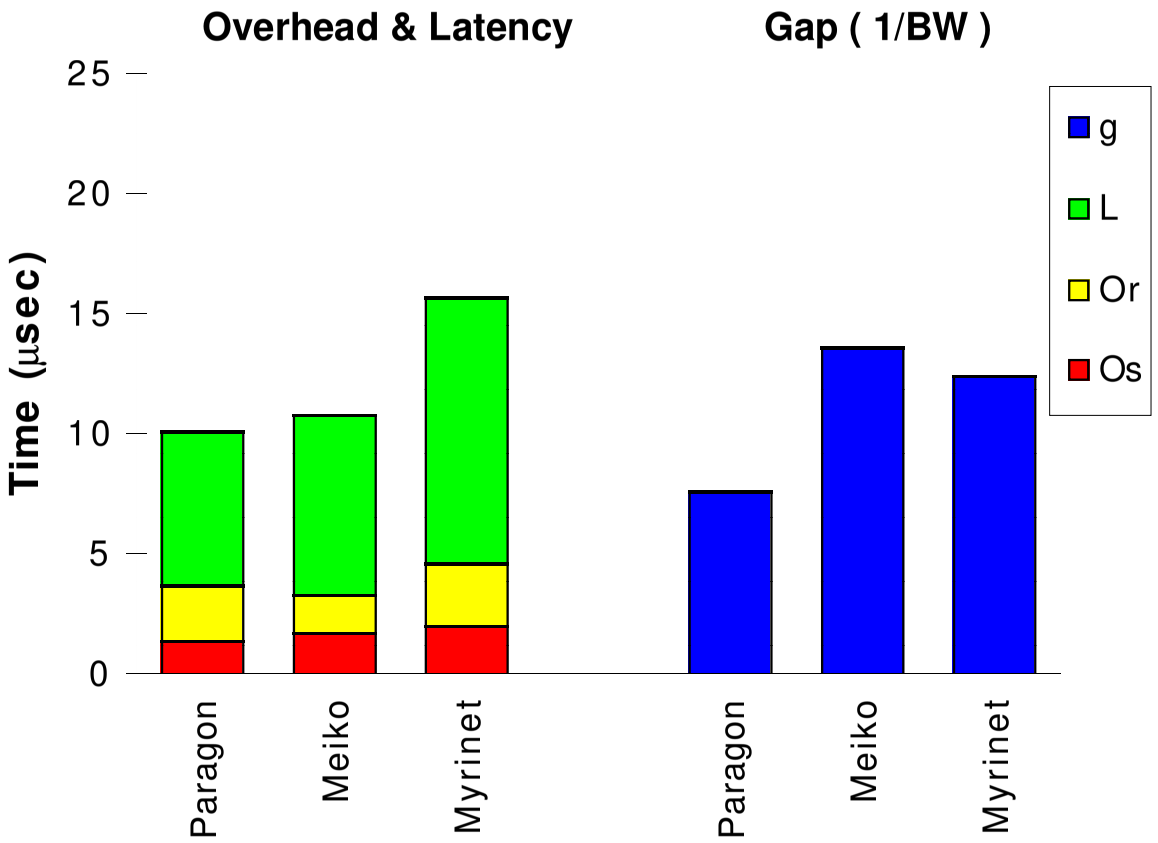
Credit: Lok Tin Liu

SS20 Myrinet Microbenchmark Signature

