

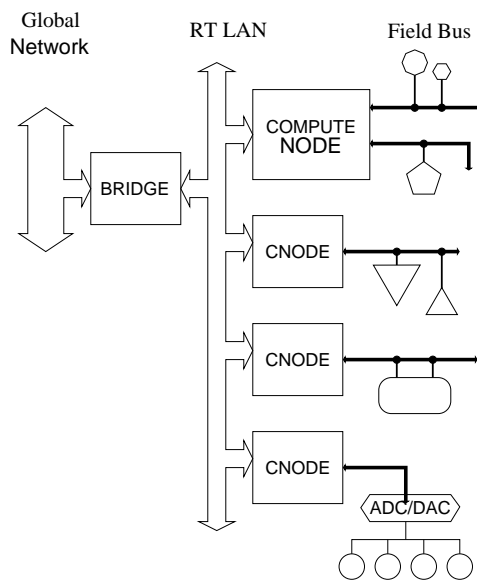
1 Embedded Communication

[Some material taken from [Kopetz, Ch. 7]; and [Wolf, Ch. 5]

1.1 Network Architecture

A typical embedded system has a multilevel network architecture. It is built around a system of computing *nodes* hosting a real-time, distributed operating system (RTOS) and interacting with each other through a local-area network (LAN). The LAN may be specially designed for real-time applications.

Each node is responsible for a collection of sensors, actuators, and other special-purpose devices. These are connected through one or more *field busses*, which may be primitive and dedicated to the qualities and characteristics of the devices. In particular, field busses may incorporate *analog-to-digital* and *digital-to-analog* converters (ADCs and DACs) that transform continuous electrical voltages to binary values. However, the trend in device control is toward standard I/O channel interfaces, such as USB.



At least one of the compute nodes is responsible for external connections to global resources. Such resources range from large-scale data sources to internet services like global time. In addition, connections for configuring, test and diagnosis, and telemetry are often connected through a bridge. Thus, any discussion of embedded-system communication must cover a broad range of characteristics and protocols, depending on the architectural level of that communication. We focus here on the LAN and field busses, leaving global networking to courses on that topic.

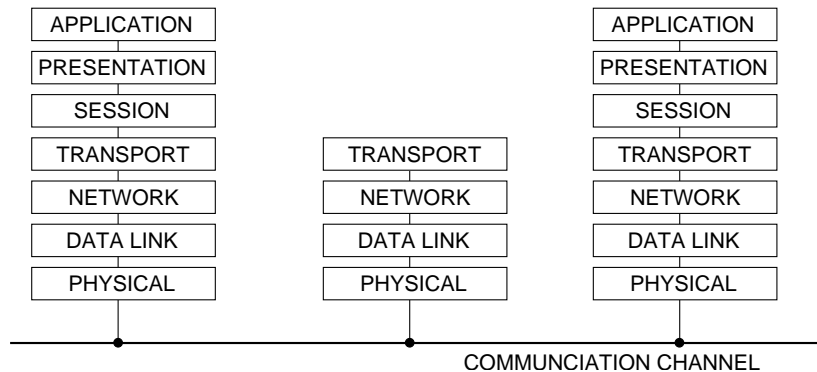
For an overview, the table below compares the characteristics of the network levels. Some of these characteristics are discussed further in the following sections

| Service Characteristics | Field Bus | RT LAN | Global Network |
|-------------------------|----------------|-------------|------------------------|
| message semantics | state | state | event |
| latency/jitter control | yes | yes | no |
| typical field length | 1-6 bytes | 6-12 bytes | 100 ⁺ bytes |
| clock synchronization | yes | yes | optional |
| fault tolerance | limited | yes | limited |
| membership service | maybe | yes | maybe |
| topology | multicast | multicast | point-to-point |
| communication control | multi-master | distributed | central or distributed |
| flow Control | implicit | implicit | explicit |
| low cost | very important | important | not very important |

[Kopetz]

1.2 OSI Reference model

The 1978 *Open Systems Interconnect* model is a starting point for any discussion of networking. Although it is intended as a standard conceptual model, OSI is often used also as an implementation model. It has seven levels. Not every network component instantiates all seven layers.



