

# Interconnecting Computers: Architecture, Technology, and Economics

Butler Lampson

Modern computer systems have a recursive structure of processing and storage elements that are interconnected to make larger elements. Above the lowest level of transistors and gates, the essential character of these connections changes surprisingly little over about nine orders of magnitude in time and space. Here are some examples:

- Functional units connected to registers and on-chip cache.

- Processor chips connected to cache memories.

- Multiple processors and caches connected to main memories.

- Computing nodes connected by a message-passing LAN.

- LANs bridged to form an extended LAN.

- Networks connected in a wide-area internet.

- All the computers in the world exchanging electronic mail.

The important properties of connections are latency, bandwidth, connectivity, availability, and cost. The basic mechanism for connecting lots of things is multiplexing, but there are many ways to implement it. This lecture describes some of these ways and considers how different levels of the recursive hierarchy interact.

# Interconnecting Computers: Architecture, Technology, and Economics

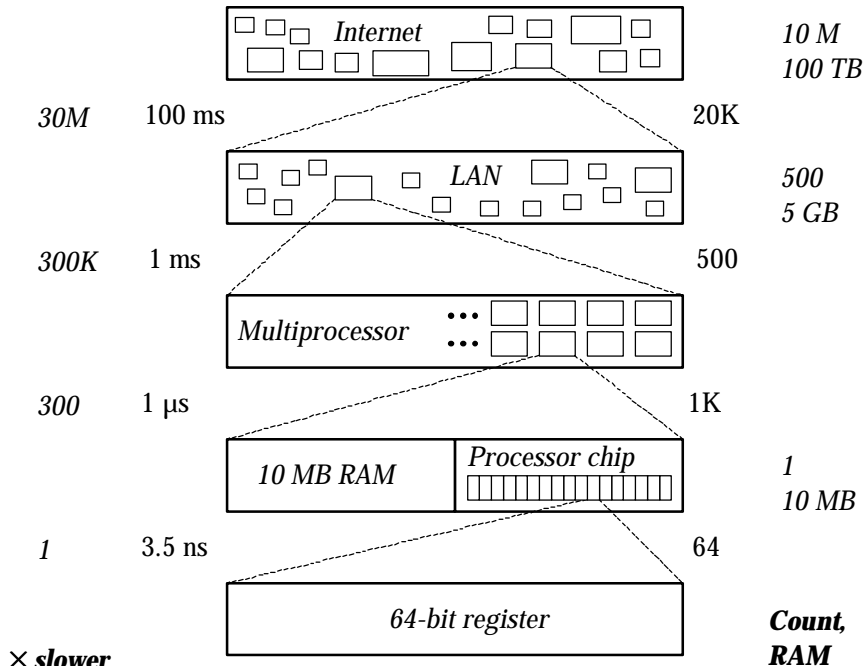
Computer systems are made up of interconnected subsystems.

The structure and methods are similar across the board.

Structure = architecture

Methods = technology

Better silicon and fiber optics make all the subsystems more alike.



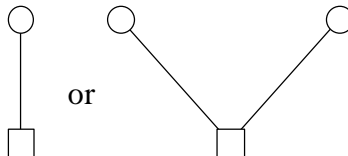
How fast?

How many?

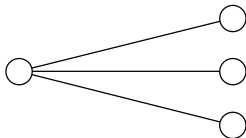
# Messages

# Storage

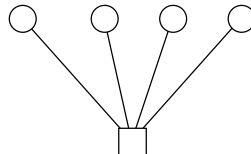
*Two-party*



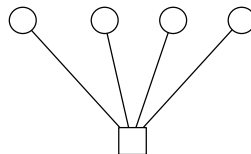
*Broadcast*



Storage is one party, or not.



*Multi-party*



Left and right nodes can be the same

# Examples of Messages

<i>System</i>	<i>Address</i>	<i>Sample address</i>	<i>Delivery</i>	
			<i>Ord-ered</i>	<i>Reli-able</i>
J-machine	source route	4 north, 2 east	yes	yes
802 LAN	6 byte flat	FF F3 6E 23 A1 92	no	no
IP	4 byte path	16.12.3.134	no	no
TCP	IP + port	16.12.3.134 / 3451	yes	yes
RPC	TCP + procedure	... / OpenFile	yes	yes
E-mail	host name + user	lampson@src.dec.com	no	yes

**Request-response is the most popular way to use messages**

It adds no concurrency (unless there are failures)

# Examples of Storage

<i>System</i>	<i>Address</i>	<i>Sample address</i>	<i>Data value</i>
Main memory	32-bit flat	04E72A39	1, 2, 4, or 8 bytes
File system	path name	/udir/bwl/Mail/inbox/214	0-4 Gbytes
World Wide Web	protocol + host name + path name	http:// src.dec.com/ SRC/docs.html	typed, variable size

