Chapter 7
Real-Time Communication

Overview

- Introduction of a conceptual model for real-time systems
- Tasks, nodes, fault-tolerant units, clusters
- Simple and complex tasks
- Interface placement and interface layout
- Temporal control and logical control
- The history state

7.1 Real-Time Communication Requirements

Protocol Latency

Protocol latency is the time interval between start of message transmission at the CNI of the sending node, and the delivery of this message across the CNI at the receiving node (message should be permanent when delivered).

Latency Jitter: variation in the protocol latency, should be minimal for predictable behavior.

Simultaneous delivery in multicast: same image of an RT entity is needed at a number of nodes at the same time. A message should be delivered at all receiver CNIs within a short and known time interval.

Support for Composability

Temporal encapsulation of the nodes: there should be no control signals passing the CNI (the communication system should be autonomous). This permits for independent implementation and validation of communication system and application software.

Fulfilling the obligations of the client: clients must not overload their server with too many service requests. The communication system should exercise flow control over client requests, thus assisting in fulfilling the temporal obligations of the clients.

Flexibility

Many real-time communication systems must support different system configurations. Within the upper bound of communication traffic, there should be no need to change software on nodes that are not affected by a change.

Example: different configurations when ordering a car.

Delivering important sporadic messages such as an “emergency shutdown” message also require flexibility of the real-time communication system.

Error Detection

Detection of communication errors: errors occurring during message transmission should be detected and if possible corrected without increasing the jitter of the protocol latency.
Of particular concern is detection of message loss at the receiver.

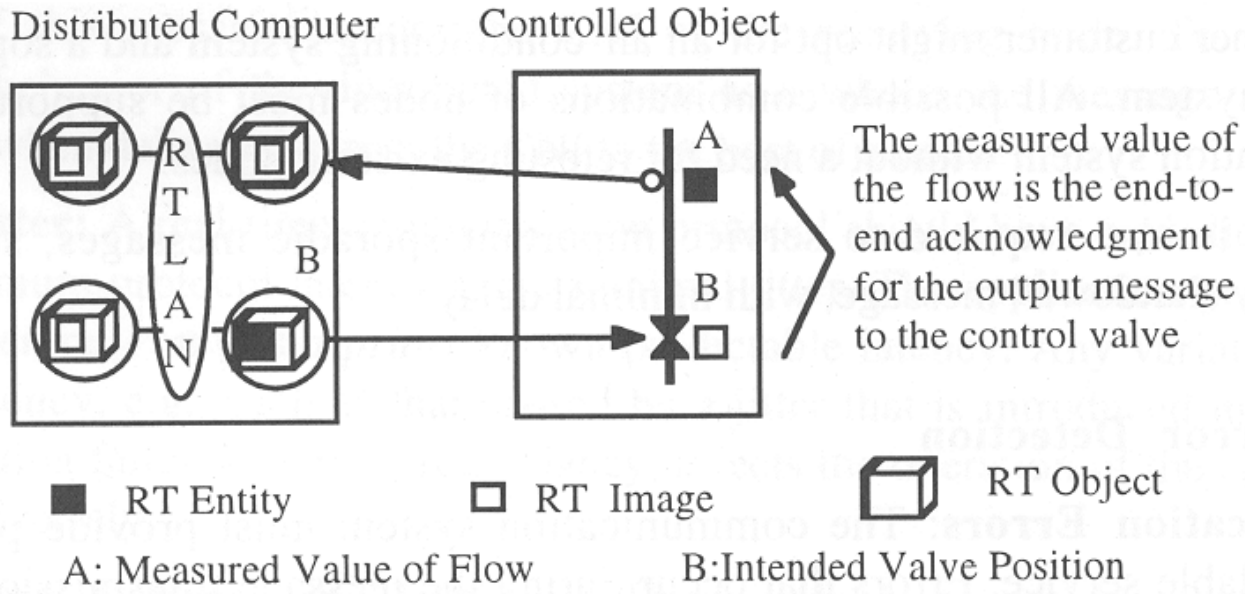
Example: node controlling a valve, which gets its command from another node. Wire is cut.

External electromagnetic interference (EMI), for example caused by a flash, can mutilate all messages for a few ms (blackout). The communication system should detect such a blackout, and continue with operation as soon as the blackout disappears.

