8. Networks

- Why networked embedded systems
- General network architecture
  - ISO seven network layers
- Networks
  - I^2C, CAN, Ethernet
- Internet-enabled embedded systems
- Sensor networks

Distributed embedded systems

- Distributed embedded systems
  - Processing elements (PEs) are connected through a network
- The application can be distributed over the PEs
  - Clients and servers
- Network
  - Internet
  - CAN

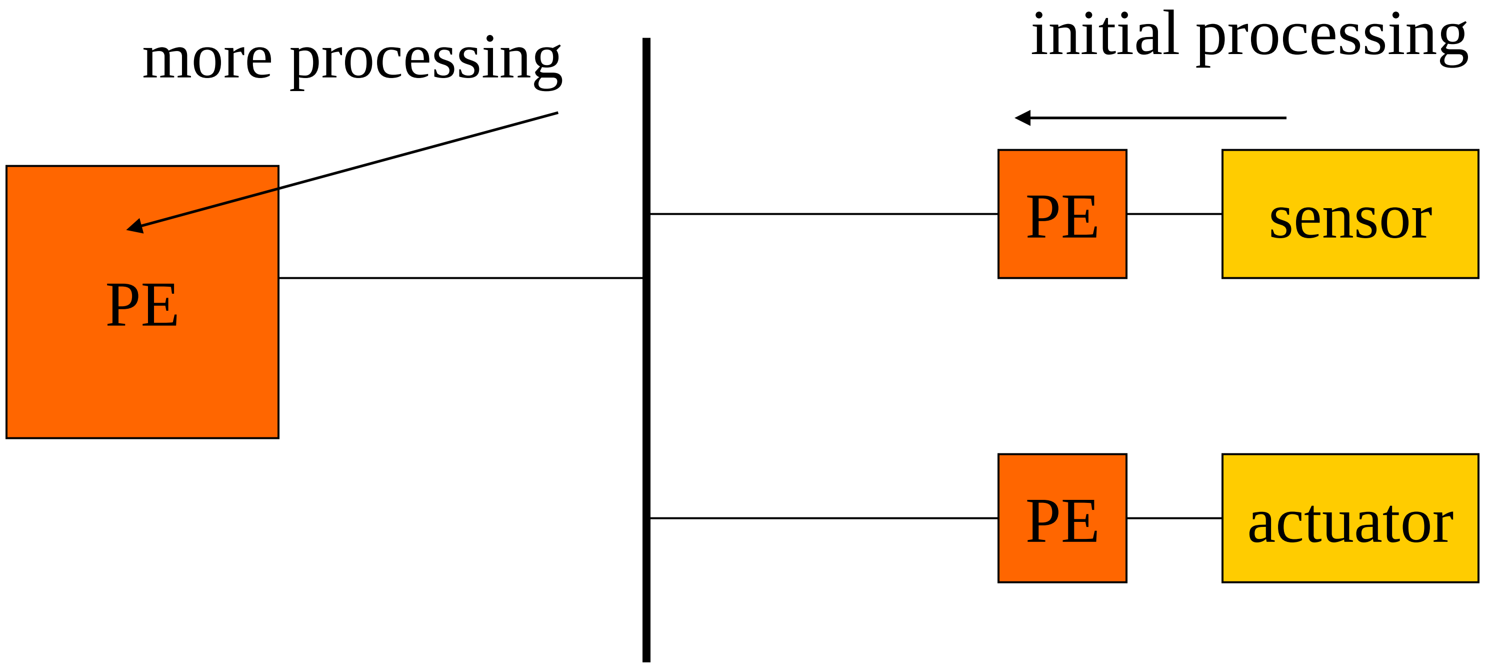
Network-based ES

- Physically distributed activities---time constants may not allow transmission to central site.
  - Engine control: short time delays required for the task
- Data reduction is another important reason for distributed processing.
- Modularity is another motivation for network-based design.
  - A large system is assembled out of existing components.

Network-based ES

- Improved debugging---use one CPU in network to debug others.
- In some cases, networks are used to build fault tolerance into systems.
- Distributed embedded system design is a good example of hardware/software co-design since we must design the network topology as well as the software running on the network node.

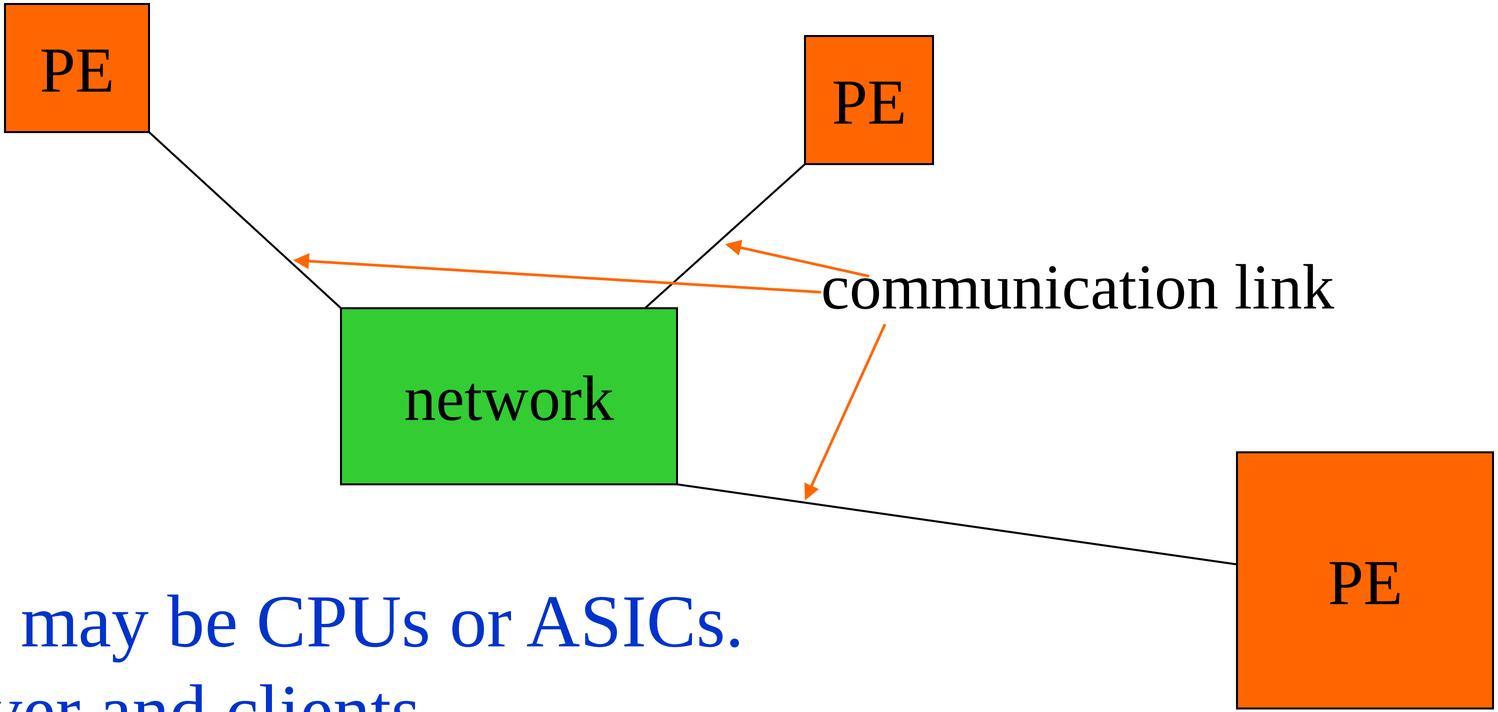
Sensor-actuator networks



It is necessary to put some PEs near where the events occur.