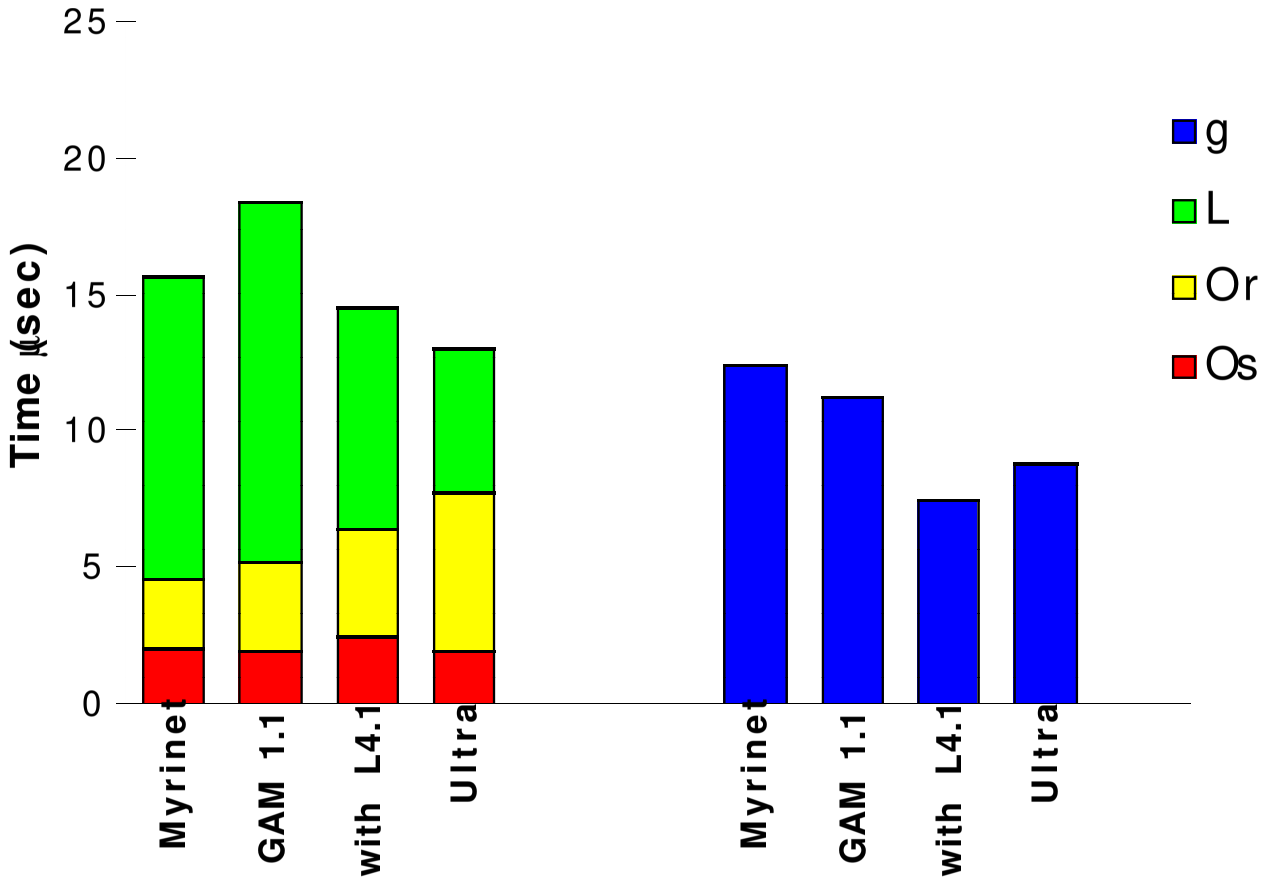


Credit: Richard Martin

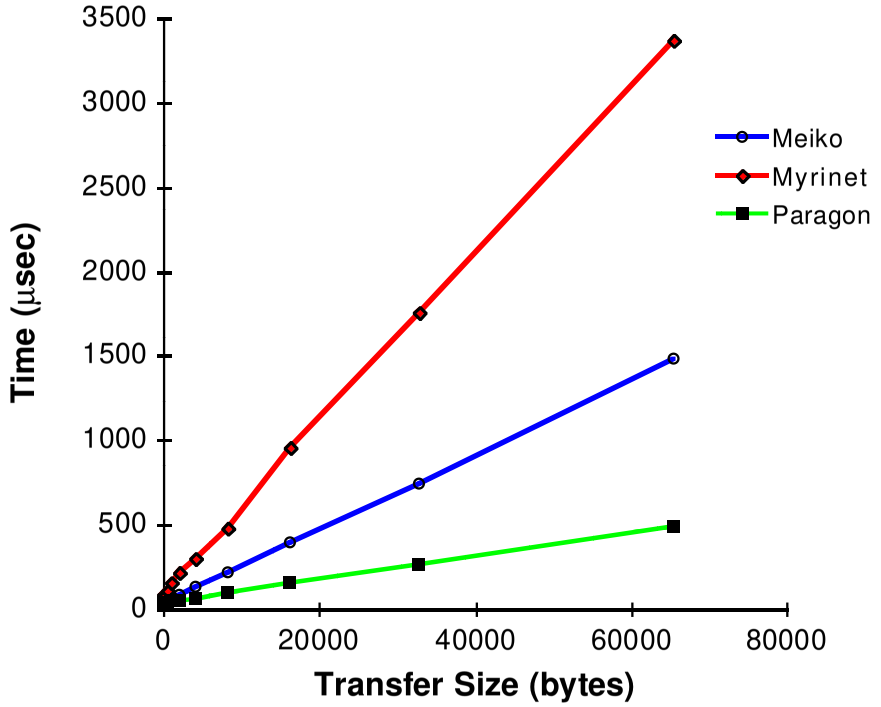
LogP Summary