Detection of node errors: Node failure must be detected by communication protocol, and must be reported consistently to all remaining nodes. This is the function of the membership service.

7.1 Real-Time Communication Requirements

End-to-end acknowledgement: Acknowledgement can come from a node that is different from receiver of message (e.g. sensor node monitoring valve action). Wrong end-to-end protocol can have serious consequences.



Fundamental design principle: “Never trust an actuator”.
Example: http://en.wikipedia.org/wiki/Three_Mile_Island_accident

7.1 Real-Time Communication Requirements

Physical Structure

Physical structure is determined by technical and economic considerations. Multicast requirements suggest structure that supports multicasting at the physical level, e.g., a bus or ring network. Fully connected point-to-point communication architecture is usually too expensive.

Bus versus ring: simple twisted pair cabling favors bus structure because of its simpler interface and higher fail-silent node failure resiliency, and simultaneous arrival of messages at all nodes. Optical cabling however favors ring structure. It easier to implement point-to-point connection of fibers than a fiber-based bus.

Physical separation of the nodes forming an FTU: separation of nodes making up an FTU (smallest replaceable unit, SRU) increases resilience against common-mode failures of these SRUs in case of a physical event (physical damage of a car section in a crash, rupture of hose). Example: steer-by-wire system.

7.2 Flow Control

Flow control refers to the control of speed of information flow from sender to receiver by the receiver, such that it can keep up with the sender.

Explicit Flow Control

In explicit flow control, the receiver sends an explicit acknowledgement message to the sender, informing the sender that the sender's previous message arrived correctly, and that the receiver is ready to accept a new message.

Most important protocol: Positive Acknowledgement-or-Retransmission (PAR) protocol.

PAR protocol: general procedure at the sender

1. sender is asked by its client to send a new message
2. sender initializes a retry counter to zero
3. sender starts a local time-out interval
4. sender sends message to receiver
5. sender receives acknowledgement message from receiver within time-out interval: client is informed of successful transmission, sender terminates
6. sender does not receive positive acknowledgement message within time-out interval: sender checks retry counter, increments the counter in case it has not reached the upper limit of retries, and goes back to 3. Otherwise it terminates and informs client about the failure.

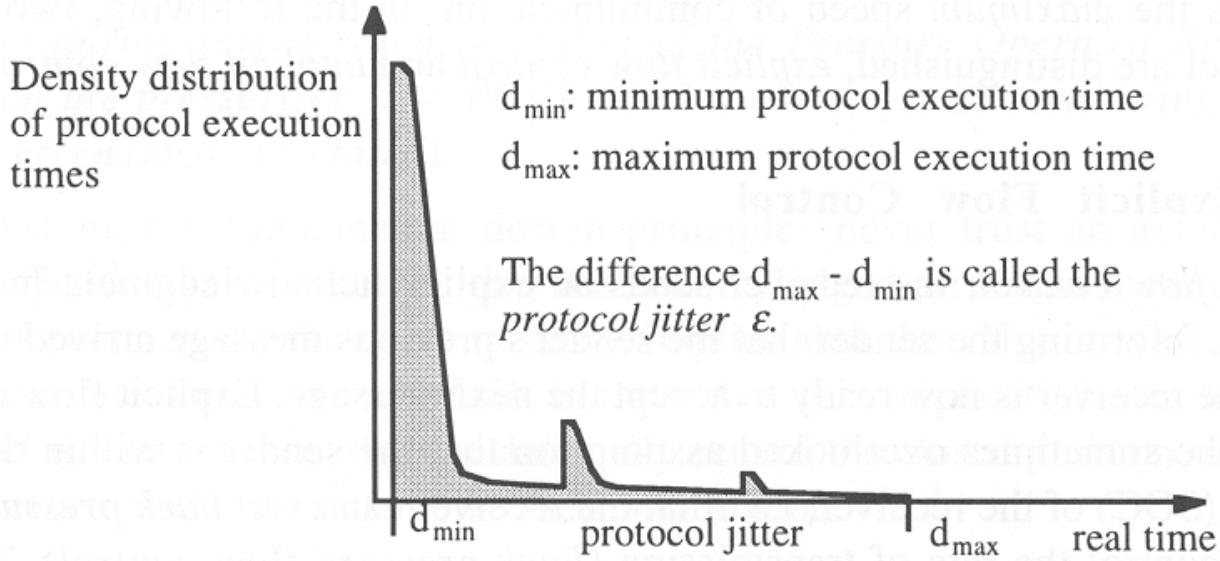
PAR protocol: general procedure at the receiver