Network elements

distributed computing platform:



PEs may be CPUs or ASICs.
Server and clients

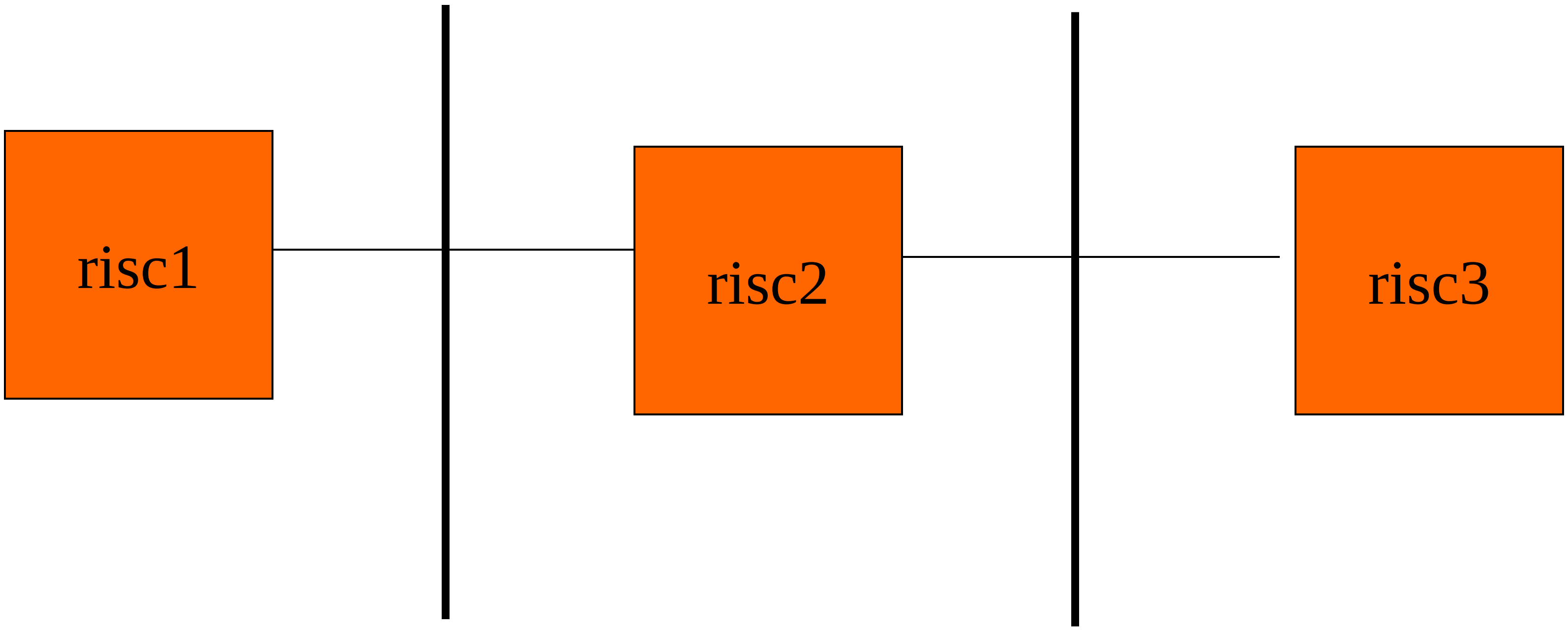
Distributed ES

- Unlike the system bus, the distributed embedded system does not have memory on the network (or bus)
- PEs do not fetch instruction over the network as they do on the microprocessor bus.
  - speed of PE and latency of network

Why distributed?

- Network-based embedded system
  - More complicated, then why?
- Physically separated & deadline is short
  - Rather than building high-speed network to carry the data to a distant, fast PE.
  - PE near to the location data colloected
- Advantage of a distributed system with several CPU
  - A part of the system can be used to debug another part.

Debugging a multi-core system



To diagnose risc 2, we can use risc1 to generate inputs and risc3 to watch output.

Network abstractions

- Networks are complex
- Ideally, they provide high-level services while hiding details of data transmission
- International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model to help us to understand networks:
  - 7-layer model.
- Provides a standard way to classify network components and operations.

OSI reference model

7. application	end-use interface
6. presentation	data format
5. session	application dialog control
4. transport	connections
3. network	end-to-end service
2. data link	reliable data transport
1. physical	mechanical, electrical

OSI layers

- Physical: connectors, bit formats, etc.
- Data link: error detection and control across a single link (single hop).
- Network: end-to-end multi-hop data communication.
- Transport: provides connections; may optimize network resources.
- Session: services for end-user applications: data grouping, checkpointing, etc.
- Presentation: data formats, transformation services.
- Application: interface between network and end-user programs.

LAN - Local Area Network

- connects computers that are physically close together (< 1 mile).
  - high speed
  - multi-access
- Technologies:
  - Ethernet 10 Mbps, 100Mbps, 1Gbps
  - Token Ring 16 Mbps
  - FDDI 100 Mbps

WAN - Wide Area Network

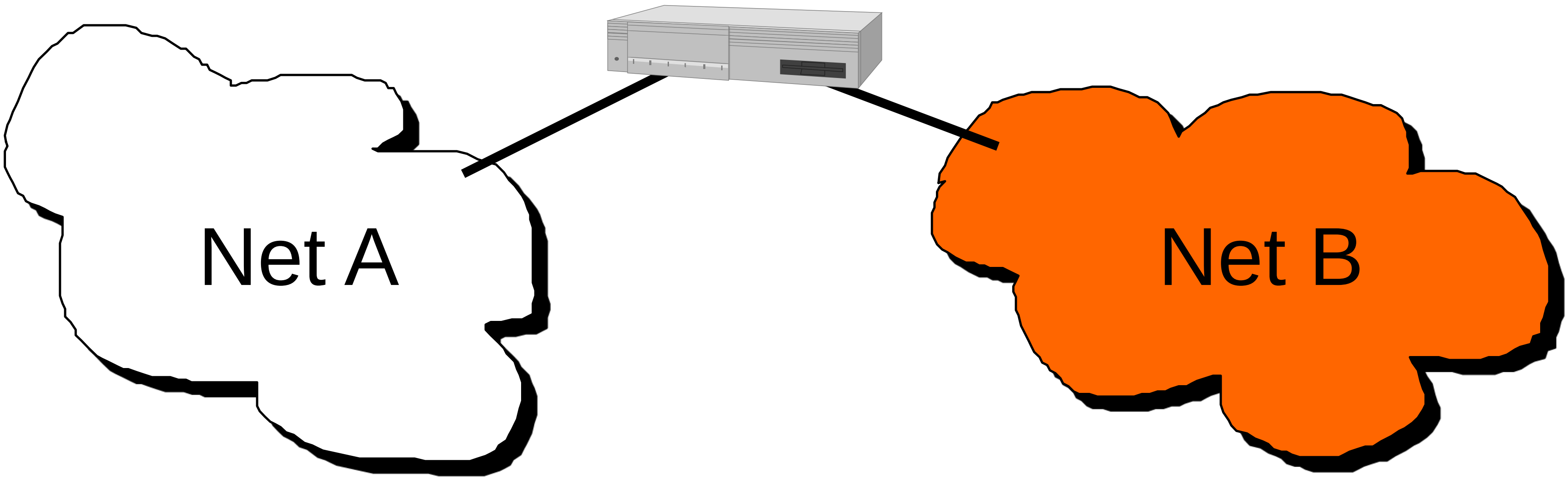
- connects computers that are physically far apart. “long-haul network”.
  - typically slower than a LAN.
  - typically less reliable than a LAN.
  - point-to-point
- Technologies:
  - telephone lines
  - Satellite communications

MAN - Metropolitan Area Network

- Larger than a LAN and smaller than a WAN
  - example: campus-wide network
  - multi-access network
- Technologies:
  - coaxial cable
  - microwave

Internetwork

- Connection of 2 or more distinct (possibly dissimilar) networks.
- Requires some kind of network device to facilitate the connection.

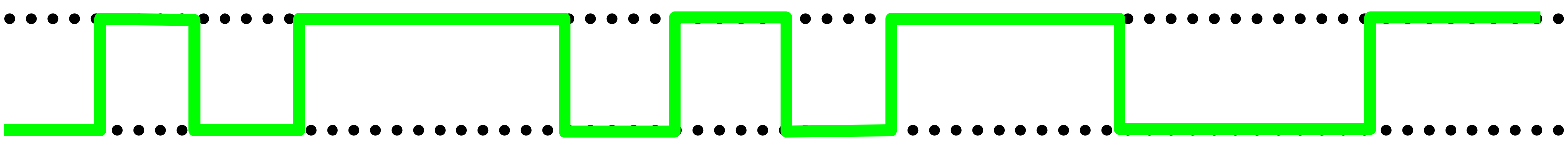


1. Physical Layer

- deals with physical characteristics of the transmission medium.
- defines electrical, mechanical, procedural, and functional specifications for activating, maintaining, and deactivating the physical link
- defines such characteristics as voltage levels, timing of voltage changes, physical data rates, maximum transmission distances, physical connectors, and other similar attributes
- Examples : EIA/TIA-232, RJ45, NRZ.

