

---

# Distributed Systems

## Introduction

Paul Krzyzanowski  
pxk@cs.rutgers.edu

---

---

# Administrative stuff

---

---

## Web site and contact

[www.cs.rutgers.edu/~pxk/rutgers](http://www.cs.rutgers.edu/~pxk/rutgers)

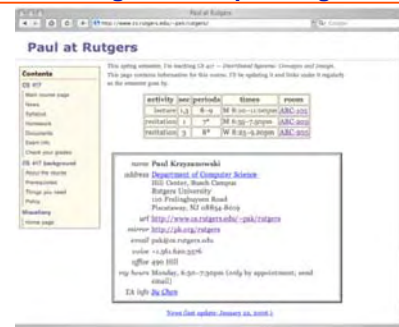
Mirror: [www.pk.org/rutgers](http://www.pk.org/rutgers)

email: [pxk@cs.rutgers.edu](mailto:pxk@cs.rutgers.edu)

---

---

## [www.cs.rutgers.edu/~pxk/rutgers](http://www.cs.rutgers.edu/~pxk/rutgers)



---

## News



---

## Text



*Distributed Systems:  
Principles and Paradigms*

Andrew S. Tannenbaum  
Maarten van Steen

Prentice Hall  
August 2002  
ISBN: 0130888931

**OPTIONAL**  
**Don't waste your money**

---

## Policy

---

### Attendance

### Assignments

- Due dates
- No MS-Word submissions!  
PDF or text with line breaks only.

### Collaboration

- Cheating

### Office hours

### Things that make noise:

- Pagers, PDAs, cell phones
  - Fine: \$10·2<sup>n-1</sup>
- 

## Grades

---

- 4 Exams (drop lowest grade)
  - + written assignments (3-4?)
  - + programming assignments (4?)
  - attendance/quizzes
- 
- = Final Grade
- 

## Grades

---

No make-up exams

If you don't do the programming assignments  
you fail

If you cheat  
you fail

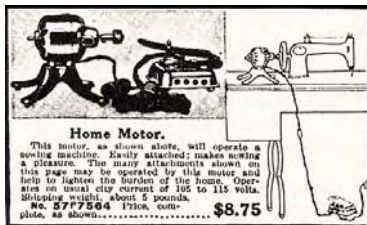
---

on with the course...

---

## Non-ubiquity

---

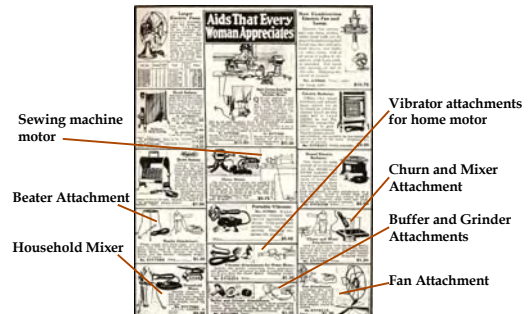


Source: 1918 Sears Roebuck catalogue via  
The Invisible Computer, Donald A Norman, 1999

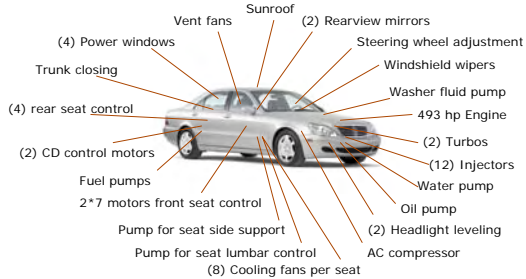
---

## Non-ubiquity

---



## Ubiquity



## Ubiquity

Over 50 microcontrollers  
 600 signal lines  
 150 message types multiplexed onto three busses  
 Over 600,000 lines of code

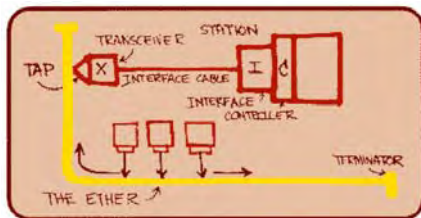


What can we do now  
 that we could not do before?