Alternative NI Software & Hardware

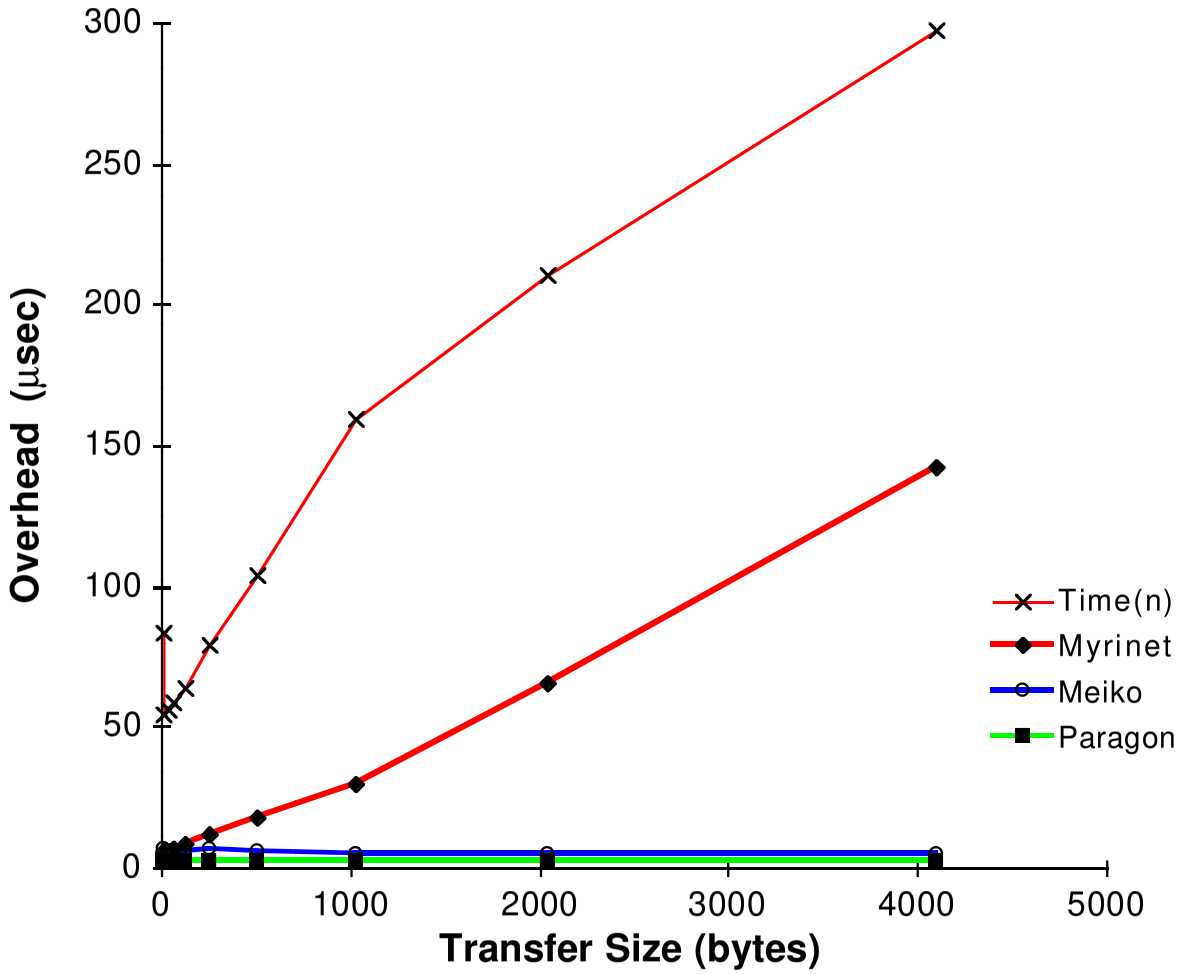


Single Bulk-Transfer Time: T(n)