1. Physical Layer

- Responsibility:
  - transmission of raw bits over a communication channel.
- Issues:
  - mechanical and electrical interfaces
  - time per bit
  - distances



2. Data Link Layer

- provides access to the media and physical transmission across the media, which enables the data to locate its intended destination
- provides reliable transit of data across a physical link by using the MAC addresses, which define a hardware or data link address in order for multiple stations to share the same medium and still uniquely identify each other.
- Concerned with network topology, network access, error notification, ordered delivery of frames, and flow control.
- Examples: Ethernet, Frame Relay, FDDI.

2. Data Link Layer

- Two sublayers
  - Data link control, which provide an error-free communication link
    - *framing* (dividing data into chunks)
      - header & trailer bits
    - addressing
  - MAC, which is needed by multiaccess networks.
    - MAC provides DLC with "virtual wires" on multiaccess networks.

3. Network Layer

- defines
  - end-to-end delivery of packets.
  - logical addressing so that any endpoint can be identified.
  - how routing works and how routes are learned so that the packets can be delivered.
  - how to fragment a packet into smaller packets to accommodate different media.
- Routers operate at Layer 3.

3. Network Layer

- Examples: IP, IPX, AppleTalk.
- Responsibilities:
  - path selection between end-systems (routing).
  - subnet flow control.
  - fragmentation & reassembly
  - translation between different network types.
- Issues:
  - *packet* headers
  - virtual circuits

4. Transport Layer

- regulates information flow to ensure end-to-end connectivity between host applications reliably and accurately.
- segments data from the sender and reassembles the data into a data stream on the receiver.
- the lower four layers are concerned with data transport issues.

4. Transport Layer

- Layer 4 protocols include
  - TCP (Transmission Control Protocol) and
  - UDP (User Datagram Protocol).
- Responsibilities:
  - provides virtual end-to-end links between peer processes.
  - end-to-end flow control
- Issues:
  - headers
  - error detection
  - reliable communication

5. Session Layer

- defines how to start, control and end sessions between applications.
- includes the control and management of multiple bi-directional messages using dialogue control.
- synchronizes dialogue between two hosts' presentation layers and manages their data exchange.
- The upper three layers (the application, presentation, and session layers) are concerned with application issues.

5. Session Layer

- The session layer offers provisions for efficient data transfer.
- Examples: SQL, ASP(AppleTalk Session Protocol).
- Responsibilities:
  - establishes, manages, and terminates sessions between applications.
  - service location lookup
- Many protocol suites do not include a session layer.

6. Presentation Layer

- The information that the application layer of one system sends out is ensured to be read by the application layer of another system.
- If necessary, translates between multiple data formats by using a common format.
- provides encryption and compression of data.

6. Presentation Layer

- Examples: JPEG, MPEG, ASCII, EBCDIC, HTML.
- Responsibilities:
  - data encryption
  - data compression
  - data conversion
- Many protocol suites do not include a Presentation Layer.

7. Application Layer

- provides network services to the user’s applications.
- does not provide services to any other OSI layer, but rather, only to applications outside the OSI model.
- Examples of such applications: spreadsheet programs, word processing programs, and bank terminal programs.

7. Application Layer

- establishes the availability of intended communication partners, synchronizes and establishes agreement on procedures for error recovery and control of data integrity.
- Responsibilities:
  - anything not provided by any of the other layers
- Issues:
  - application level protocols
  - appropriate selection of “type of service”

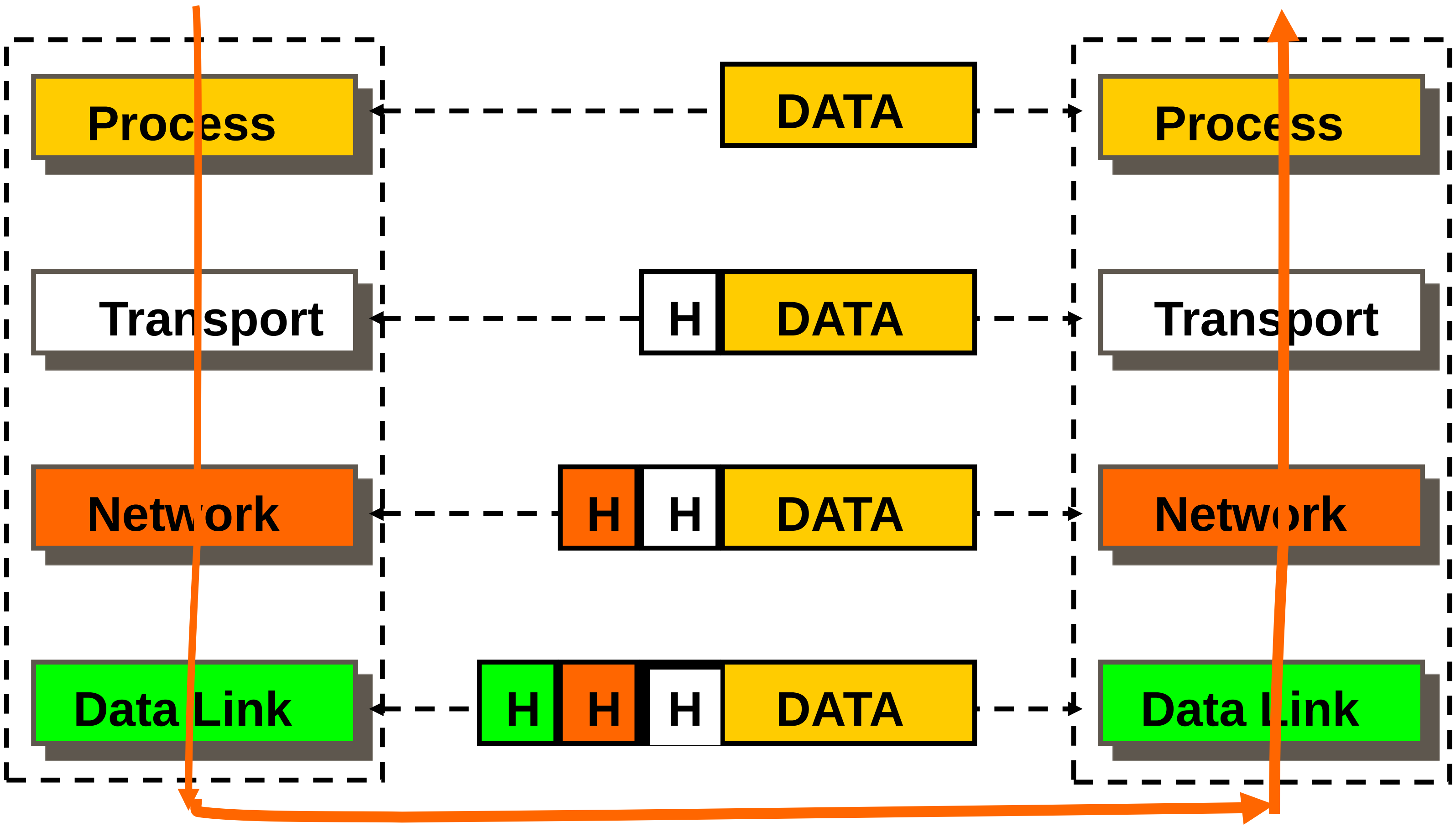
TCP/IP protocols

- Application layer (telnet, ssh, http, ftp, etc)
  - Main functionality: run application protocols
- Transport layer (TCP, UDP)
  - Main functionality: reliability
- Network layer (IPv4, IPv6)
  - Main functionality: routing/fragmentation/internetworking
- Host to data link layer (Ethernet)
  - Main functionality: medium access, encoding

Layering & Headers

- Each layer needs to add some control information to the data in order to do its job.
- This information is typically appended to the data before being given to the lower layer.
- Once the lower layers deliver the the data and control information - the peer layer uses the control information.

Headers



Process
DATA
Process
Transport
H
DATA
Transport
Network
H
H
DATA
Network
Data Link
H
H
H
DATA
Data Link

Important Summary

- ⌘ Data-Link: communication between machines on the same network.
- ⌘ Network: communication between machines on possibly different networks.
- ⌘ Transport: communication between processes (running on machines on possibly different networks).

Connecting Networks

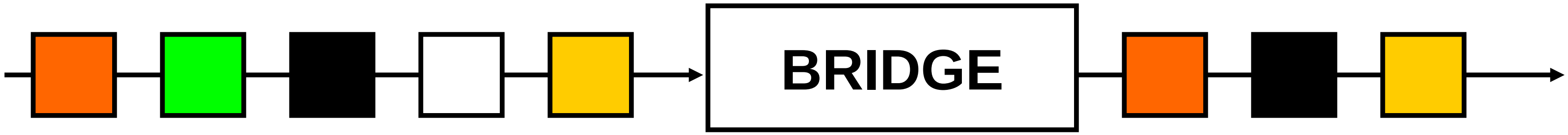
- Repeater: physical layer
- Bridge: data link layer
- Router: network layer
- Gateway: network layer and above.