## Technology advances

Networking  
 Processors  
 Memory  
 Storage  
 Protocols

## Networking: Ethernet - 1973, 1976



**June 1976:** Robert Metcalfe presents the concept of *Ethernet* at the National Computer Conference  
**1980:** Ethernet introduced as de facto standard (DEC, Intel, Xerox)

## Network architecture

### LAN speeds

- Original Ethernet: 2.94 Mbps
- **1985:** thick Ethernet: 10 Mbps  
1 Mbps with twisted pair networking
- **1991:** 10BaseT - twisted pair: 10 Mbps  
Switched networking: **scalable bandwidth**
- **1995:** 100 Mbps Ethernet
- **1998:** 1 Gbps (Gigabit) Ethernet
- **1999:** 802.11b (wireless Ethernet) standardized
- **2001:** 10 Gbps introduced
- **2005:** 100 Gbps demonstrated (over optical link)

10,000x  
faster

## Network Connectivity

180,000  
more hosts

### Then:

- large companies and universities on Internet
- gateways between other networks
- dial-up bulletin boards
- 1985: 1,961 hosts on the Internet

### Now:

- One Internet (mostly)
- 2005: 353,284,187 hosts on the Internet
- widespread connectivity  
High-speed WAN connectivity: 1- >10 Mbps
- Switched LANs
- wireless networking

## Computing power

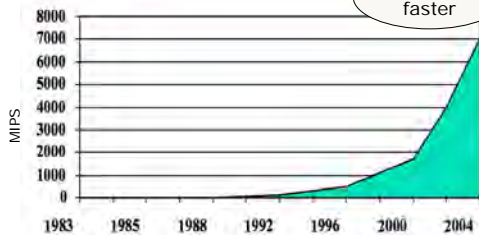
### Computers got...

- Smaller
- Cheaper
- Power efficient
- Faster

Microprocessors became technology leaders

## Computing power

1,700x  
faster



1974: Intel 8080: 2 MHz; 6K transistors  
2004: Intel P4 Prescott: 3.6 GHz, 125M transistors  
2004: HP PA-8800: 300 million transistors - 6.5GB/sec BW

## Storage: RAM

9,000x cheaper  
4,000x more capacity

year	\$/MB	typical
1977	\$32,000	16K
1987	\$250	640K-2MB
1998	\$2	64MB-256MB
1999	\$1	64MB-256MB
2000	\$0.68	128MB-1GB
2001	\$0.41	128MB-2GB
2002	\$0.08	256MB-2GB
2003	\$0.07	256MB-2GB+
2005	\$0.06	512MB-2GB+

## Storage: disk

99,000x cheaper  
30,000x more capacity

year	\$/GB	typical
1977	\$4,800,000	310 KB
1987	\$17,400	40 MB
1998	\$20.48	6-12 GB
1999	\$16.38	10-20 GB
2000	\$5.43	10-75 GB
2001	\$2.76	20-80 GB
2002	\$1.13	40-160 GB
2003	\$0.676	60-250 GB
2004	\$0.43	80-320 GB
2005	\$0.316	80-500 GB

## Protocols

### Faster CPU →

**more time for protocol processing**

- ECC, checksums, parsing (e.g. XML)
- Image, audio compression feasible

### Faster network →

**→ bigger (and bloated) protocols**

- e.g., SOAP/XML, H.323

---

**Just because we can...**

---

### **Why do we want to network?**

---

- Price/performance ratio
    - don't get twice computing for twice the price with multiprocessors
  - Distributing applications may make sense
    - ATMs, graphics, remote monitoring
  - Interactive communication & entertainment
    - work and play together:
      - email, gaming, telephony, instant messaging
  - Remote content
    - web browsing, music & video downloads, IPTV, file servers
  - Mobility
  - Increased reliability
  - Incremental growth
    - how do you upgrade a mainframe?
- 

### **Media delivery and storage**

---

#### More storage:

- Store rich media
  - 300 GB, 3.5 MB/song = >87,000 songs
  - 300 GB, 1.5 MB/photo = >200,000 photos
  - 300 GB, 78 MB/photo → <4,000 photos

#### Faster network:

- Faster downloads
    - 2.5 Mbps, 3.5 MB/song = 11 seconds
    - 2.5 Mbps, 1.5 MB/photo = 5 seconds
- 

