Networks

Distributed Systems

Philipp Kupferschmied

Universität Karlsruhe, System Architecture Group

May 6th, 2009

Why Communication?

- Threads of a distributed application collaborate
- Lack of shared memory $\Rightarrow$ messages

Preconditions:

- Physical interconnection network
  - Electric signals on wires, etc.
- Rules obeyed by each communication partner
  - Communication protocols
- Common language and common semantics
  - Otherwise no mutual understanding

Information, Data, Signals

information: abstract idea of what I want

data: formal representation of information using a known vocabulary

signal: physical representation of data in time/space on a medium

Information, Data, Signals

information: abstract idea of what I want

data: formal representation of information using a known vocabulary

signal: physical representation of data in time/space on a medium

Information, Data, Signals

information: abstract idea of what I want

data: formal representation of information using a known vocabulary

"tree"

signal: physical representation of data in time/space on a medium

Information, Data, Signals

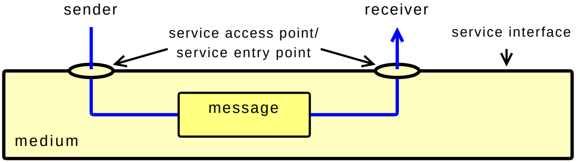
information: abstract idea of what I want

data: formal representation of information using a known vocabulary

"tree"

signal: physical representation of data in time/space on a medium

Base Model of Communication



- Participants act as sender (client) or receiver (server)
- Using a service by a client takes place via a service-interface on a specific service entry point
- Via the medium the spatial distance is met

Medium—The Physical Essence in Networks

- Signals need a medium
  - Air for sound waves
  - Paper for written letters
  - Copper wire for electrical signals
  - Fibre for optical signals
  - "The ether" for wireless transmission
- Some media relay the signals to many receivers, others don't
  - Broadcast, shared medium
  - Point-to-point communication

What May Happen When Communicating

- Identify the communication partner
- Need to find the communication partner
- Get a communication line (with certain properties)
- Make sure we talk "the same language"
- Come to a common understanding what we agreed upon
- Deal with partial or complete network breakdown
- (more)

Reliable vs. Unreliable Communication

- Reliable Communication
  - All messages are delivered either exactly once
  - ... or at least once
  - ... or at most once
- Unreliable Communication
  - Each message may be delivered once
  - ... or multiple times
  - ... or not at all

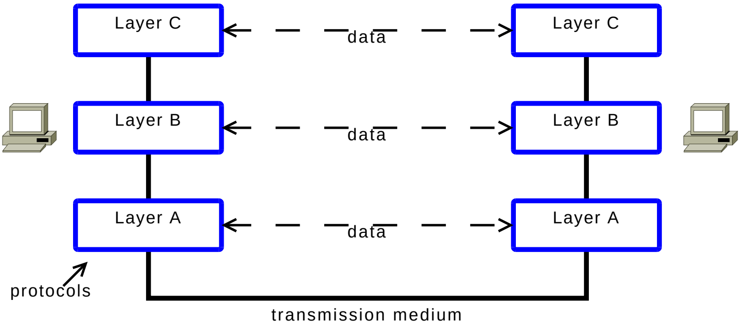
Uni- vs. Bidirectional Communication

- Notification
  - One way, unidirectional
  - Sender continues immediately
  - Usually ‘unreliable’ (may get lost)
  - E.g., remote log messages
- Request
  - Bidirectional
  - Sender waits for response
    - ACK or
    - Some result
  - Usually ‘reliable’ required
  - E.g., RPC

Point-to-Point vs. End-to-End

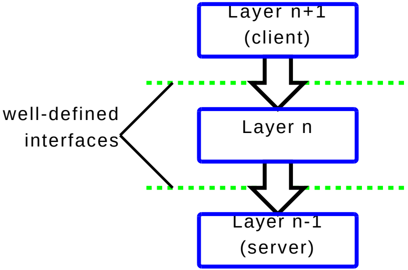
- Point-to-point
  - Data exchange between directly physically connected hosts
  - \+ No routing required
  - − Completely meshed network required
- End-to-end
  - Data exchange between between potentially non-neighbored hosts
  - − Routing required (depending on topology)
  - \+ Less expensive network topology possible