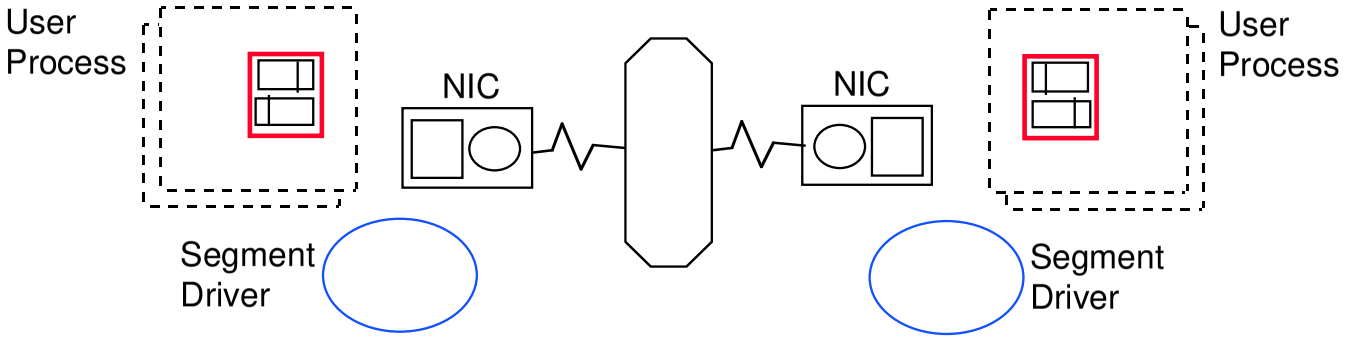
- **Typically Fit to linear model: $T_0 + n/B$**
  - **T_0 often meaningless**
  - **completion ambiguous**
  - **Does not reveal processing "overhead"**

Bulk Issue Overhead Comparison



AM – second cut

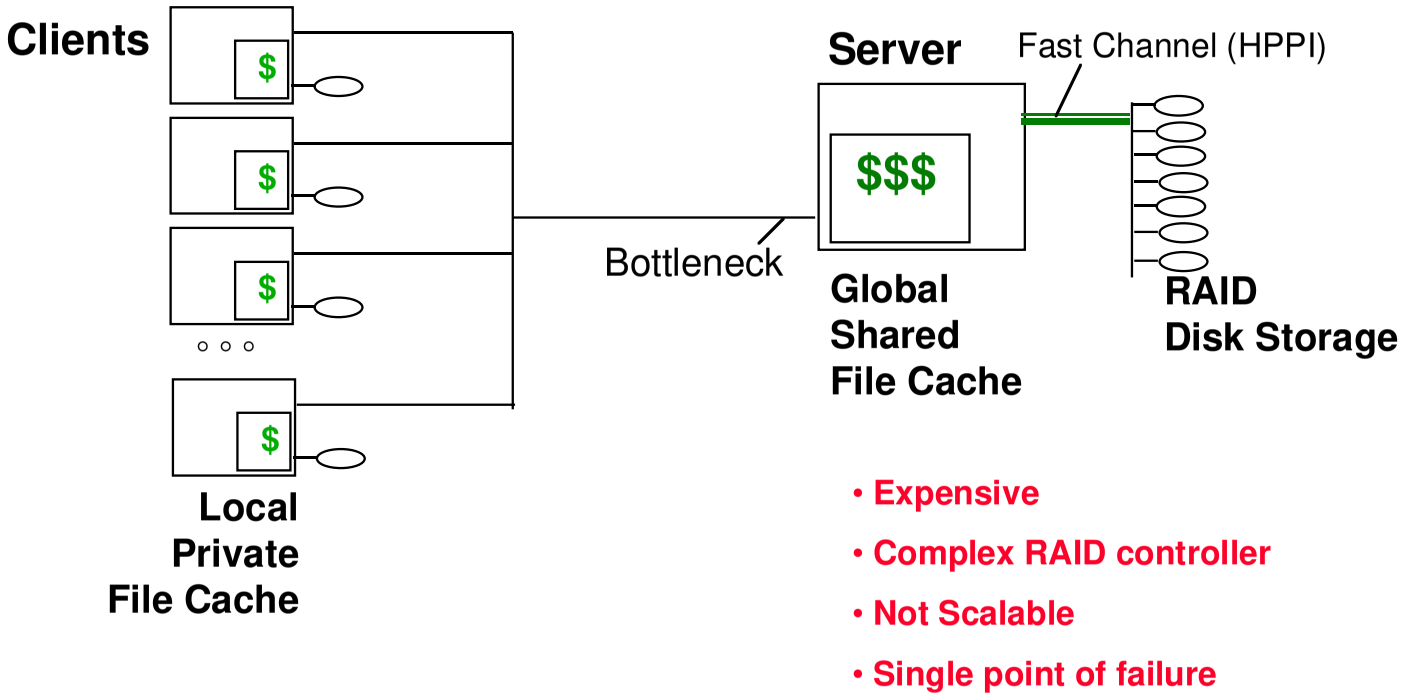
- Virtualize network as communication "endpoint"
  - send-pool, recv pool, translation table, key
  - event mask, condition variable
  - memory segment
- Allow many processes to have multiple endpoints
- Bind endpoints to comm. resource on demand
  - via Virtual Memory system
- "Smart" network interface provides routing, protection, error handling
  - return-to-sender error model