### **Media delivery and storage**

---

#### Audio

- MPEG-3 telephone quality: 8kbps
- MPEG-3 FM quality: 56-64 kbps **36 hours/GB**
- Telephone audio: 64 kbps
- MPEG-3 compressed near-CD audio: 96kbps
- MPEG-3 CD quality audio: 112 Kbps **20 hours/GB**
- CD quality audio: 1.4 Mbps
- HDTV audio: 3 Mbps

#### Video

- Mpeg-1 compressed NTSC video: 1.5 Mbps **1.5 hours/GB**
- Jpeg compressed NTSC video: 3-7 Mbps
- HDTV video: 22-40 Mbps (MPEG-2) **4.5 minutes/GB**

#### Digital movies

- Compressed: 45-65 GB/movie
  - Decompressed: 1.6 TB
  - Stream at 1.5 Gb/sec
- 

### **Problems**

---

#### Designing distributed software can be difficult

- Operating systems handling distribution
- Programming languages?
- Efficiency?
- Reliability?
- Administration?

#### Network

- disconnect, loss of data, latency

#### Security

- want easy and convenient access
- 

---

## **Building and classifying distributed systems**

---

## Flynn's Taxonomy (1972)

number of instruction streams  
and number of data streams

### SISD

- traditional uniprocessor system

### SIMD

- array (vector) processor
- Examples:
  - APU (attached processor unit in Cell processor)
  - SSE3: Intel's Streaming SIMD Extensions
  - PowerPC Altivec (Velocity Engine)

### ~~MIMD~~ MIMD

- multiple computers, each with:
  - program counter, program (instructions), data
- parallel and distributed systems

## Subclassifying MIMD

### memory

- shared memory systems: multiprocessors
- no shared memory: networks of computers, multicomputers

### interconnect

- bus
- switch

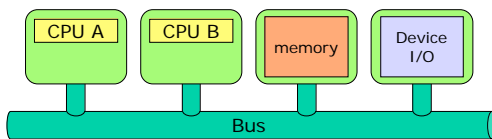
### delay/bandwidth

- tightly coupled systems
- loosely coupled systems

## Bus-based multiprocessors

All CPUs connected to one bus (backplane)

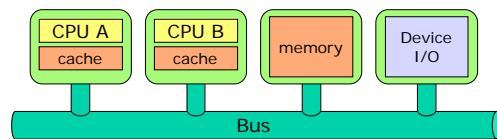
Memory and peripherals are accessed via shared bus



## Bus-based multiprocessors

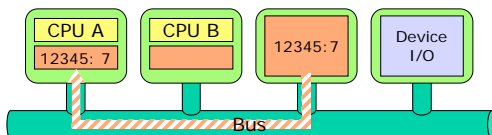
Dealing with bus overload  
- add local memory

CPU does I/O to cache memory  
- access main memory on cache miss



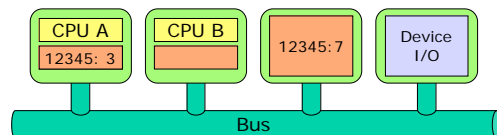
## Working with a cache

CPU A reads location 12345 from memory



## Working with a cache

CPU A modifies location 12345

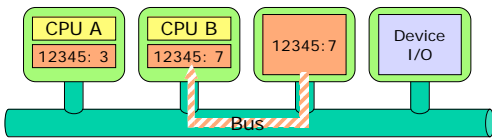


### Working with a cache

CPU B reads location 12345 from memory

Gets old value

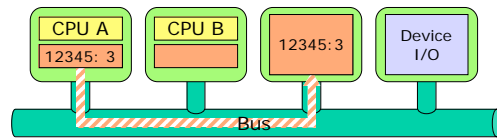
Memory not coherent!