Layered Communication



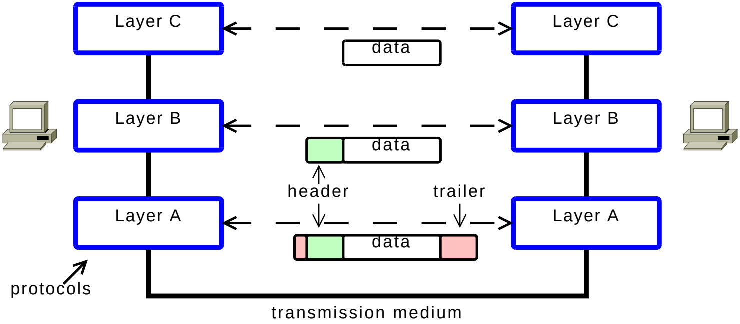
- Seemingly: horizontal communication within layers
- Actually: vertical communication between layers per system

Protocols and Interfaces



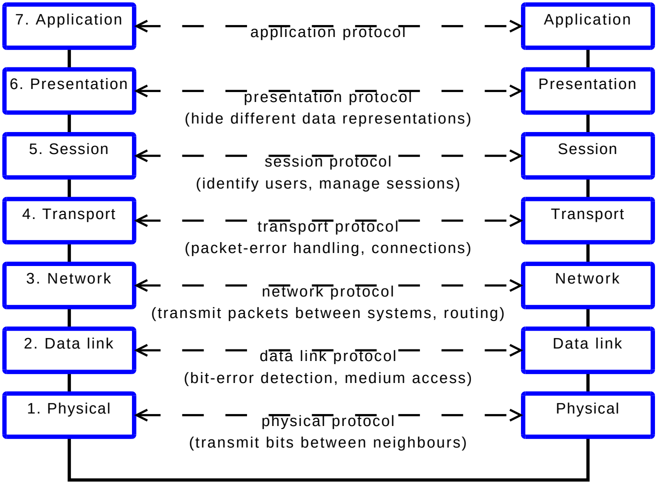
- Two adjacent layers in a protocol stack work together
  - Server offers service
  - Client wants service
- Interface must be specified

Headers and Encapsulation

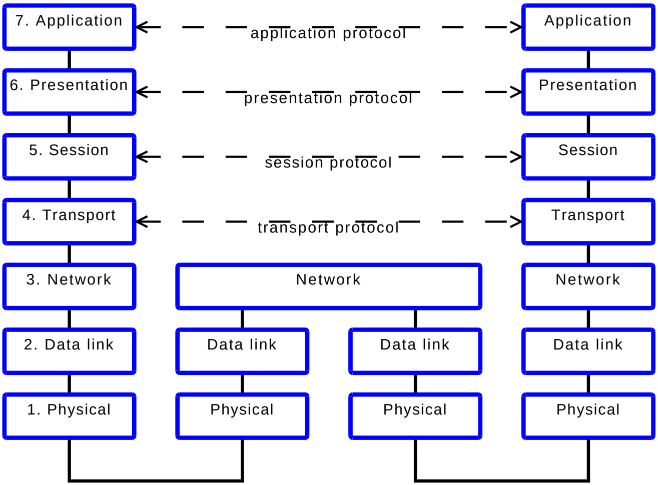


- Each layer wraps data from previous layer
- Type of contents
  - Explicitly given in outer protocol header
  - Implicitly inferred from received 'data'