1. receiver receives new message
2. receiver checks whether this message has already been received (e.g., last acknowledgement did not get to sender)
3. if message is new, it is delivered to client and an acknowledgement message is sent to sender
4. if message had already been received, another acknowledgement message is sent back to the sender

Note that the point in time when the sender is informed that the message has been received can be substantially different from the point in time when the receiver's client accepts the message.

- There are many variants of the PAR protocol, but all of them share these principles:
- The client at the sender's site initiates the communication
- The receiver has the authority to delay the sender via the bi-directional communication channel
- A communication error is detected by the sender, and not by the receiver. The receiver is not informed when a communication error is detected.
- Time redundancy is used to correct a communication error, thereby increasing the protocol latency in case of errors

Example protocol latency distribution of a PAR protocol with a retry counter of 2:



Example: action delay of PAR. Consider bus system without global time where a token protocol controls media access to the bus. The token protocol has a maximum token rotation time TRT of 10 ms. The time to transport the message on the bus is 1 ms. The granularity of the local clock can be neglected. A PAR protocol is implemented at the transport level on top of the medium access protocol.

1. What is the minimum time-out the PAR protocol must be set to?
2. What is the error detection latency of this protocol with a maximum of two retries?
3. What is the jitter of this protocol?
4. What is the action delay in this system, i.e. the time until the message becomes permanent?

Implicit Flow Control

7.2 Flow Control

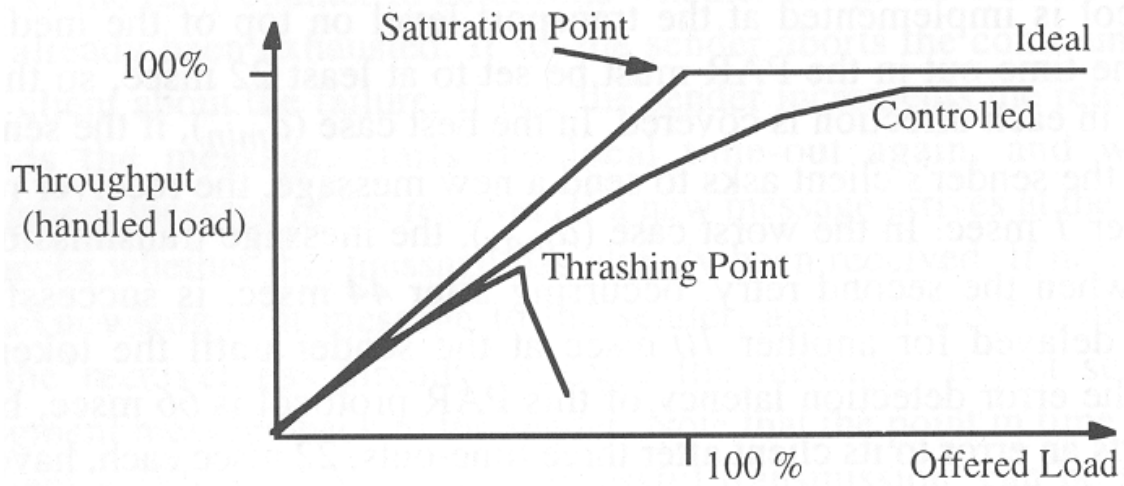
In implicit flow control, the sender and receiver agree a priori, i.e., at system start up, on the points in time when messages are sent. This requires a global time base.

Error detection is the responsibility of the receiver, which knows by looking at its global clock when an expected message fails to arrive.
Fault tolerance can be implemented by sending k physical copies of each message by ways of different channels. As long as at least one of the k copies arrives, the communication is successful.

The number of message to be delivered is always constant. Communication is unidirectional.

Thrashing

Trashing designates the phenomenon of sudden decrease of system throughput with increasing load.



Real-time systems must be free of thrashing phenomena.