### Write-through cache

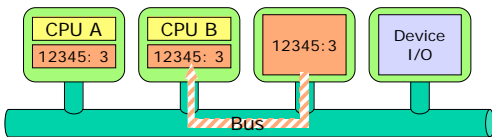
Fix coherency problem by writing all values through bus to main memory

CPU A modifies location 12345 – *write-through*  
main memory is now coherent



### Write-through cache ... continued

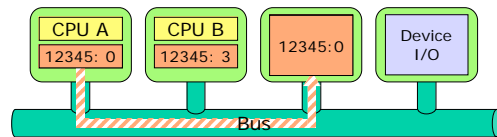
CPU B reads location 12345 from memory  
- loads into cache



### Write-through cache

CPU A modifies location 12345  
- write-through

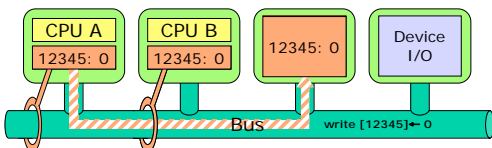
Cache on CPU B not updated  
Memory not coherent!



### Snoopy cache

Add logic to each cache controller:  
monitor the bus

Virtually all bus-based architectures use a  
snoopy cache

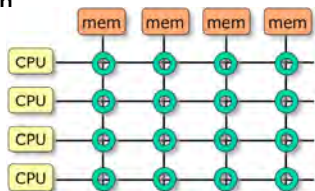


### Switched multiprocessors

Bus-based architecture does not scale to  
a large number of CPUs (32, 64)

### Switched multiprocessors

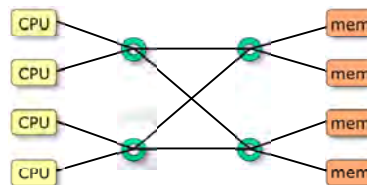
divide memory into groups and connect chunks of memory to the processors with a **crossbar switch**



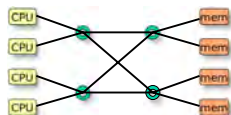
$n^2$  crosspoint switches – expensive switching fabric

### Crossbar alternative: omega network

Reduce crosspoint switches by adding more switching stages



### Crossbar alternative: omega network



with  $n$  CPUs and  $n$  memory modules:  
need  $\log_2 n$  switching stages,  
each with  $n/2$  switches

Total:  $(n \log_2 n) / 2$  switches.

- Better than  $n^2$  but still a quite expensive
- delay increases:
  - 1024 CPU and memory chunks
  - overhead of 10 switching stages to memory and 10 back.

### NUMA

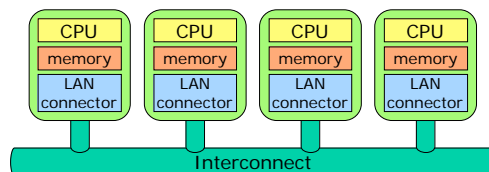
- Hierarchical Memory System
- Each CPU has local memory
- Other CPU's memory is in its own address space
  - slower access
- Better *average* access time than omega network if most accesses are local
- Placement of code and data becomes difficult
- SGI Origin's ccNUMA

### Bus-based multicomputers

- No shared memory
- Communication mechanism needed on bus
  - Traffic much lower than memory access
  - Need not use physical system bus
    - Can use LAN (local area network) instead

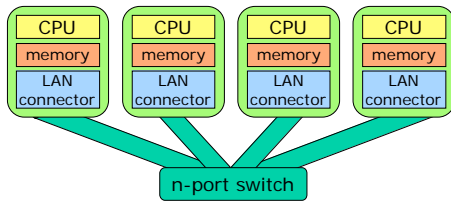
### Bus-based multicomputers

Collection of workstations on a LAN



## Switched multicomputers

Collection of workstations on a LAN



## Software

### Single System Image

Collection of *independent* computers that appears as a *single system* to the user(s)

- *Independent*: autonomous
- *Single system*: user not aware of distribution

### Distributed systems software

Responsible for maintaining single system image

## Loosely vs. Tightly coupled software

### Loosely coupled

- Limited interaction
- Mostly fully-functioning stand-alone machines
- If network goes down, processes can run
- E.g., share printer, backup devices, web

### Tightly coupled

- Strong dependence on other machines for all aspects of system

## Loosely coupled SW, loosely coupled HW

- Most common environment now
- Workstations on LAN
  - Each with own OS
- Primitive *explicit* interaction with others
  - e.g., rcp, rlogin, ftp, http
- File servers on network
  - Accept requests for files & provide data
  - *File access transparency*
- High degree of autonomy