Repeater in the physical layer

- Copies bits from one network to another
- Does not look at any bits
- Allows the extension of a network beyond physical length limitations

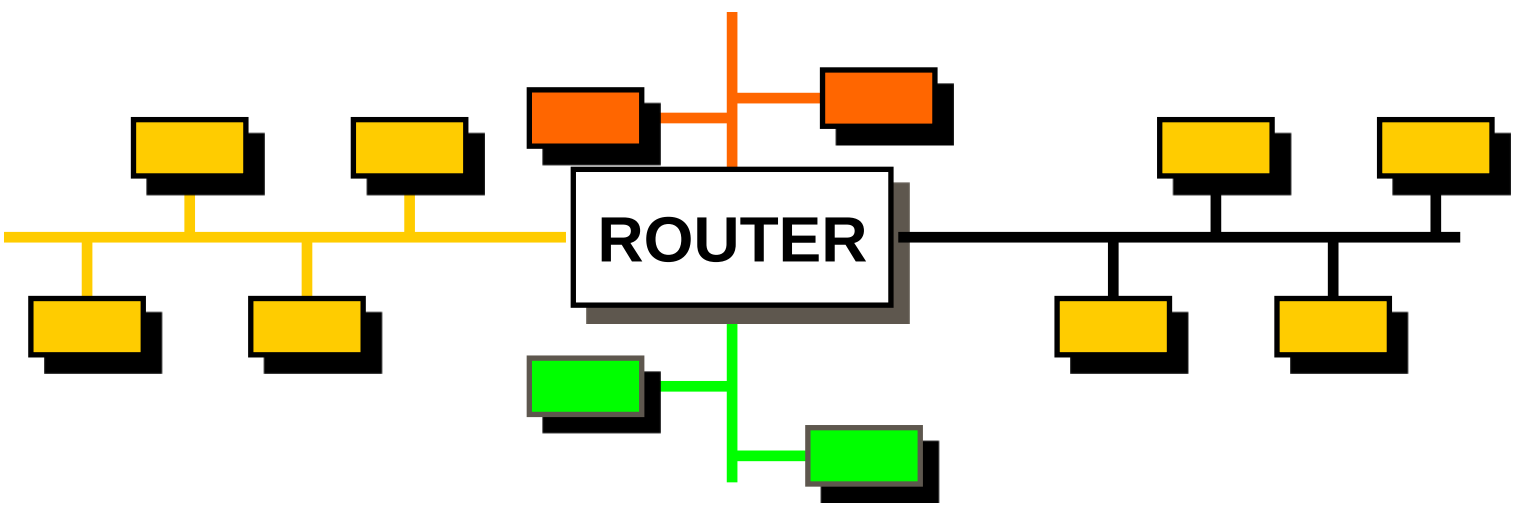
Bridge in the data link layer

- Copies frames from one network to another
- Can operate selectively - does not copy all frames (must look at data-link headers).
- Extends the network beyond physical length limitations.



Router in the network layer

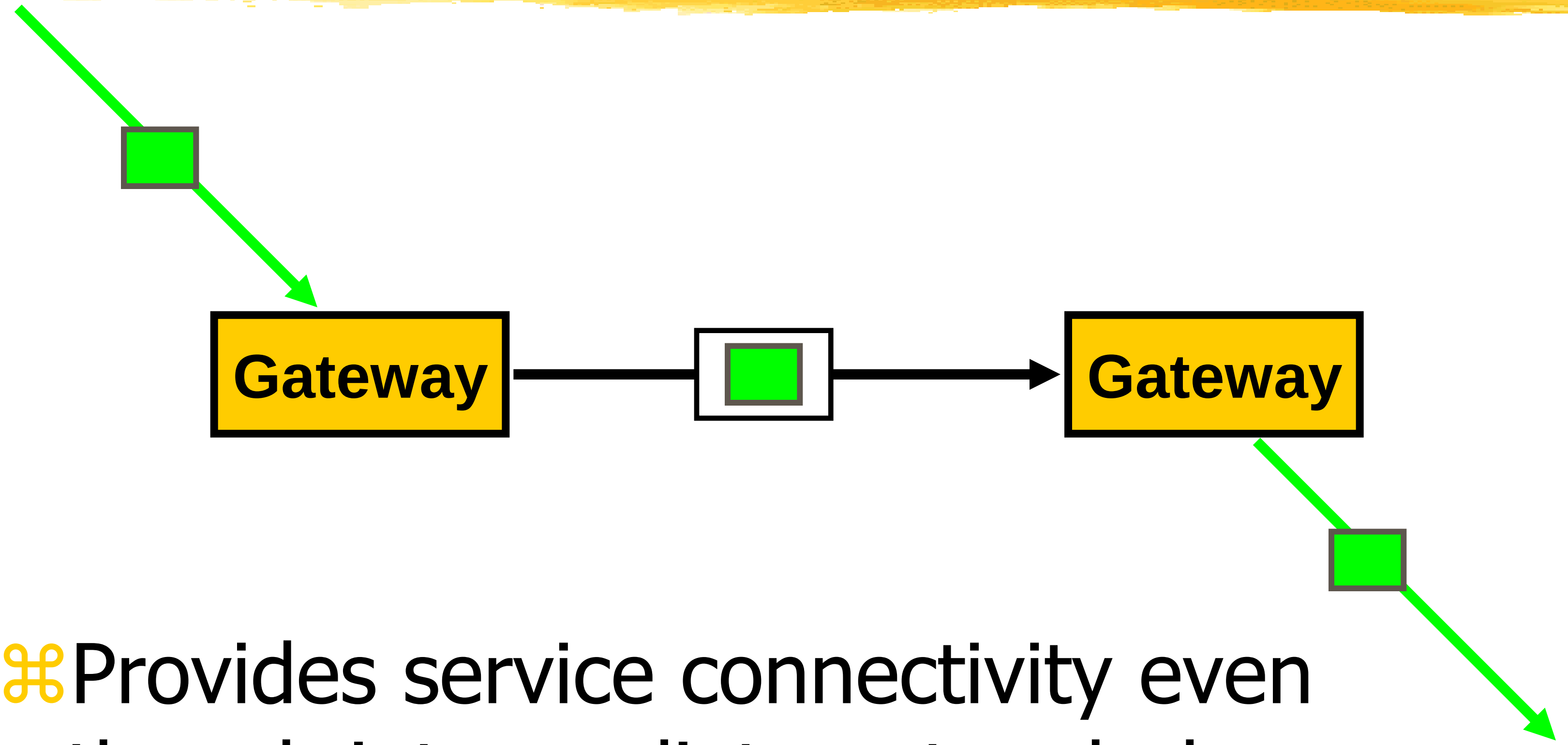
- Copies packets from one network to another.
- Makes decisions about what *route* a packet should take (looks at network headers).



Gateway

- Operates as a router
- Data conversions above the network layer.
- Conversions:
  - encapsulation - use an intermediate network
  - translation - connect different application protocols
  - encryption - could be done by a gateway