The ISO/OSI Reference Model



The ISO/OSI Reference Model



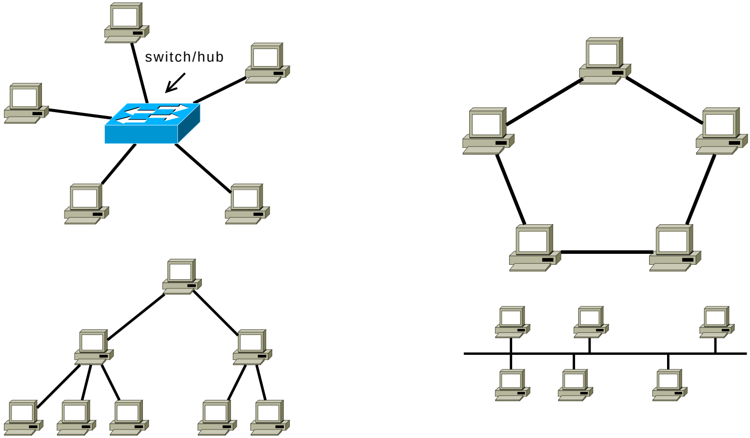
Role of ISO/OSI Layers 1–4

1. Physical layer (e.g., RS-232, Manchestercode)
   - Unreliable point-to-point communication
   - Carry signals between neighbors
2. Data link layer (e.g., Ethernet)
   - 'Reliable' point-to-point communication
   - Medium access control (MAC), bit error detection
3. Network layer (e.g., IP)
   - Unreliable end-to-end communication
   - Routing
4. Transport layer (e.g., TCP)
   - Reliable end-to-end communication
   - Detect packet loss, out-of-order reception

Role of ISO/OSI Layers 5–7

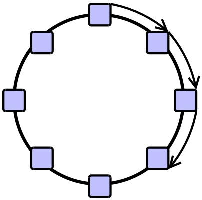
5. Session layer
    - Manage sessions
    - Used for example by RPC
6. Presentation layer
    - Convert data formats
        - Little-/big-endian
        - Relative/absolute data
        - Textual vs. numeric representation
    - Mostly unused/integrated in application layer
7. Application layer (e.g., HTTP, FTP)
    - Whatever you want

Overview



- Communication networks: usually star, bus, ring, . . .
- Multicomputer/DS: usually ring, lattice, hypercube, . . .

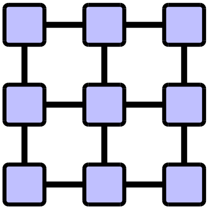
Token Ring

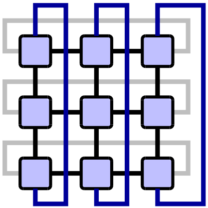


- Constant, low node degree (always 2)
- (Practically) limited extensibility

- Sender puts message on line
  - If he received the token message
- All nodes inspect message and pass it on
- Original sender does not pass the message on
  - Converts message into token message
  - Medium access controlled by token

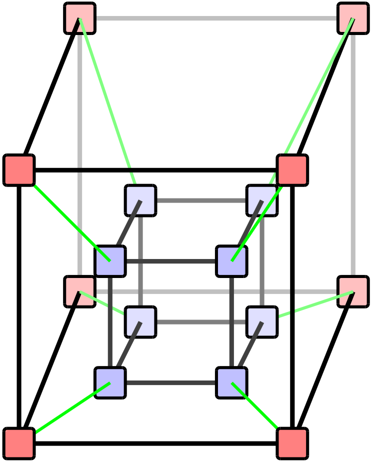
Lattice/Torus





- Torus $\sim$ generalized ring
- Constant node degree ($2\times$ #dimensions)
- Incrementally extensible in (small) steps
- Good for distributed apps. with 'local' communication pattern (e.g., meterologic simulations)

Hypercube

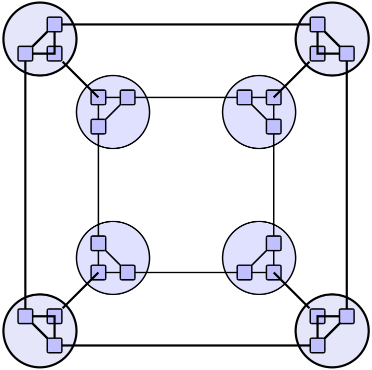


- Logarithmic diameter (max. distance between nodes)
- Extensible by doubling number of nodes only
  - Take two *d*-dimensional hypercubes
  - Combine corresponding nodes with edges
  - Also increases node degree (#dimensions)

Analysis Hypercube

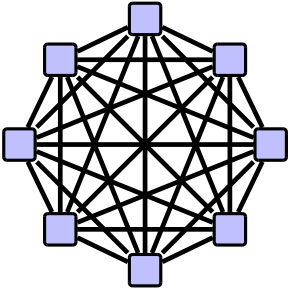
- $N = 2^d$ nodes
- d neighbours per node (degree d)
- $d \times 2^d$ edges
- Diameter: $\log_2 N = \log_2 2^d = d$
- Simple routing
- Many paths (robust, parallel communication)
- Expensive