Scheduling Parallel Progs in a NOW

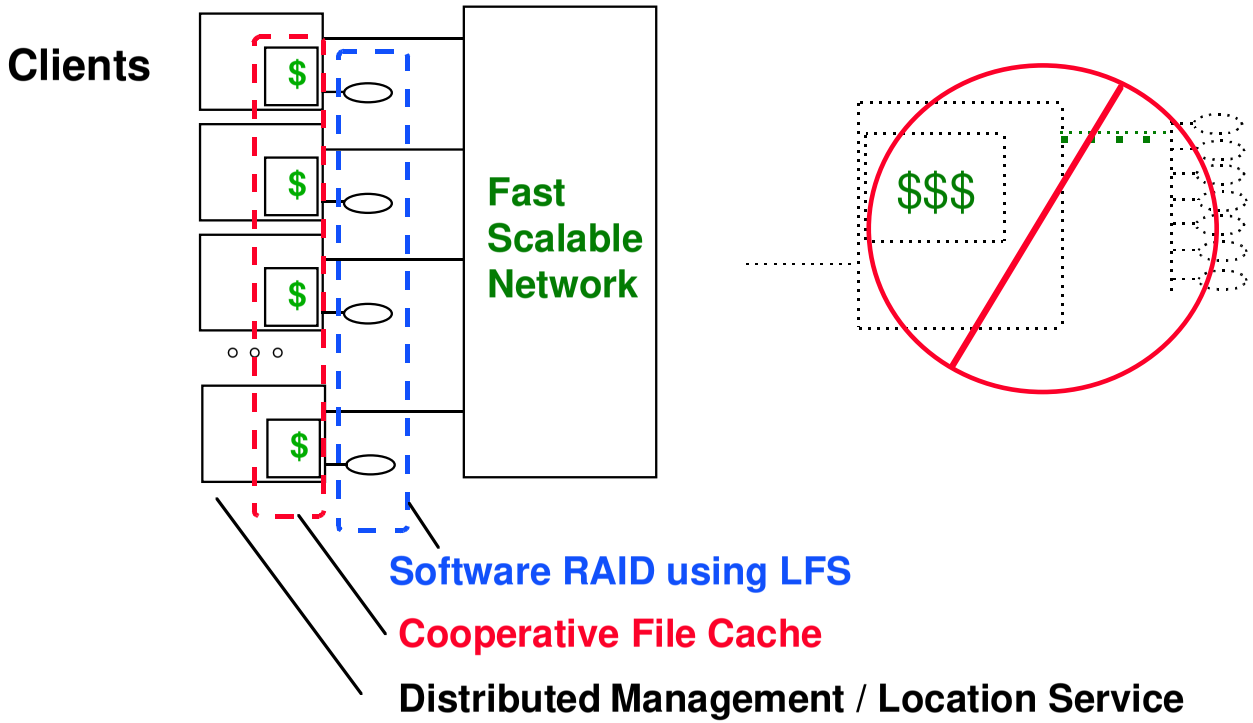
- **Conventional Wisdom: gang schedule or else!**
  - literature shows slowdowns of 10x - 100x
- **Explicit gang scheduling is expensive and inflexible**
- **Many interacting factors**
  - blocking algorithm
  - local scheduler
  - programming model
- **Implicit co-scheduling**
  - use natural dependences in the program and conventional OS interface to co-ordinate scheduling across processors
  - requires adaptive blocking and priority scheduling