Encapsulation

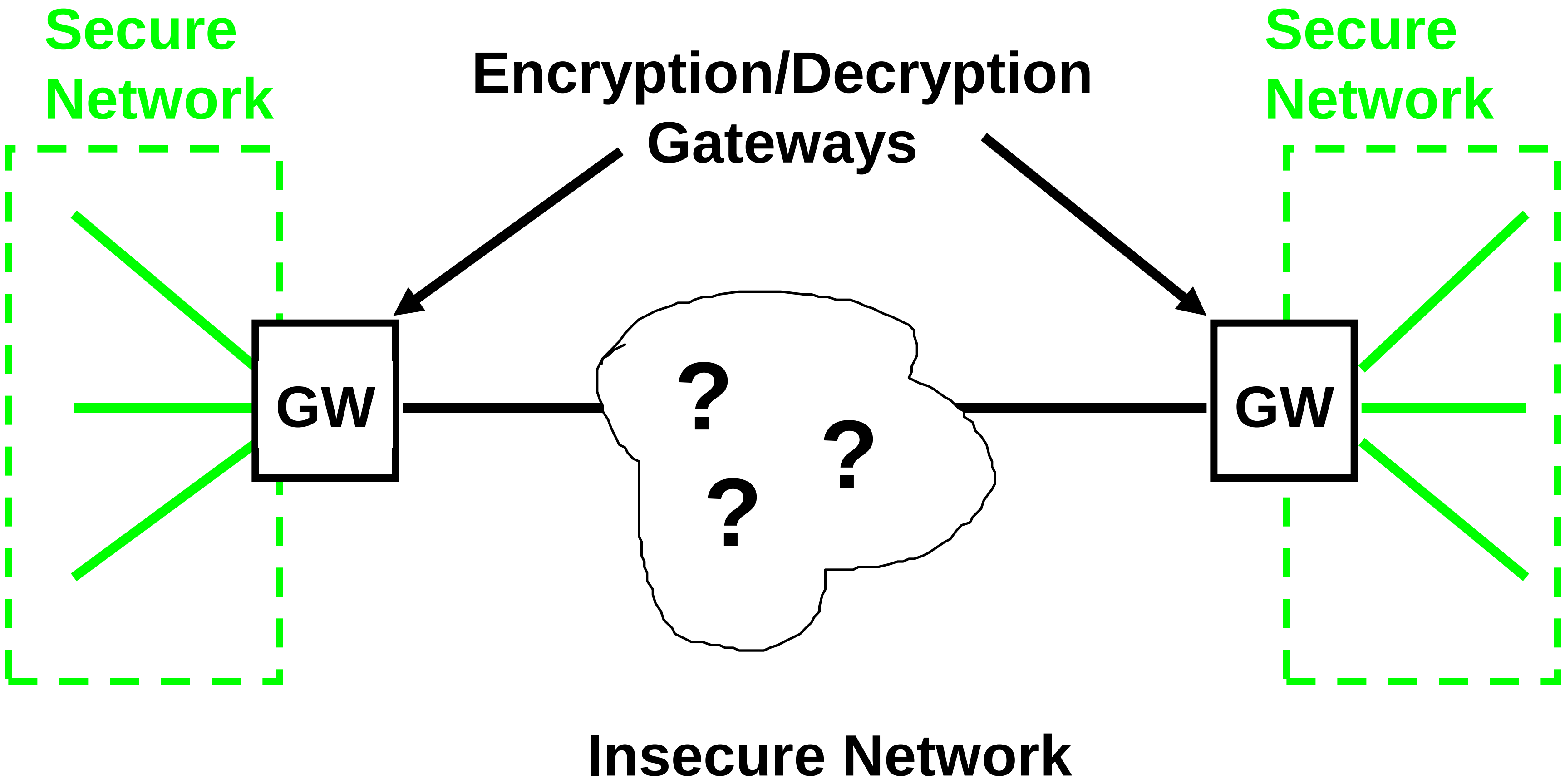


- Provides service connectivity even though intermediate network does not support protocols.

Translation

- Translate from green protocol to brown protocol

Encryption gateway



Hardware vs. Software

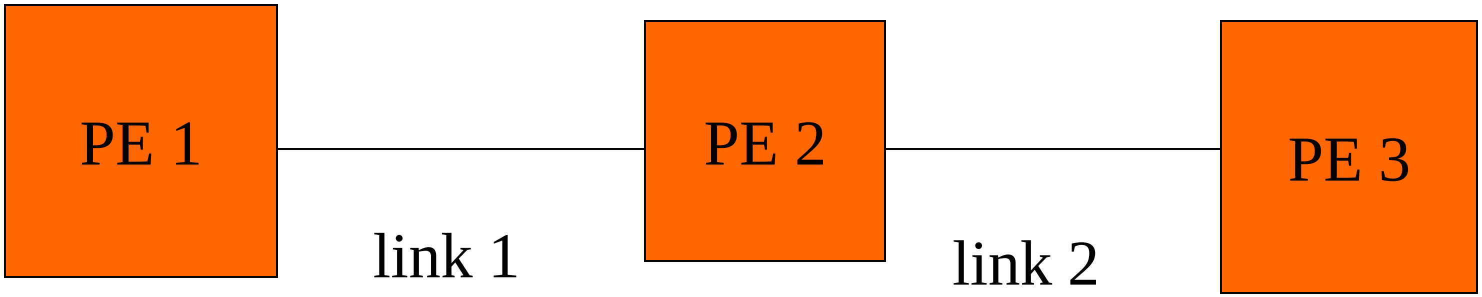
- Repeaters are typically hardware devices.
- Bridges can be implemented in hardware or software.
- Routers & Gateways are typically implemented in software so that they can be extended to handle new protocols.
- Many workstations can operate as routers or gateways.

Hardware architectures

- Many different types of networks:
  - topology;
  - scheduling of communication;
  - routing.

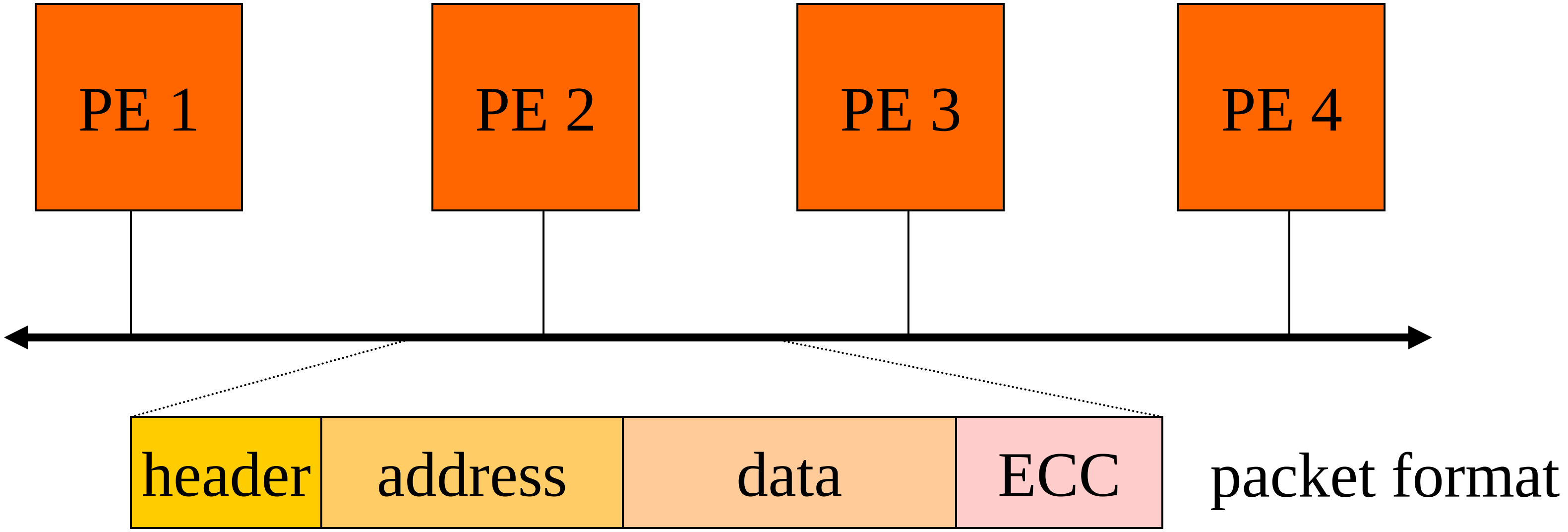
Point-to-point networks

- One source, one or more destinations, no data switching (serial port):



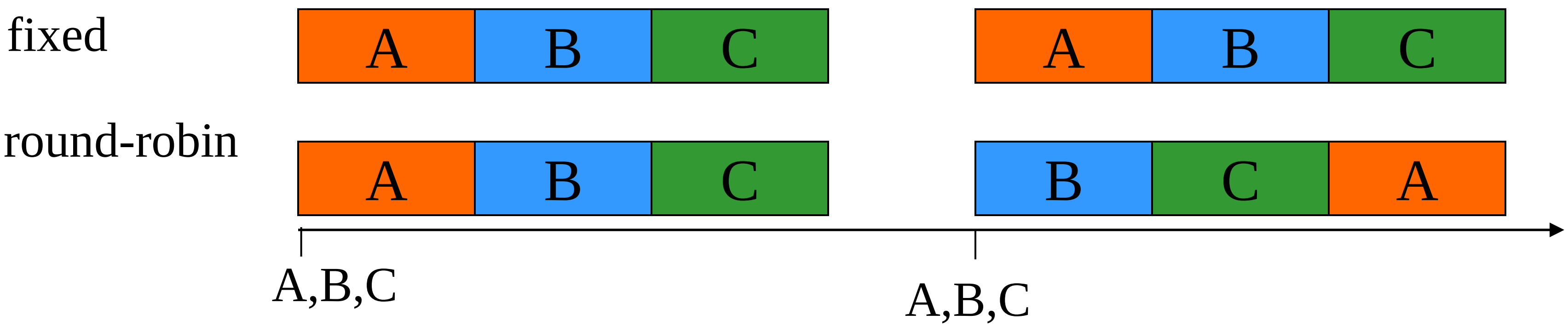
Bus networks

- Common physical connection:

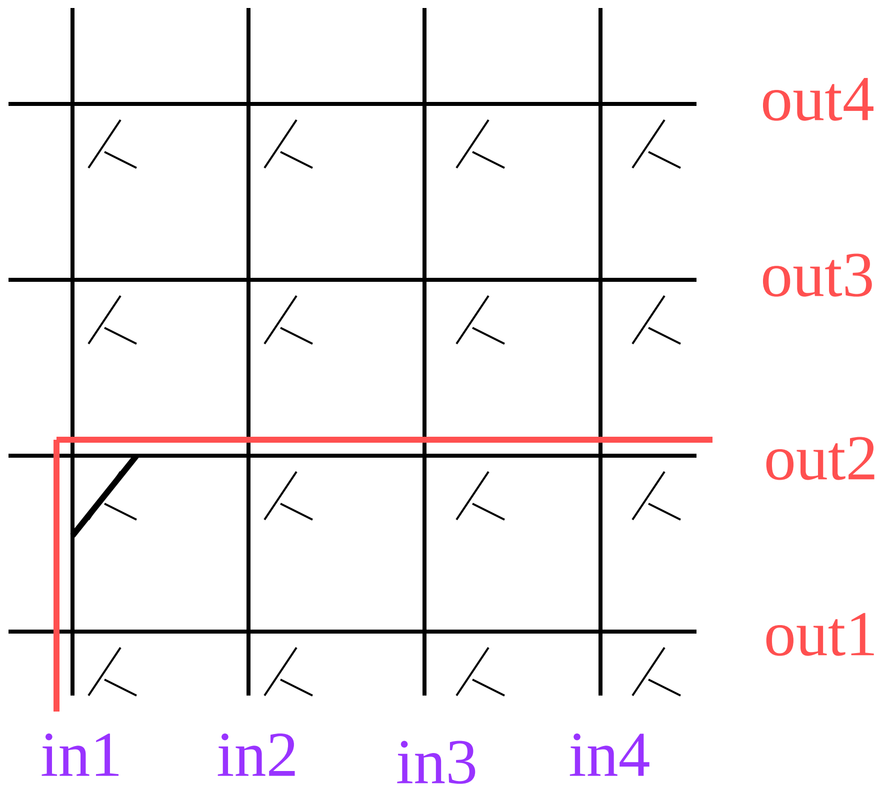


Bus arbitration

- Fixed: Same order of resolution every time.
- Fair: every PE has same access over long periods.
  - round-robin: rotate top priority among PEs.



Crossbar



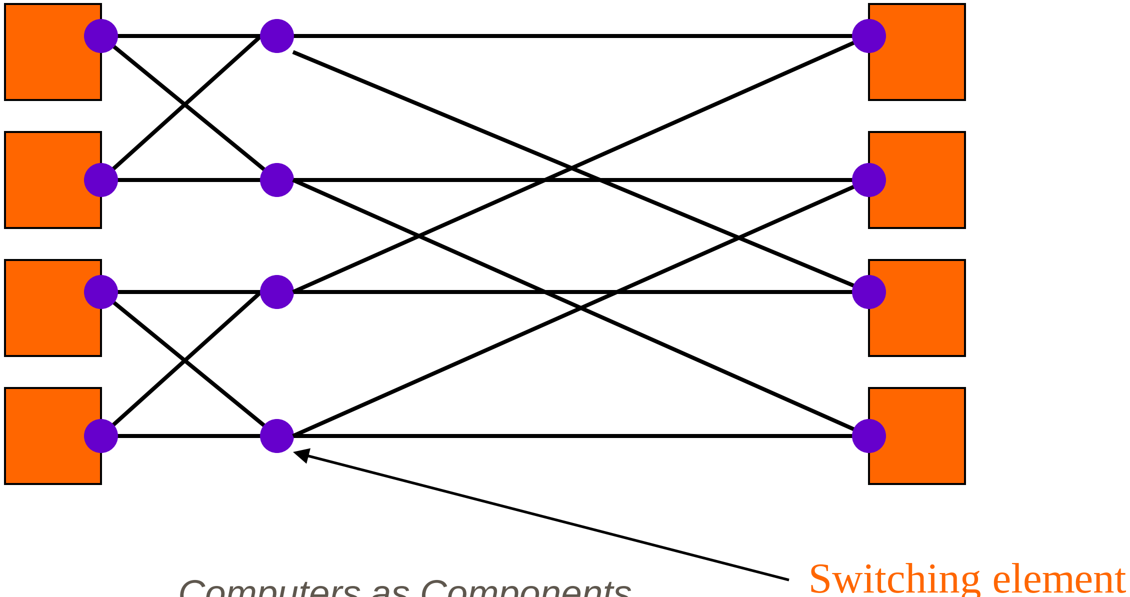
Crossbar characteristics

- Non-blocking (for 1-to-1 mapping)
- Can handle multiple multi-cast combinations.
- Size proportional to n^2.

Blocking: if some combinations of sources and destinations
For which messages cannot be delivered sinumtaneously.

Multi-stage networks

- Use several stages of switching elements.
- Often blocking.
- Often smaller than crossbar.



Message-based programming

- Transport layer provides message-based programming interface:

```
send_msg(adrs,data1);
```

- Data must be broken into packets at source, reassembled at destination.

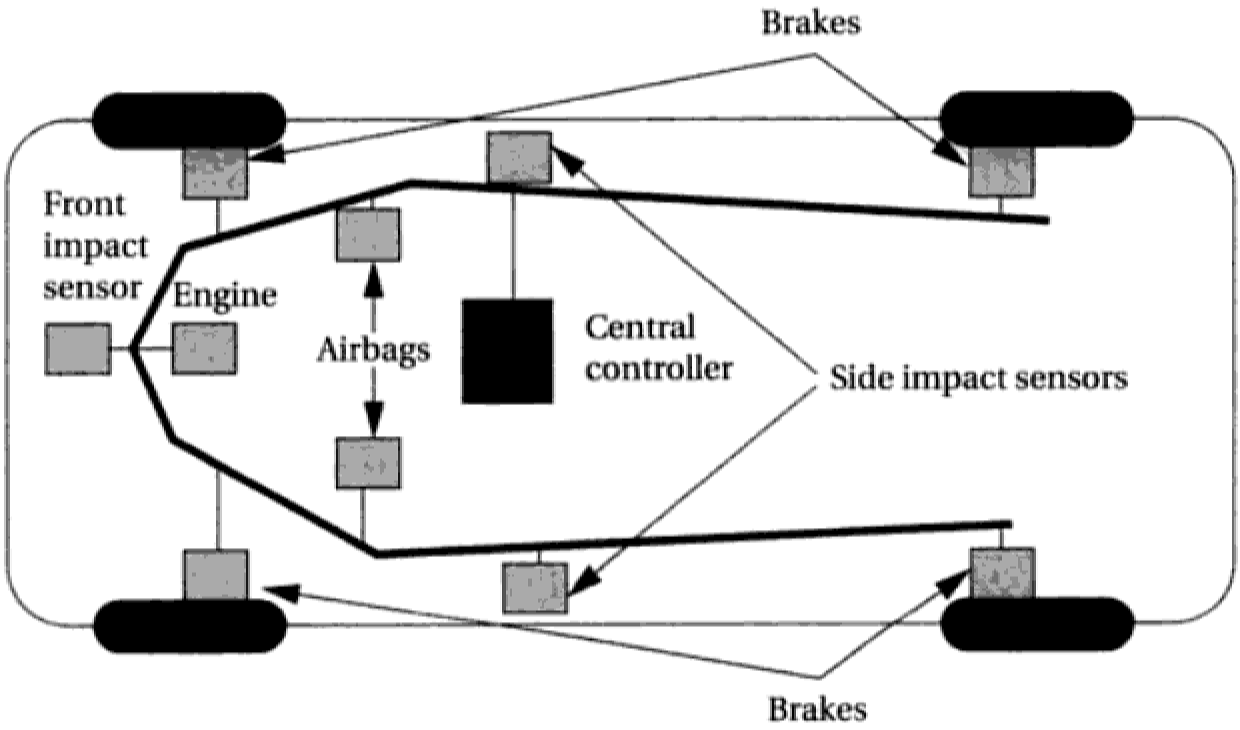
Blocking vs non-blocking

- Message passing: blocking
  - Simplest implementation
  - The routine is not returning until its data has received or transmitted
- Nonblocking network interface requires a queue of data to be sent

Data-push programming

- Data-push programming: make things happen in network based on data transfers.
  - Nodes send data out without any request from their intended users.
- Data-push programming makes sense for periodic data, which reduces data traffic on the network by automatically sending it when it is needed.