Cube Connected Cycles



- $d \times 2^d$ nodes (d: dimension)
- Logarithmic diameter
- Constant, low node degree (always 3)
- Extensible by (more than) doubling number of nodes only

Completely Meshed



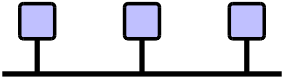
+ Low diameter (1)
– High cost
– Limited extensibliity by max. node degree

Summary

- Lattice $L(a_1 \star a_2 \star \ldots \star a_d)$
- Torus $T(a_1 \star a_2 \star \ldots \star a_d)$
- Hypercube $H(d)$
- Cube Connected Cycles $CCC(d)$

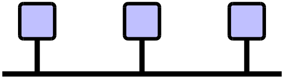
	nodes	edges	degree	diameter
Lattice	$\prod_{k=1}^{d} a_k$	$\sum_{k=1}^{d} \prod_{i \neq k} a_i(a_k - 1)$	$\leq 2d$	$\sum_{k=1}^{d} (a_k - 1)$
Torus	$\prod_{k=1}^{d} a_k$	$d \prod_{k=1} a_k$	$= 2d$	$\sum_{k=1}^{d} \lfloor a_k/2 \rfloor$
$H(d)$	2^d	$d \times 2^{d-1}$	$= d$	d
$CCC(d)$	$d \times 2^d$	$3d \times 2^{d-1}$	$= 3$	$2d + \lfloor d/2 \rfloor$

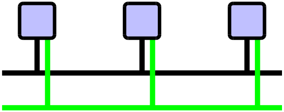
Bus-Connected Nodes



- Cheap
- Blocking
- Poor scalability
- E.g., PCI, SCSI, Ethernet

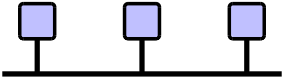
Bus-Connected Nodes

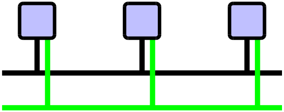


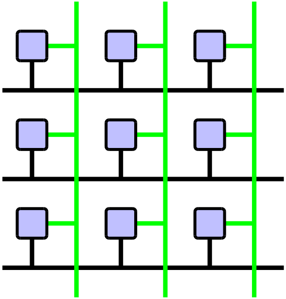


- Cheap
- Blocking
- Poor scalability
- E.g., PCI, SCSI, Ethernet

Bus-Connected Nodes

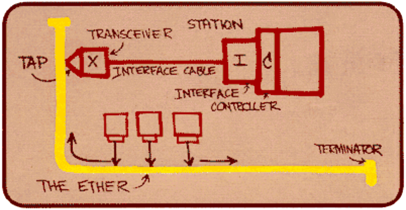






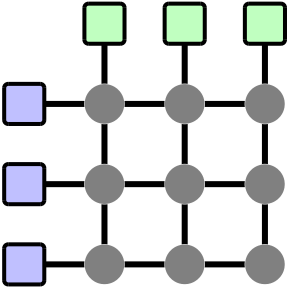
- Cheap
- Blocking
- Poor scalability
- E.g., PCI, SCSI, Ethernet

Example: Ethernet



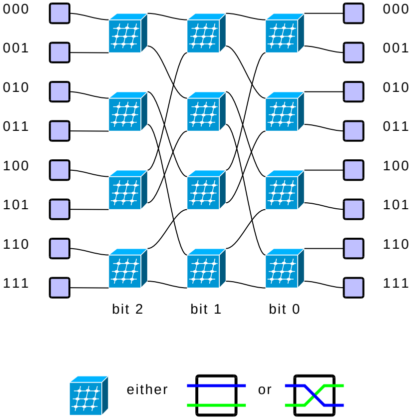
- Developed 1973–1976 at Xerox
- Physical bus, hubs and switches
- Medium access: CSMA/CD
- 2.94 MBit/s (today: 10 GBit/s)

Crossbar



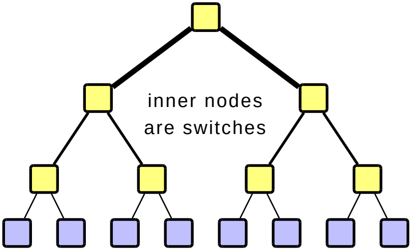
- \+ Non-blocking
- \+ Fast
- − Expensive
- − Not scalable