At each level a protocol is defined for communication at that level. We are mainly concerned in these notes with the lower levels.

The *communications channel* is any *medium* (or *carrier*) that can convey information: an electrical conductor, radio waves, optical fiber, shared-memory regions, and others. The examples below assume digital-electrical carriers, which remain common.

1.2.1 Directionality

In some networks, communication is *directional*, providing only for “point-to-point” connections. At one end of the channel a *sending buffer* provides am-

plification to help assure that the value on the carrier—voltage in the case of digital-electronic communication—is strong enough to be valid throughout the physical carrier’s length. At the other end, a *receiving buffer* and re-digitizes the possibly degraded value on the carrier before passing it to the receiving processing element (PE). Information flows only from sender to the receiver (Fig. ??). In order to communicate in the other direction, a second carrier is needed, with the opposite buffering.

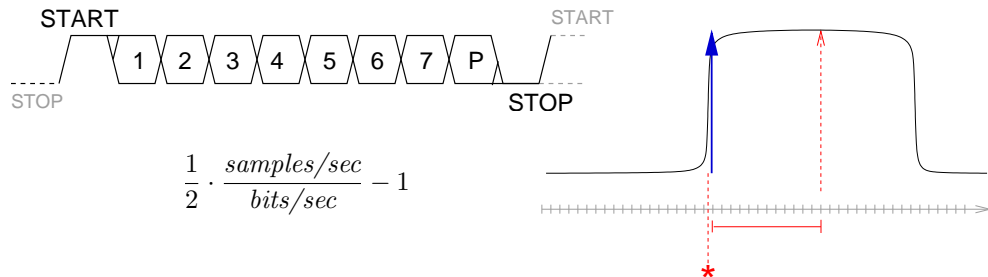
A two-way point-to-point connection is sometimes called *bidirectional*.

Many embedded communication networks are *omnidirectional*. Processing elements (PEs) are connected in parallel to a single carrier with a *transceiver* devices (Fig. 2. Like a bidirectional connection, each PE has both a receiving buffer and a sending buffer. However, the sending buffer is a *tri-state* device, which can *disconnect* the sending buffer. Any information presented on the carrier is broadcast to all connected PEs.

Two PEs sending at the same time is called a *collision*. As we shall see later, there are various protocols for dealing with collisions. For the moment, notice that a transceiver is capable of sending and receiving at once.

1.2.2 Physical Layer

EXAMPLE. RS232. RS232 is a standard family of protocols for asynchronous, directional, serial communication. The carrier is a single wire, so information can be transmitted only one bit at a time.



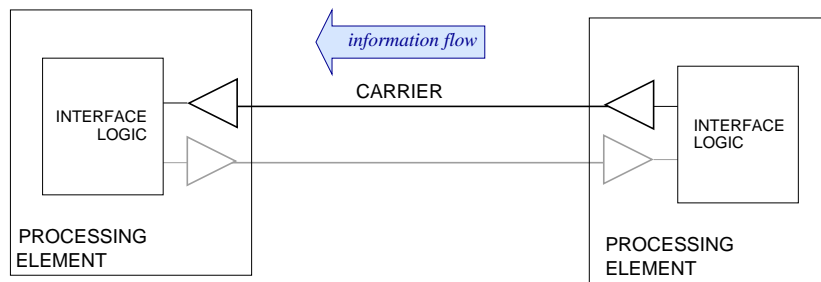
Ten bit-times are needed to send seven bits of information. The START bit must be 1 and the STOP bit must be 0. This assures at least one signal transition every ten bits.

Oversampling is used to estimate the “center” time, when a bit of information can be most reliably received.