## Tightly coupled SW, loosely coupled HW

Make network of machines appear as one timesharing system

- Single system image

User should not be aware of multiple systems

- Gives us true distributed system

## Tightly coupled SW, loosely coupled HW

To achieve this we need:

- **single global IPC mechanism**
  - Any process can talk to any other without caring whether it is local or remote
- Global protection scheme
- Consistent process management
- Uniform resource naming
  - File system looks the same with same permissions
- Same system call interface
  - Each kernel controls its own resources (memory/paging)

## Design issues: Transparency

High level: hide distribution from users

Low level: hide distribution from software

- **Location transparency:** users don't care where resources are
- **Migration transparency:** resources move at will
- **Replication transparency:** users cannot tell whether there are copies of resources
- **Concurrency transparency:** users share resources transparently
- **Parallelism transparency:** operations take place in parallel without user's knowledge

## Design issues

### Reliability

- **Availability:** fraction of time system is usable
  - Achieve with redundancy
- **Reliability:** data must not get lost
  - Includes security

### Performance

- Communication network may be slow and/or unreliable

### Scalability

- Distributable vs. centralized algorithms
- Can we take advantage of having lots of computers?

## Service Models

### Centralized model

- No networking
- Traditional time-sharing system
- Direct connection of user terminals to system
- One or several CPUs
- Not easily scalable
- Limiting factor: number of CPUs in system
  - Contention for same resources



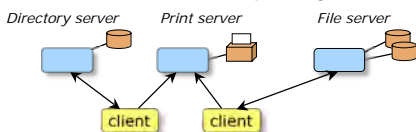
### Client-server model

Environment consists of **clients** and **servers**

**Service:** task machine can perform

**Server:** machine that performs the task

**Client:** machine that is requesting the service



### Workstation model

assume client is used by one user at a time

### Peer to peer model

- Each machine on network has (mostly) equivalent capabilities
- No machines are dedicated to serving others
- E.g., collection of PCs:
  - Access other people's files
  - Send/receive email (without server)
  - Gnutella-style content sharing
  - SETI@home computation

### **Processor pool model**

---

What about idle workstations  
(computing resources)?

- Let them sit idle
- Run jobs on them

Alternatively...

- Collection of CPUs that can be assigned processes on demand
  - Users won't need heavy duty workstations
    - GUI on local machine
  - Computation model of Plan 9
- 

### **Grid computing**

---

Provide users with seamless access to:

- Storage capacity
- Processing
- Network bandwidth

Heterogeneous and geographically distributed systems

---

### **Thin clients**

---

Examples:

- X terminal (e.g. NCD, PC running Exceed, ...)
- Network PC (Intel/Microsoft's NetPC)
- Network computer (NC)
- Web pad, set-top box (information appliance)

No need for:

- Much administration
- Expansion slots
- Disks (sometimes)

Definitions vary

---

### **Thick clients**

---

Example:

- Conventional PC running Windows XP, Linux

Client performs bulk of data processing operations

Data stored on server

---

---

### **Multi-tier client-server architectures**

---

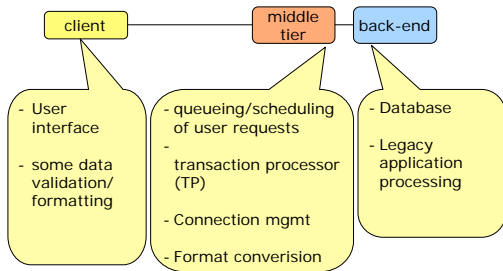
### **Two-tier architecture**

---

- Common from mid 1980's-early 1990's
  - UI on user's desktop
  - Application services on server
  - Performance deteriorates with large user communities
    - Server may spend a significant amount of time managing connections
    - Legacy services may have to run on environments poorly adapted fo networking
    - Databases are performance hogs
-

### Three-tier architecture

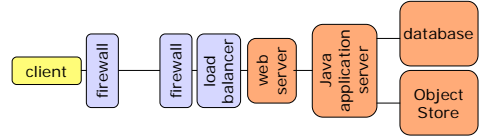
---



### Beyond three tiers

---

Most architectures are multi-tiered



---

**The end**

---