Omega Nets



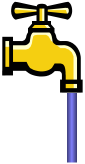
- Hierarchical crossbar
+ Less expensive
− Higher latency
− Limited connectivity
  - 000 → 000 precludes 100 → 001
- Simple routing src → dst
  - interpret dst bitwise, MSB first
  - 0: upper port, 1: lower port

Fat Tree Nets



- Upper levels with higher bandwidths or multiple redundant links

Circuit vs. Packet Switching



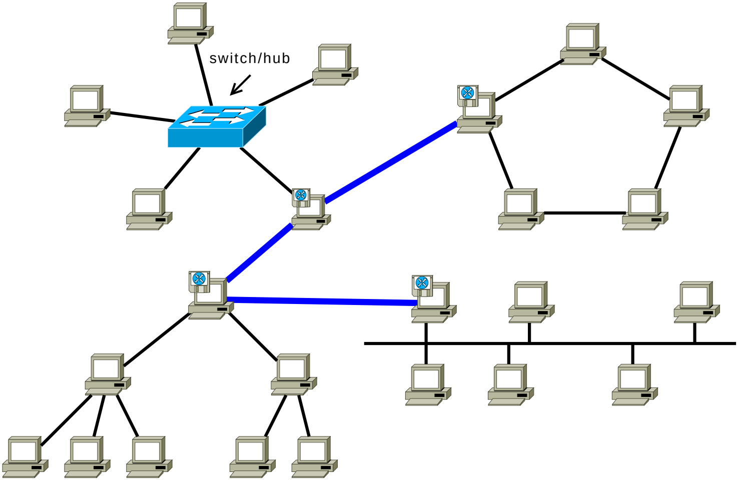
Circuit switching

- Guarantees bandwidth
- Connection established if all systems can fulfill requirements
- State in intermediate systems represents the connection
- E.g., telephone, desired for streaming

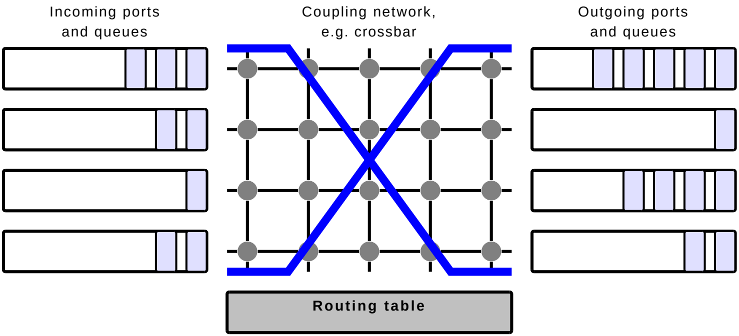
Packet switched

- Gains bandwidth by multiplexing
- Packets are treated independently
- E.g., Internet, suited for message-oriented communication

Connecting LANs

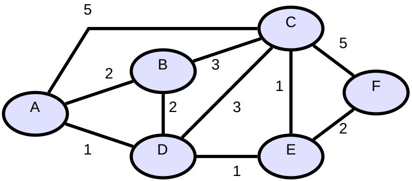


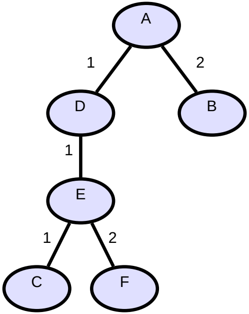
Schematic Outline of a Router



- Packets are queued in each router
  - Increased lantency
  - Packets are dropped if incoming queue is full

Link-State Routing (Dijkstra)





- Convert graph to tree
  - Root = node whose routing table is desired
  - Successively add node with smallest accumulated distance

– Requires whole system image ⇒ not scalable

Distance Vector Routing

- Simple adaptive routing protocol
- Originally used as 'RIP' for the Internet (now: BGP)
- Every router maintains a table
  - (node, out-port, distance) per node
- Tables are regularly exchanged with neighbours
  - If (distance+1) $<$ my distance, update out-port and distance
- Problems
  - Slow convergence to consistent state
  - Count-to-infinity for broken nodes
  - High overhead due to exchange of complete tables