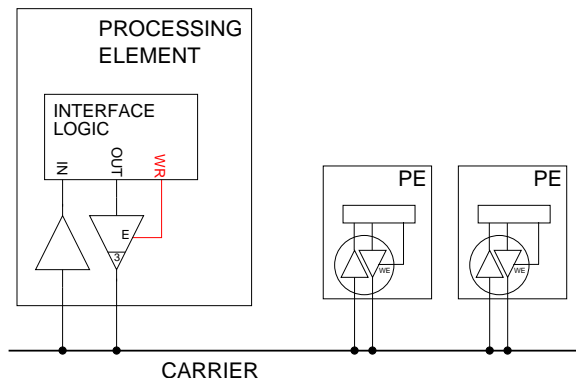
1.3 Overview and Terminology

1.3.1 Embedded Network Characteristics

- *Multicast*, not point-to-point



(a)



(b)

Figure 1: Directional (a) and omnidirectional (b) connections

- Low protocol latency, low jitter
- Encapsulated
- Dependable and resilient to transient failures
- End-to-end acknowledgment (e.g. message to actuator monitored by an independent sensor)

[At 3-mile Island] “Perhaps the most important and damaging failure . . . was that of the Pressure Operated Relief Valve (PORV) The PORV did not close; yet its monitoring light was signaling green (meaning closed).” [Koptez, p. 148]

1.3.2 Terms

SND, RCV, MSG – identifiers for the participants in a given message transmission from a *sender* to a *receiver* of a message. “The sender SND transmits the message MSG to the receiver RCV.”

permanence – relationship between MSG and all other messages sent before it. Informally, the worst-case time that MSG can be considered “received.” For example, in some systems SND can *withdraw* MSG while for some period of time, after which MSG becomes “committed.”

action delay – The time interval between the start of transmission of MSG and the point in time when MSG becomes permanent at RCV.

send time – the total time of sending a message

In systems with global time: the send time, t_{send} , measured by SND’s clock can be part of the message. If it knows the maximum message delay, d_{max} , then RCV can infer that MSG will become permanent at $t_{snd} + d_{max} + 2g$, where g is the local time granularity.

In systems without global time: RCV must wait $d_{max} - d_{min}$ ticks *after* MSG arrives before committing to an action:

$$t_{prm} = t_{snd} + 2d_{max} - d_{min} + g$$

EXAMPLE:

1. Operator B sends message $MSG_{BA} \equiv$ about-to-release-pressure to pressure monitor A
2. B sends $MSG_{BC} \equiv$ release-pressure-now to control valve C to release pressure

3. C releases pressure
4. Pressure sensor D sees sudden drop in pressure
5. D sends $MSG_{DA} \equiv \text{pressure-dropping!}$ to operator A.
6. Once MSG_{DA} arrives, how long should A wait before raising alarm?

1.4 PAR protocol

Positive Acknowledgment or Retransmission (PAR) protocol:

SND:

0. *{New message requested}*
1. reset retry-counter (CNT) to 0
2. start a local time-out (TO) interval
3. send message
4. await ACK message before time-out
 - if
 - ACK received before TO --> inform client of success and terminate
 - retry-counter exceeds limit --> inform client of failure and abort
 - retry-counter under limit --> increment retry-count, go to Step 2
 - fi

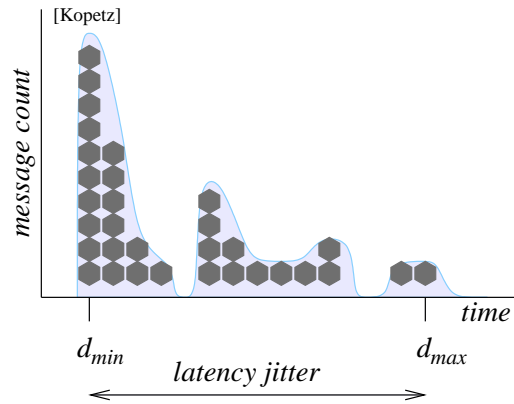
RCV:

0. *{new message arrives}*
1. send ACK to SND
2. if
 - MSG not already received --> pass MSG to client
 - MSG already received --> skip
- fi

Variations:

- SND initiates communication
- RCV has authority to delay SND
- SND detects communication errors
- time redundancy used for error correction

1.4.1 Latency Distribution



EXAMPLE

A *token ring* is a network access protocol in which one of the communication nodes holds a boolean “token” granting it exclusive access to the carrier (Fig. 3. This node and this node only may send a message while it holds the token. It may (must) eventually “pass” the token to a neighboring node.