Mechanism that cause thrashing: mechanisms that cause a more than proportional increase in resources with increasing load. Examples:

- Retry mechanism of PAR protocol. As communication system slows down with increasing load, the PAR protocol reaches time-outs more often. This generates additional load.
- Operating system scheduling service. With increasing computational load the amount of computing resources for generating the schedule increases. This further reduces the resources available for the actual computational tasks.

The only successful technique to avoid thrashing in explicit flow-control schemes is to monitor the resource requirements of the system and exercise back-pressure flow control as soon as a decrease in the throughput is observed.

Flow Control in Real-Time Systems

From the point of flow control, the most critical interface in a real-time system is the process interface between the controlled object and the computer system. Usually, not all events occurring in the controlled object are in the sphere of control of the computer system. A system may be overloaded by “event showers” and thereby miss important deadlines.

Example: Crash of a Swedish JAS39 Gripen fly-by-wire plane:

JAS 39 Gripen crash in Stockholm 1993 Aug 08 report summary

The Swedish Accident Investigation Board's preliminary report on the Gripen crash on Aug 8 is presented in the air force's magazine FlygvapenNytt 3/93. Summarily, the conclusions are:

7.2 Flow Control

- The only equipment malfunction before the crash was the electronic map which had nothing to do with the crash
- The flight control system, the engine and all other systems worked as specified until the aircraft impacted
- No external cause is suspected
- The pilot was properly trained and equipped
- The limits for minimum altitude and maximum angle of attack were exceeded insignificantly and did not have anything to do with the crash
- The manufacturer and customer knew that large and rapid stick movements could cause divergent Pilot Induced Oscillations, but considered the likelyhood of it actually happening insignificant, so all pilots weren't informed
- The red warning light was too late in telling the pilot that the control system was saturated, for him to do anything about it
- The low altitude of 270 m made it impossible for the pilot to try to regain control

The crash sequence started with a low speed 360 deg left turn at 280 m. The afterburner was lit, speed 285 km/h, load 2 G, bank angle 65 deg and angle of attack 21 deg. After finishing the turn, the control stick was moved to the right almost to the endpoint and slightly forward. The left wing's rear control surface rapidly went to the bottom position. The aircraft to bank to the right 20 deg past horizontal, angle of attack decreased to less than 10 deg. In order to fast regain a horizontal wing attitude, the pilot rapidly pulled the stick almost all the way to the left and continued to keep it slightly forward.

This caused the control surfaces to move with their maximum deflection speed, and as the flight control system then had little or no control surface movement on its own to work with, the stability margin was reduced. At the same time the alert system informed the pilot of this, and he no longer recognized the aircraft's behavior.

The aircraft started to roll to the left and pitch up. In response to this the control stick was moved almost to the right endpoint and some forward. The result was a roll to the right to 35 deg bank and a lowering of the nose to 7 deg below the horizon, whereupon the stick was pulled forcefully back and to the left. At the same time, the artificial stability system tried to

raise the nose, which in combination with the pilot's command caused a powerful pitch up. By this time, the control stick was fully forward, but the aircraft was already unflyable.
From exit of the turn until ejection the sequence took 6.2 s. The time from when the pilot didn't recognize the flying characteristics until stall took 2.7 s, but the control system warning wasn't shown until 1.2 s before the stall.

The cause of the crash was the misjudgements that PIO was so unlikely and that the warning light would tell about any problems early enough for no mention of this in the pilot's manual was necessary.

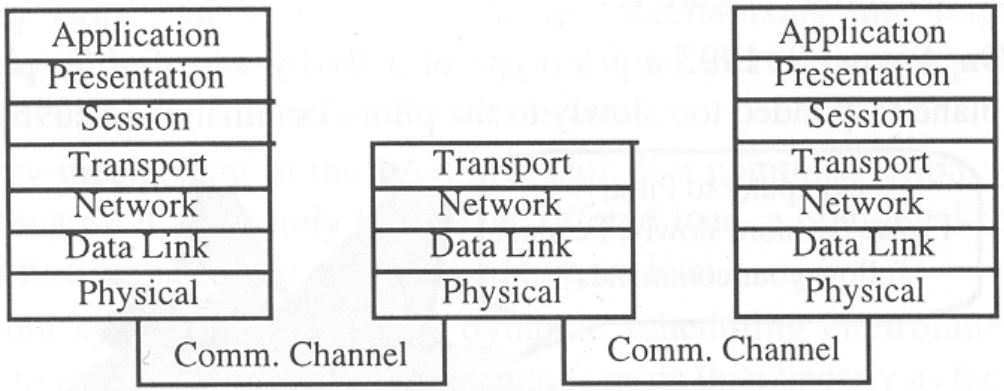
Example: monitoring and control system for an electric power grid with more than 100,000 different RT entities and alarms in case of a severe thunderstorm.