# Universality

*In the Turing tarpit everything is possible, but nothing is easy.*

Alan Perlis

## **Can simulate messages with storage, and vice versa**

### **Messages → storage**

Send `load` and `store` messages to a ‘storage server’

### **Storage → messages**

Build message queues and poll them

Reinterpret `load` and `store` operations

# Performance

**Bandwidth**

**Latency**

**Connectivity**

**Predictability**

**Availability**

# Implementation Components

*An engineer can do for a dime what any fool can do for a dollar.*  
Anonymous

## Links

## Nodes

Converters

Multiplexers

Switches

## Effects of progress in silicon

$2 \times$  every 1.5–2 years

Implies more concurrency, more complexity

## Fiber optics for long-haul communication

25 THz/fiber is available

# Physical Links

<i>Medium</i>	<i>Link</i>	<i>Bandwidth</i>		<i>Latency</i>		<i>Width</i>
Alpha chip	on-chip bus	2.2	GB/s	3.6	ns	64
PC board	RAMbus	0.5	GB/s	150	ns	8
	PCI I/O bus	133.0	MB/s	250	ns	32
Wires	HIPPI	100	MB/s	100	ns	32
	SCSI	20	MB/s	500	ns	16
LAN	FDDI	12.5	MB/s	20 +	$\mu$ s	1
	Ethernet	1.25	MB/s	100 +	$\mu$ s	1
Wireless	WaveLAN	.25	MB/s	100 +	$\mu$ s	1
Fiber	OC-48	300	MB/s	5	$\mu$ s/km	1
Coax cable	T3	6	MB/s	5	$\mu$ s/km	1
Copper pair	IDSN	16	KB/s	5	$\mu$ s/km	1
Broadcast	CAP 16	3	MB/s	3	$\mu$ s/km	6 MHz