Data-push network



The sensors generally need to be sampled periodically. In such a system, it makes sense for sensors to transmit their data automatically rather than waiting for the controller to request it.

System buses

- Multibus [Intel] &VME [Motorola]: multi-card computer system
- ISA bus: I/O cards for PC-based systems
- PCI bus: high-speed interfaces for PC-based applications; replace ISA

Interconnection networks

- For embedded systems
- I^2C bus: microcontroller-based systems
- CAN bus: for automotive electronics
- Echelon LON network: for home and industrial automation

I^{2}C (inter-integrated circuit) bus

- Designed for low-cost, medium data rate applications.
- Command interface in a MPEG2 video chip
- Characteristics:
  - serial;
  - multiple-master;
  - fixed-priority arbitration.
- Several microcontrollers come with built-in I^{2}C controllers.

I^2C bus

- ~ up to 100 kbps (standard) ~up to 400 bps (extended)
- Two lines
  - SDL: serial data line
  - SCL: serial clock line

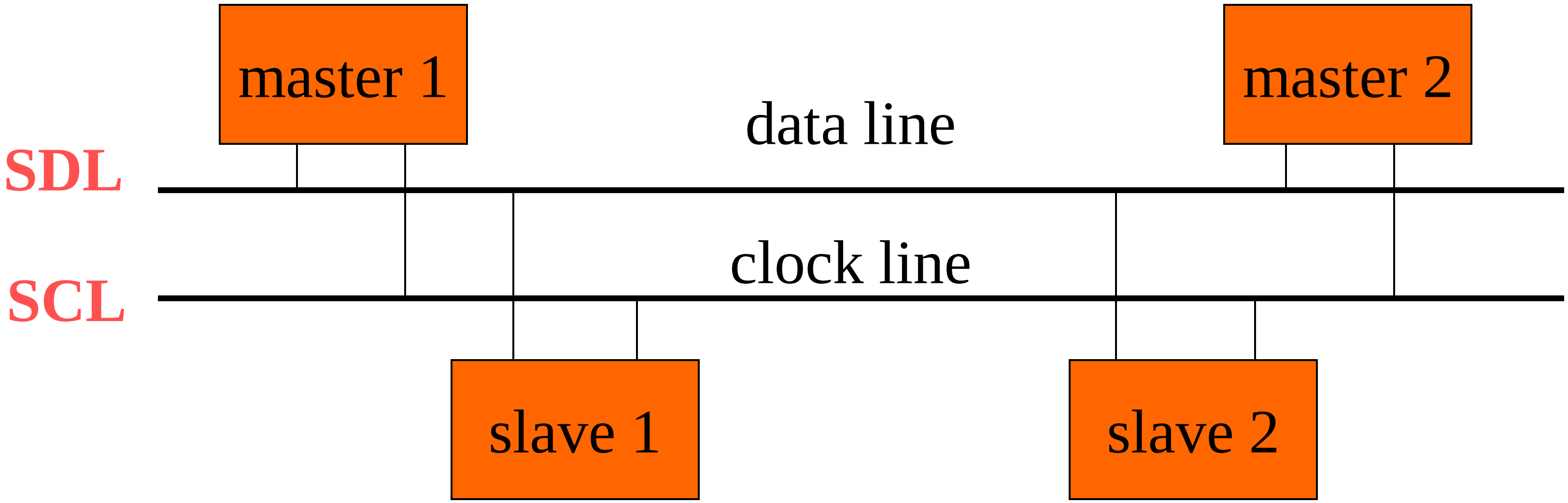
Terminology

- Transmitter – The device sending data to the bus
- Receiver – Device receiving data from the bus
- Master – device initiating a transfer, generates to clock and terminates a transfer
- Slave – Device addressed by the master
- Multi-master – more than one master can attempt to control the bus
- Arbitration – procedure to insure that only one master has control of ther bus at any instant
- Synchronization – procedure to sync then clocks of two or more devices

Inter-Integrated Circuit

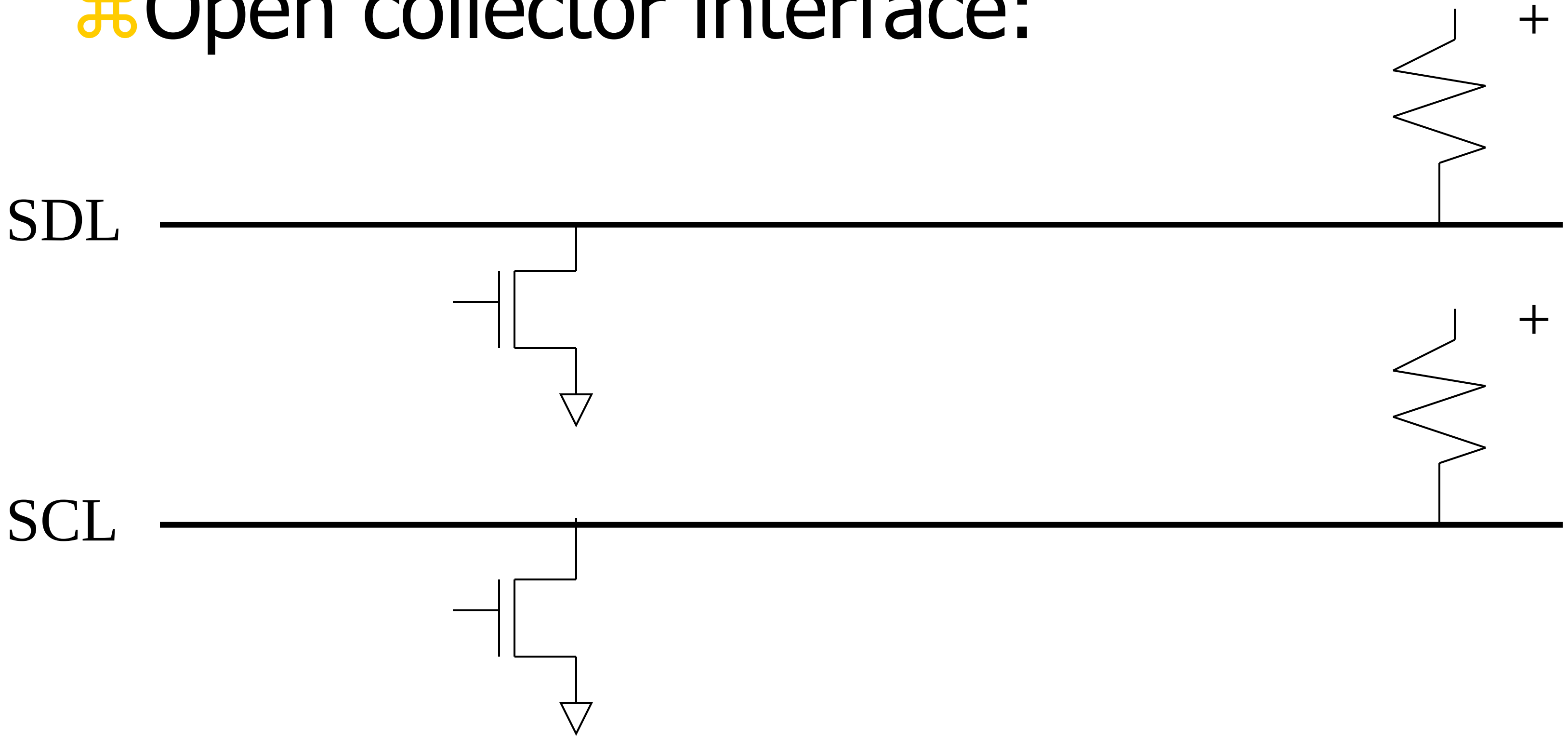
- Multi-master, two wire bus , up to 100 kbits/sec
  - One data line (SDA)
  - One clock line (SCL)
  - Master controls clock for slaves
  - Each connected slave has a unique 7-bit address

I²C physical layer



I^{2}C electrical interface

⌘Open collector interface:



I^2C signaling

- Sender pulls down bus for 0.
- Sender listens to bus---if it tried to send a 1 and heard a 0, someone else is simultaneously transmitting.
- Transmissions occur in 8-bit bytes.