Interface between implicit and explicit flow control: is difficult to achieve and usually implemented by proper buffering. Determining the right buffer size is a delicate design issue.

7.3 OSI Protocols for Real-Time?

The OSI Reference Model

OSI reference model provides conceptual reference architecture for two computers to communicate with each other. The model is layered with seven layers, each encapsulating a protocol devoted to one particular aspect of the communication problem.



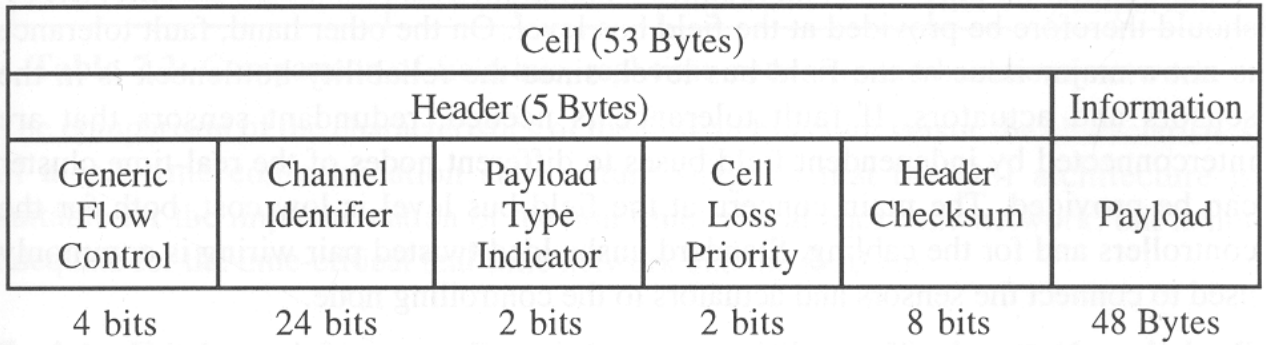
OSI protocol was initially just a reference model, but has been used as an implementation architecture, resulting in implementations with a stack of PAR protocols. The following assumptions are the base of many OSI conforming protocols:

- point-to-point connection between the two communication partners
- event-triggered messages
- PAR type communication protocols with explicit flow control and retransmission in case of error
- real-time performance, i.e., latency and latency jitter, is not an issue

Asynchronous Transfer Mode (ATM) and Real Time

7.3 OSI Protocols for Real-Time

The Asynchronous Transfer Mode (ATM) technology has been developed to provide real-time communication with low jitter over broadband networks. The information is packed into ATM cells with a size of 53 bytes.



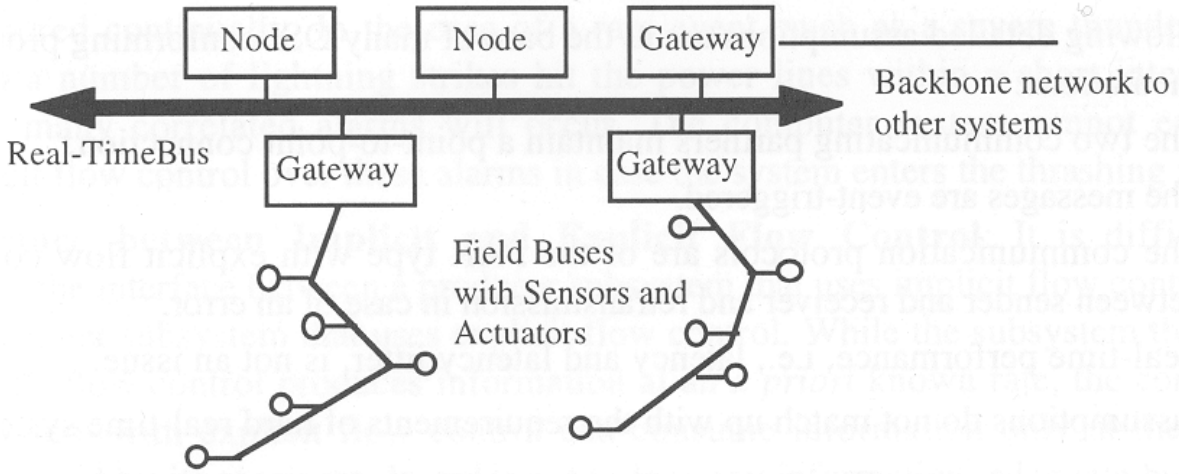
The application is free to decide which end-to-end protocols to implement on top of the basic ATM service. ATM technology is well suited for wide area real-time systems.