# Converter Nodes

**Internal converters are cheap**

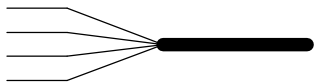
**Terminal converters are often messy**

Interface to communication standard

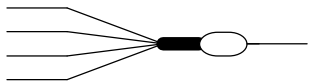
Adapter hardware

Driver software

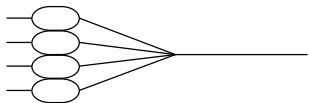
# Multiplexer Nodes



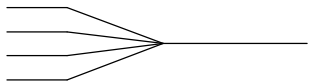
perfect (lossless) mux



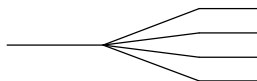
output buffered mux



input buffered mux



unbuffered mux



demux



broadcast

# Multiplexer Techniques

**Frequency division**

**Code division**

**Time division**

Fixed

Variable — address in each packet or cell

**Multiplexer does arbitration**

Scheduling, with flow control, and buffering

Usually centralized

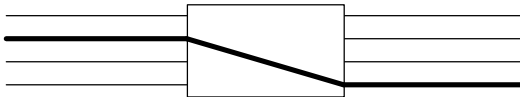
Collision, backoff, and retry

Usually distributed

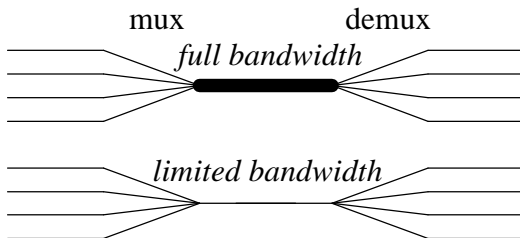
**Demultiplexer does addressing**

Centralized, or distributed by broadcast

# Switch Nodes

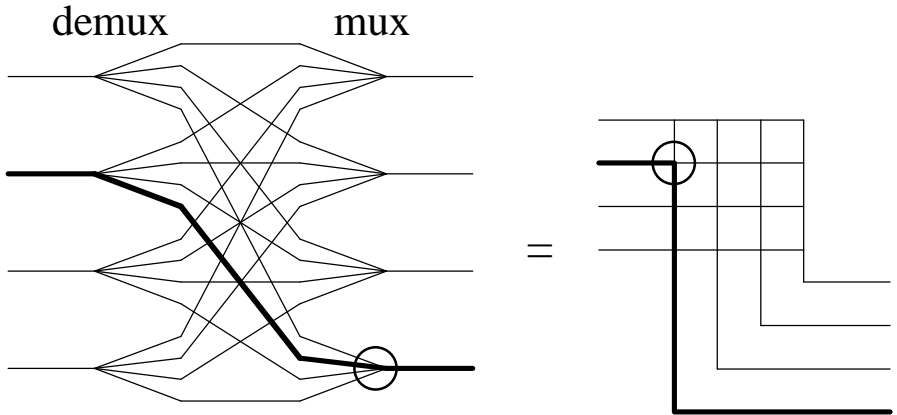


(a) The usual representation of a switch



(b) A mux–demux implementation

# Parallel Switch Nodes



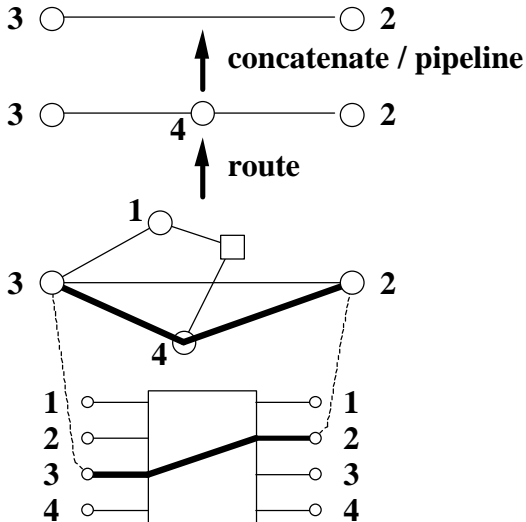
# Switch Examples

<i>Medium</i>	<i>Link</i>	<i>Bandwidth</i>	<i>Latency</i>	<i>Links</i>
Alpha chip	register file	13.2 GB/s	3.6 ns	6
Wires	Cray T3D	85 GB/s	1 $\mu$ s	2K
	HIPPI	1.6 GB/s	1 $\mu$ s	16
LAN	FDDI Gigaswitch	275 MB/s	10–400 $\mu$ s	22
	Switched Ethernet	10 MB/s	100–1200 $\mu$ s	8
Copper pair	Central office	80 MB/s	125 $\mu$ s	50K

# Recursion: Concatenating Links

*Any idea in computing is made better by being made recursive.*

Brian Randell



# Routing

Turn an address into a path through the mesh

Ways to do it

*used in*

Source routing

Multiprocessor grids

Cascaded I/O busses

Virtual circuits

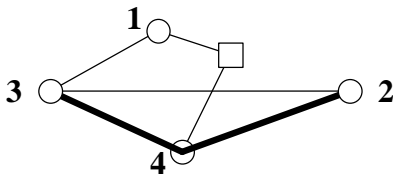
ATM

Hierarchical routing

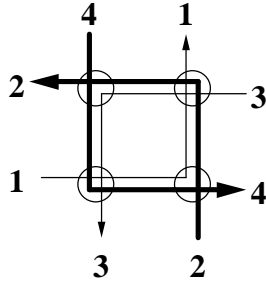
Internet

Flat routing

802 LANs



# Deadlock



**Must have an ordering on link resources.**

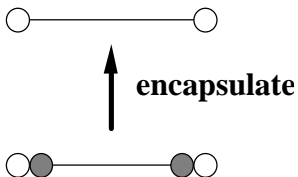
**On a grid, can use**

East or west first, then north or south

# Recursion: Layers

*There are three rules for writing a novel.  
Unfortunately, no one knows what they are.*

Somerset Maugham



# Layers under TCP

*What*

**a TCP reliable transport link**

on an Internet packet link

on PPP header compression protocol

on HDLC data link protocol

on a 14.4 Kbit/sec modem line

on an analog voice telephone line

on a 64 Kbit/sec line multiplexed

on a T1 line multiplexed

on a T3 line multiplexed

on an OC-48 fiber

*Why*

function: reliable stream

function: routing

performance: space

function: packet framing

function: byte stream

compatibility

function: bit stream

performance: aggregation

performance: aggregation

performance: aggregation

# Layers under E-mail

*What*

*Why*

mail folders

function:

organization

on a mail inbox

function:

storage

on SMTP mail transport

function:

routing

on FTP file transport

function:

char arrays

**on TCP reliable transport link**

• • •

# Layers under Memory

*What*

*Why*

load from cache

function: data access

miss to second level cache

performance: space

*or* miss to other processors

function: sharing

*or* reference on I/O bus

function: device access

miss to RAM

performance: space

*or* miss to network

function: sharing

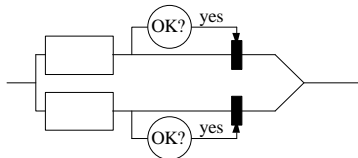
page fault to disk

performance: space

# Fault Tolerance

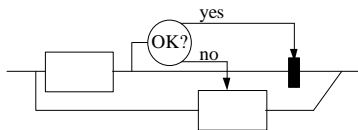
## Mirroring

- Duplicate components
- Detect errors
- Ignore bad components



## Retry: no state change on error

- Detect errors and retry



## Recovery

- Checkpoint
- Detect errors
- Crash
- Reconfigure without the bad components
- Retry from the checkpoint

## Failover to backup component

**What about timeouts? Fault tolerance requires real time.**

# Conclusions

## Uniform at many different scales

Storage and message interfaces

Links, converters, and switches

Compose by concatenation, routine, and layering

Bandwidth, latency, and connectivity

Congestion, flow control, and buffering

## Reasons

These are good ideas

$2 \times$  advances in silicon encourage similar designs