Protocol

- Transfers are byte oriented, msb first
- Start: SDA goes low while SCL is high
- Master sends address of slave (7-bits) on next 7 clocks
- Master sends read/write request bit
  - 0-write to slave
  - 1-read from slave
- Slave ACKs by pulling SDA low on next clock
- Data transfers now commence

Complete I2C Transfer

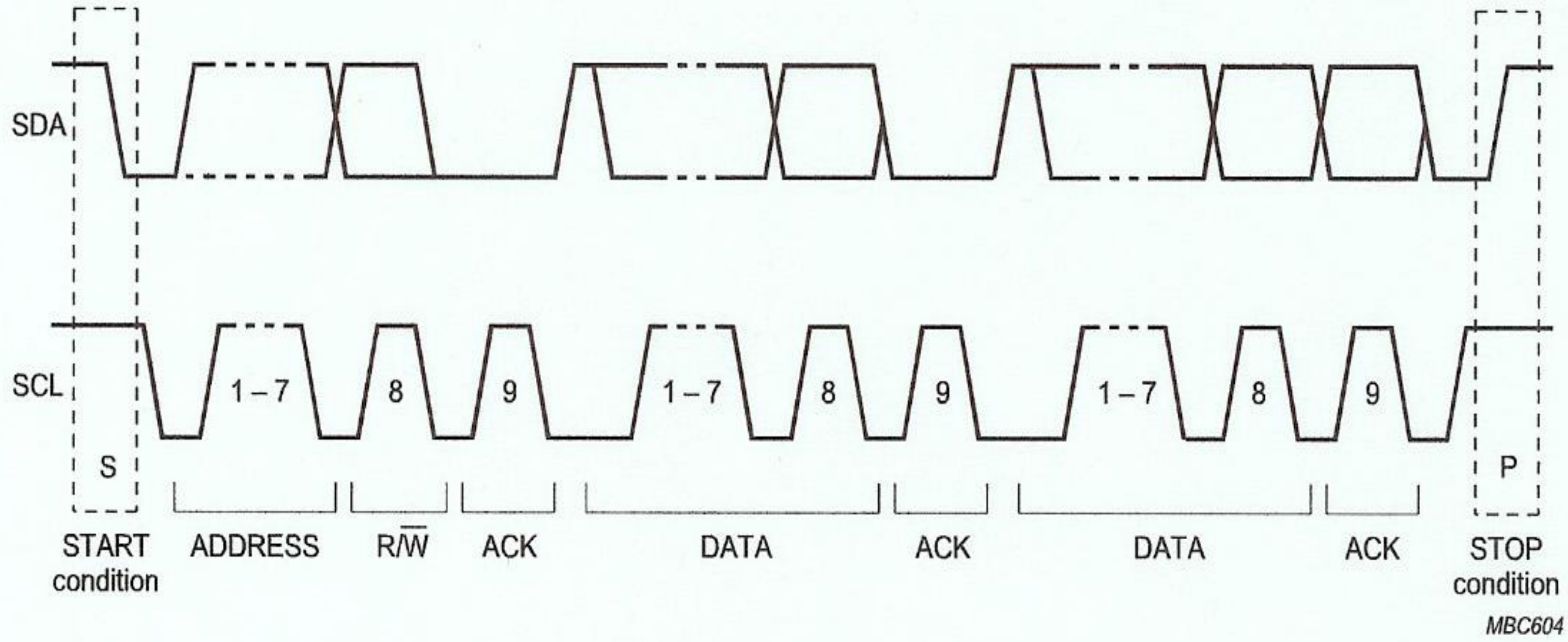


Fig.10 A complete data transfer.

Master-to-Slave Data Transfer

- Clock is controlled by master
- Data is written to slave on next 8 clock pulses
- Data receipt is ACKed by slave on 9th pulse by pulling SDA low
- When slave releases SDA, master can send next byte
- Master will eventually set a Stop condition by making a low to high transition on SDA with SCL is high

Master Writes to Slave

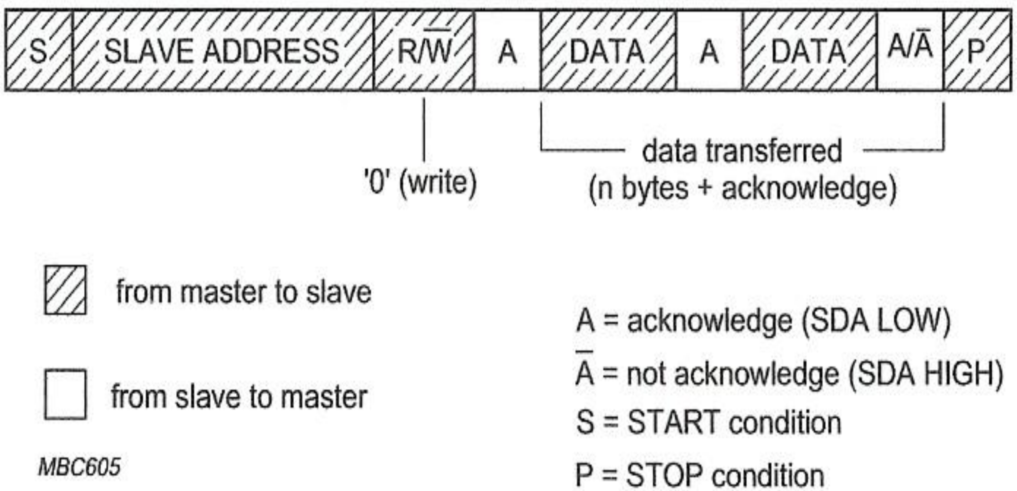


Fig.11 A master-transmitter addressing a slave receiver with a 7-bit address. The transfer direction is not changed.

Master Reads from Slave

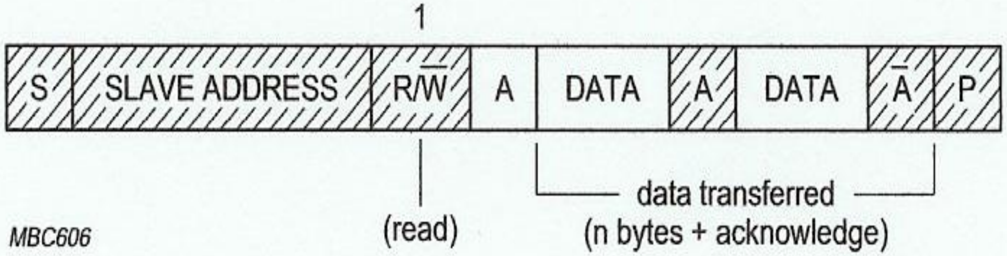
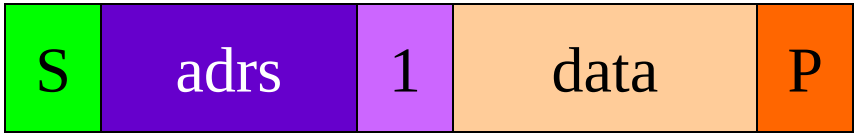


Fig.12 A master reads a slave immediately after the first byte.

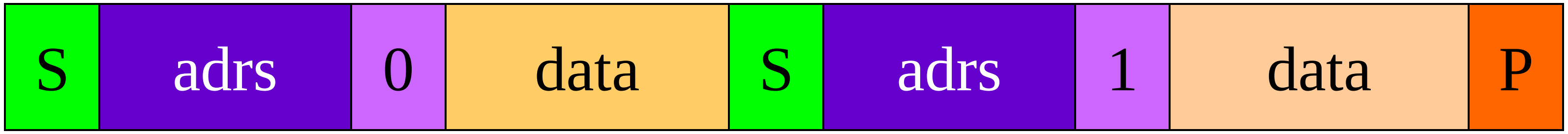
I²C transmissions

multi-byte write

read from slave



write, then read



I2C Extensions

- 10 bit addressing (up to 1024 addresses)
- Fast mode – up to 400 kbits/sec
- High-Speed – up to 3.4 Mbits/sec

I^2C bus arbitration

- Sender listens while sending address.
- When sender hears a conflict, if its address is higher, it stops signaling.
- Low-priority senders relinquish control early enough in clock cycle to allow bit to be transmitted reliably.