Consider such a network with:

| | | |
|-------------------------|---|--|
| token rotation time: | = | 10 msec |
| msg. transmission time: | = | 1 msec |
| PAR timeout: | = | 22 msec |
| | | (a miss in each direction) |
| d_{min} | = | 1 msec |
| d_{max} | = | 55 msec |
| | | (44 for retry, 10 waiting for token) |
| error detection latency | = | 66 msec |
| | | (sender reports error after 3 tries) |
| action delay | = | $2d_{max} - d_{min} = 109$ msec |
| | | (receiver must hold for message to become permanent, p. 109) |

1.5 References

1. Hubert Zimmermann. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, **28**(4):425–432 (April 1980).

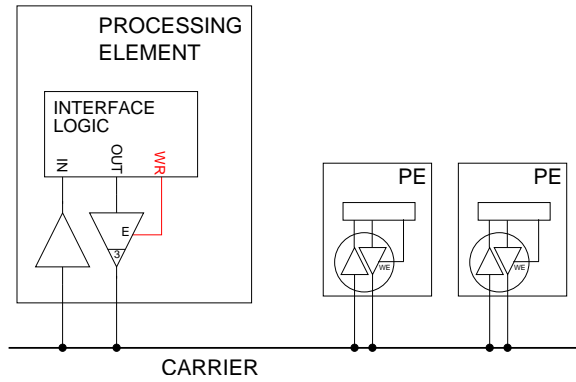


Figure 2: Processing elements connect to a common carrier through a *tranceiver*, whose input re-enforces (“buffers”) the digital value on the carrier. The *tri-state* output buffer presents a binary digital value (1 or 0) when *enabled*. When not enabled, the output buffer, in effect, disconnects from the carrier.

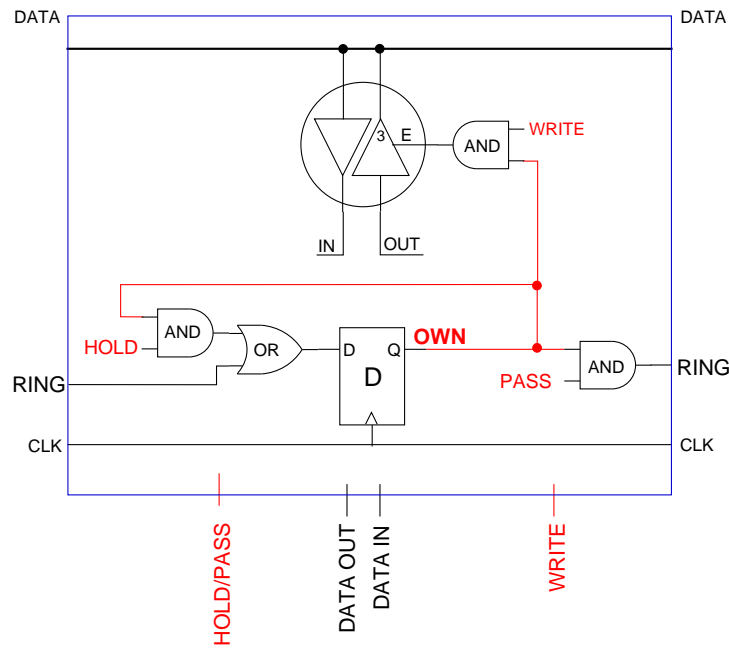


Figure 3: In a *token ring*, a flip-flop holds a bit designating which processing element has *WRITE* access to the carrier. In a valid state, exactly one PE’s flip-flop *HOLD*s a 1. When that PE decides to *PASS* its token, the 1 is presented to the neighboring PE.