Real-Time Communication Architecture

We can distinguish three different types of communication networks:

- field bus
- real-time network
- backbone network

7.3 OSI Protocols for Real-Time



Field bus: interconnects a node of the distributed computer system to the sensors and actuators in the controlled object. Field bus messages have a short data field, containing state data, and are transmitted periodically with strict real-time requirements for latency and latency jitter.

Fault tolerance is not a major issue at the field bus level, since the reliability bottleneck is in the sensors and actuators. If fault tolerance is needed, redundant sensors that are interconnected by independent field buses to different nodes of the real-time cluster can be provided.

Main concern at the field bus level is low cost, both for the controllers and cabling. Standard unshielded twisted pair cabling is commonly used.

Real-time network: is at the core of the real-time cluster, and must provide the following services:

- Reliable and temporally predictable message transmission with low latency and minimal latency jitter
- Support for fault-tolerance to handle replicated nodes and replicated communication channels
- Clock synchronization in the range of microseconds
- Membership service with low latency for detecting node failures

A dependable real-time network must have replicated communication channels. The system must avoid a crash due to the failure in any single unit (e.g., babbling idiot failure). To avoid a central point of control failure, the system should be based on distributed control.

Backbone network: serves to exchange non time-critical information between the real-time cluster and the data-processing systems of an organization. Examples: production schedules, data collected regarding product quality and production times, standardized production reports.

Service characteristic	Field bus	Real-time network	Backbone network
Message semantics	state	state	event
Latency/jitter control	yes	yes	no
Typical data 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

The OSI architecture is adequate for the implementation of the backbone network, but is not suitable for the real-time network and field bus.

7.4 Fundamental Conflicts in Protocol Design

External Control versus Composability

Composability in the temporal domain requires:

- the CNI of every node is fully specified in the temporal domain
- the integration of a set of nodes into the complete system does not lead to any change of the temporal properties of the individual CNIs
- the temporal properties of every host can be tested in isolation with respect to the CNI

There is however the possibility that application functions interact in a manner that precludes high-level composability.

Example: Chain of telephones forwarding in a cyclic fashion. This cannot be detected by the low level communication system.

Example: These properties cannot be guaranteed in event-triggered systems, since the temporal control signals originate external to the communication system in such systems.

Example: If all nodes can compete at any point in time for a single communication channel on a demand basis, then conflicts with their extra transmission delay cannot be avoided.

7.4 Fundamental Conflicts in Protocol Design

Flexibility versus Error Detection

Flexibility implies that the behaviour of a node is not restricted a priori.
Error detection in architecture without replication is only possible if the actual behaviour of a node can be compared to some a priori knowledge of the expected behaviour.

These two requirements stand in conflict to each other.

Example: take an event-triggered system with no regularity assumptions, where access to a single bus is determined solely on message priority: if there is no restriction on the rate at which a node may send messages, it is impossible to avoid monopolization of the network by a single node that sends a continuous sequence of messages of the highest priority.

Sporadic Data versus Periodic Data

Periodic data must be transmitted with minimum latency jitter.
Sporadic data must be transmitted with minimum delay, at demand.

Mixing these two models will either cause an increase in the latency jitter of the periodic data, which would be transmitted at pre-calculated points in time, and would have to be moved due to the interleaving sporadic data.

The other way around, the sporadic data would have to be delayed if at the same time a slot for periodic data was scheduled.

It is impossible to satisfy both goals simultaneously.