Traditional Client / Server File system



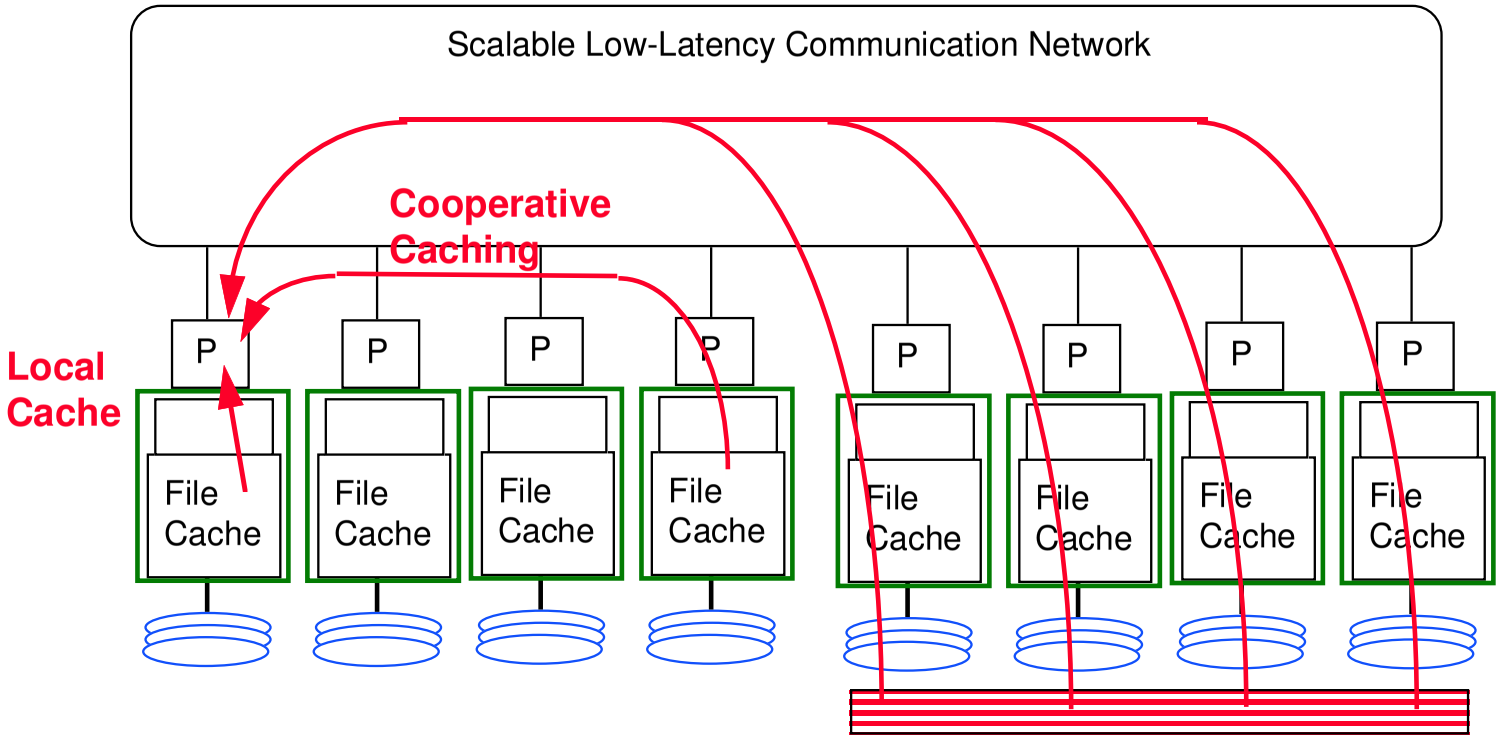
- **Server resources at a premium**
- **Client resources poorly utilized**