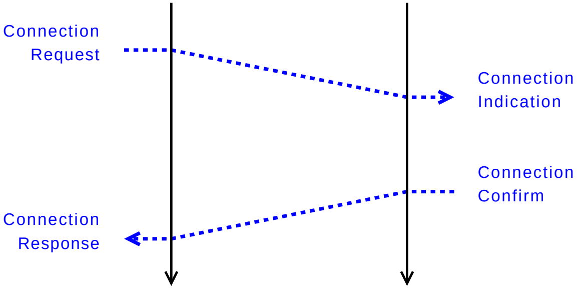
Purpose

- Use unreliable, packet switched network
- Provide reliable transmission of data streams
  - Introduce ACKs and sequence numbers
  - Retransmit unACKed packets after timeout
- Multiplex multiple connections on a link
- Flow control
- Congestion control
- E.g., TCP
  - UDP (more or less) offers IP at transport layer

Notions

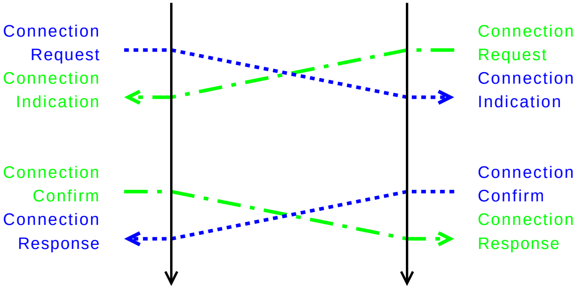
- Acknowledgements
  - Positive: OK, packet received (ACK)
  - Negative: Packet missing, resend! (NACK)
- Sequence numbers
  - Allow correct reassembly of data
    - Packets may come in out-of-order
    - Packets may be lost
- Timeouts
  - Determine whether a packet should be resend
  - E.g., if no ACK was received after 3 s

Establishing a Connection



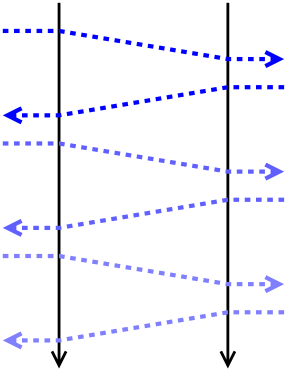
- Are connections bidirectional?

Establishing a Connection



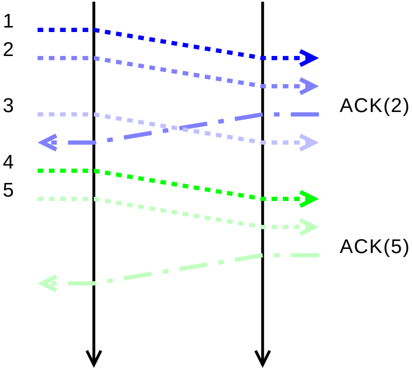
- Are connections bidirectional?
- Does this open one or two connections?

Stop and Wait



- Sender sends one packet and waits for ACK
  - Resend after timeout
  - If ACK was lost, receiver gets duplicates!
  - ⇒ Use sequence numbers to identify duplicates
- – Slow due to waiting for ACKs

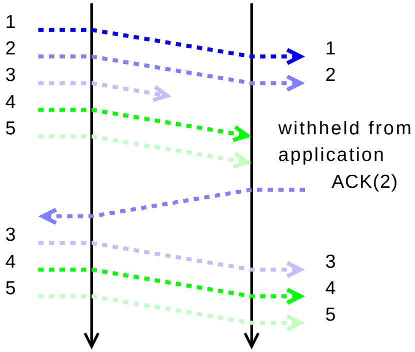
Sliding Window



- Send consecutive packets i to $i + W - 1$
- Send further packets only after i has been ACKed

\+ Increased throughput due to
  - Multiple outstanding packets
  - Aggregated ACKs
  - Possibly overlapping data and ACK packet transmission

Retransmission



- A sends 5 packets
- Packet 3 gets lost
- Packets 4 and 5 are withheld from the application at B
- B ACKs reception of all packets ≤ 2 (or NACKs packet 3)
- A resends 3–5 (or just 3)

+ Complete data is passed in-order to the application

− Sender must keep sent packets

The End

Feel free to ask questions!