NOW Client / Serverless File System



- **Parallelize "everything" for performance and availability at low cost**

xFS Components

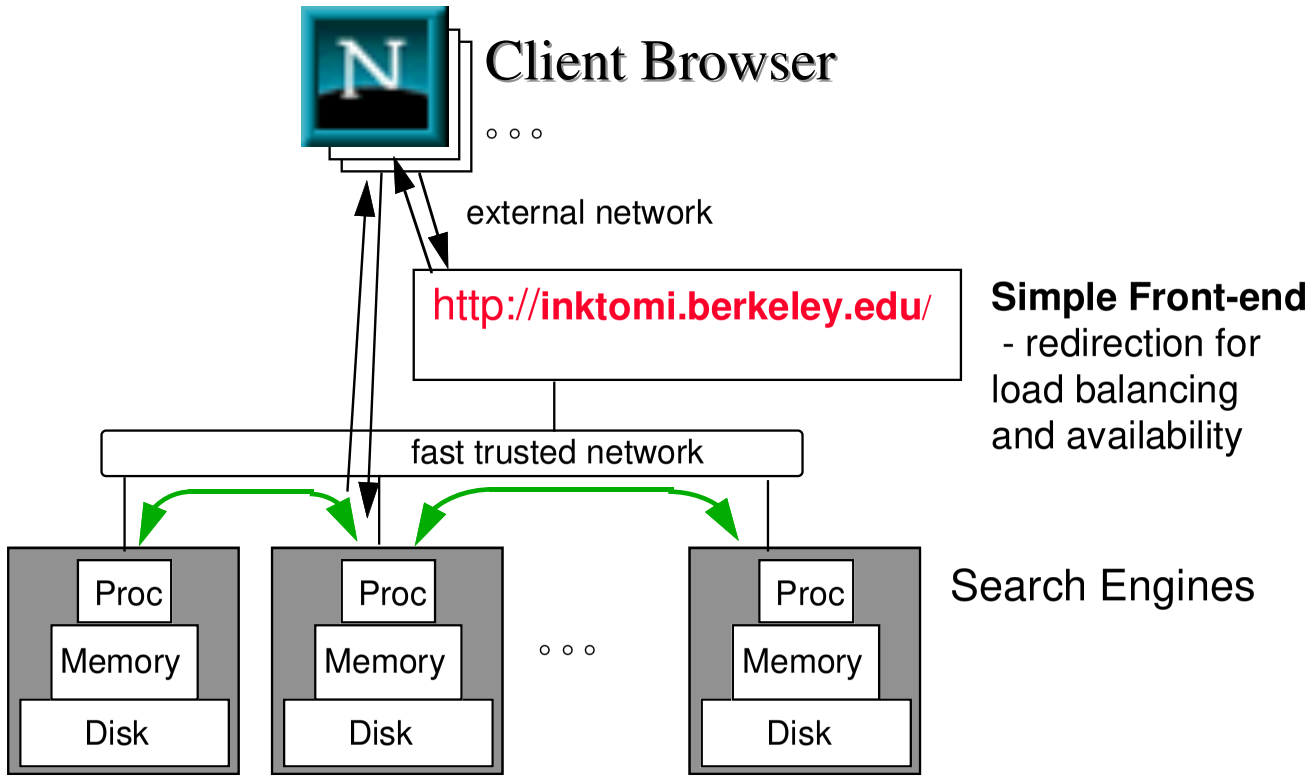


G = Node Comm BW / Disk BW

Scalable (web) Servers

- Deliver well-defined service from a known URL
- High capacity
  - it's exponential out there!
- Growth path
- Bursty Demands
- Availability
  - there's no "night" on the web

Inktomi Search Index and Crawler



Distributed query load, distribute the data base, parallel search, compact into memory

Other Components

- **Network virtual memory**
  - remote memory is closer than local disk
- **Global Management of Resources**
  - load balancing and migration of sequential prog.
- **Parallel Computing**
  - co-ordinated scheduling of parallel prog. processes
  - interaction with interactive work
- **System administration**

- **Self-tuning systems**
  - when everyone can do everything, who does what?

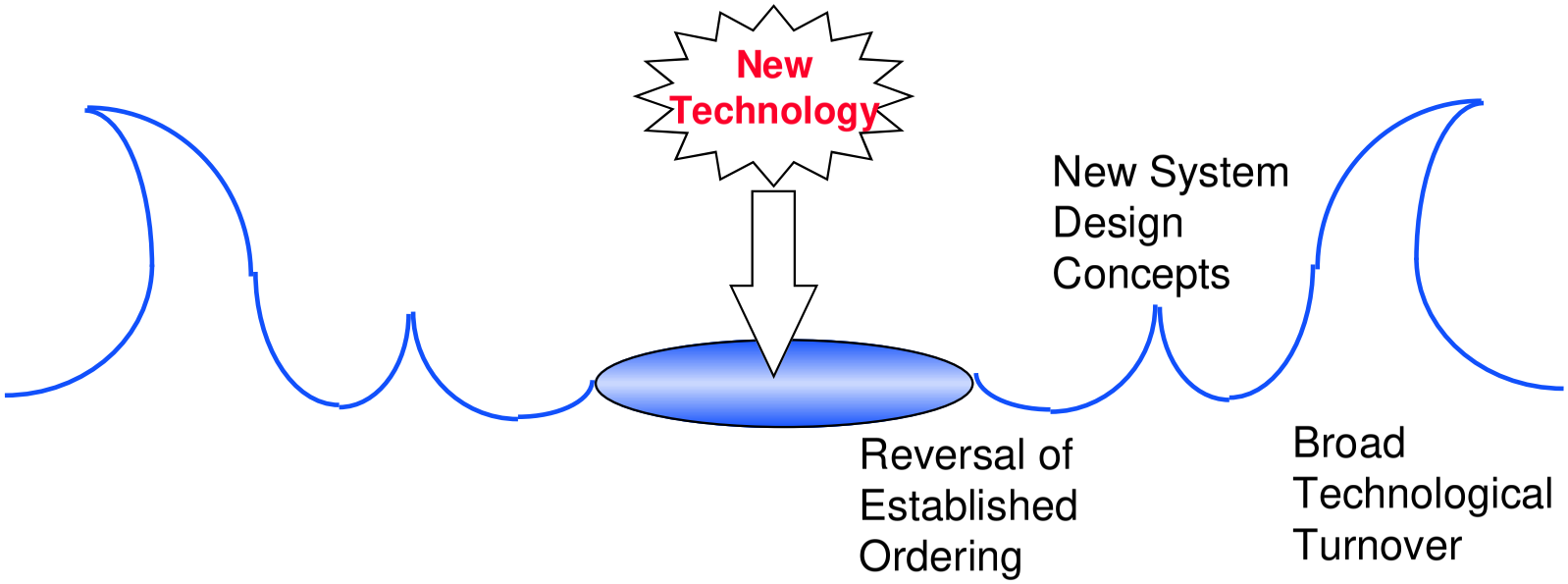
Desktop vs. Machine Room ?

- My view: BOTH !!!
- Desktop
  - Tremendous idle capacity, even at coarse grain
  - Very few unexpected reboots
  - Institutional infrastructure will support it
  - Want fast communication to my display
- Machine room
  - Simpler adminstration
  - Electrical vs optical
  - Always want some headroom in systems
  - Want baseline of resources for production
- Economics and workload will determine mix

Conclusion

- **Technologies aligned to exploit NOW now**
  - 32-bit μprocessors, switch based LANs, active messages, striped files, file caches, process migration
- **1990s building block is desktop HW&SW**
  - Need higher-level system research use building blocks: stand on shoulders, not toes
- **Challenges for NOW**
  - Leverage technology yet extend to make NOW ready
  - add low overhead user communication, global OS, serverless file system
- **More than Parallel Computing**
  - Make sequential users happy NOW
    - » recruit DRAM, disks, and CPUs
    - » provide cheap, high availability

Technology Ripples

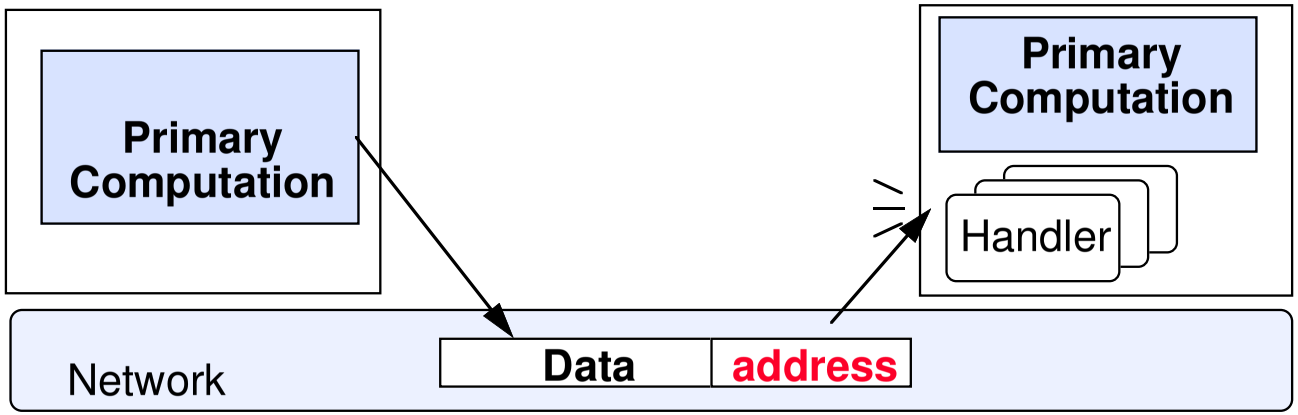


To find out more

- Anderson, Culler, Patterson "A Case for Networks of Workstations: NOW"*IEEE Micro*, Feb. 1995.
- Culler, et al. "Assessing Fast Network Interfaces", IEEE Micro, Feb. 1996
- Kronenberg, Levy, Strecker "Vax Clusters: A closely-coupled distributed system," *ACM Transactions on Computer Systems*, May 1986 v 4(3):130-146.
- Pfister, *In Search of Clusters – the coming battle in lowly parallel computing*, Prentice Hall, 1995, 415 pages.
- Dusseau, et al, "*Effective Distributed Scheduling of Parallel Workloads*", Sigmetrics 1996

- http://now.cs.Berkeley.edu/
  - Active Message Spec.

Active Messages



- **User generates msg => user knows how to handle it**
  - **Handler executes directly upon arrival**
  - **pulls message out of the network layer and integrates it into the computation or replies**
- **Simple protection, scheduling, allocation, liveness, . . .**

von Eicken, Culler, Goldstein, Schauser. "Active Messages: A Mechanism for Integrated Communication and Computation", ISCA 1992.