Single Locus of Control versus Fault Tolerance

7.4 Fundamental Conflicts in Protocol Design

Any protocol relying on a single locus of control has a single point of failure (e.g., central master). Even in a token based system, the node that owns the token at that moment becomes a single locus of control. In case of a failure of such a node, a time-out mechanism must be used to regain control of the token by another node. This is similar to switching to backup master in case of a central master system.

Probabilistic Access versus Replica Determinism

Medium access protocols based on probabilistic mechanisms cannot guarantee that access to replicated communication channels is always resolved identically by competing nodes. Each replica may come to a different correct result, thereby leading to inconsistency in the system as a whole.

7.5 Media-Access Protocols

Media access strategy specifies which node is allowed to access the single communication channel at a particular point in time. This determines many properties of a distributed real-time system architecture. In this lecture we focus on event-triggered protocols. Time-triggered protocols will be treated in lecture 8.

Characteristics of a Communication Channel

Main characteristics of a communication channel are

- bandwidth
- propagation delay

Bandwidth:

Bandwidth indicates the number of bits that can be transmitted over a communication channel per unit time (typically described in bit/s). The bandwidth is limited by physical characteristics of the channel. For example, an unshielded twisted pair channel in a car may transmit up to 1 Mbit/s, limited by EMI constraints. A single wire channel in the same environment may only be able to transmit 10 kbit/s. An optical channel is not constrained by EMI, and can transmit many GBit/s.

Propagation delay:

Propagation delay is the time interval it takes for a bit to travel from one end of the channel to the other.
It is determined by the physical length of the channel and the transmission speed (electromagnetic, optical) within the channel. In vacuum, an electromagnetic wave travels at

about 300.000 km/s, or 30 cm/ns. In typical cables, the transmission speed is lower than that in vacuum, about 2/3 of that value. To travel across a length of 1 km, it takes a signal about 5 µs; or it travels 20 cm/ns.

The bit length of a channel denotes the number of bits that can traverse the channel within one propagation delay. For example, if the channel bandwidth is 100 Mbit/s and the channel is 200 m long, the bit length of the channel is 100 bits, since the propagation delay of this channel is 1 µs.

Limit to protocol efficiency:

At least a time interval of one propagation delay between two successive messages is required by a MAC protocol to a single channel in a bus system. This limits the data efficiency. Assume bit length of a channel of bl bits, and message length of m bits. The upper bound for data efficiency for any media access protocol in a bus system is

$$data\ efficiency < m/(m+bl)$$

Example: Consider bus of 1 km length, bandwidth 100 Mbits/s. Message length is 100 bit. The bit length of the channel is 500 bits (signal takes 5 µs to travel from end to end), and the limit to the data efficiency is 100/(500+100) = 16,6%.

CSMA/CD-LON

Carrier Sense Multiple Access/Collision Detection (CSMA/CD) is a protocol not requiring central local of control. Most prominent example is Ethernet.

Another example targeted for real-time systems is the LON protocol from Echelon. The LON media access protocol relies on a random generator to reduce probability of collisions at the start of transmission and during retransmission after a collision. A node wishing to transmit accesses the channel after a random delay after the carrier of the previous transmission has disappeared. The size of the randomizing window is a function of channel load and is designed to minimize the probability of collisions under high load (stochastic back-pressure flow control).

The LON protocol is defined in ISO standard 14908.

CSMA/CA-CAN

Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocols that avoid collisions, e.g. by bit arbitration. A prominent example of this type of protocol is the CAN (Control Area Network) Protocol, mostly used for automotive systems applications.

Field	Arbitration	Control	Data Field	CRC	A	EOF
bits	11	6	0-64	16	2	7

As CAN message consists of six fields, as shown above:

- 11 bit arbitration field, also acting as message identifier
- 6 bit control field
- 0-54 bit data field
- 16 bit CRC field ensuring Hamming distance of 6

- 2 acknowledge fields

The arbitration logic assumes the existence of recessive and dominant states on the communication channel, such that the dominant state can overwrite the recessive state. This is only possible if the propagation delay of the channel is smaller than length of a bit cell.

When a node intends to transmit a message, it puts the first bit of its message identifier on the channel. If at the same instant in time another node does the same, the one with the first recessive bit in its message identifier backs off. For example, assuming a “0” is the dominant state, a node with a message identifier consisting of all “0” always wins. The message identifier thus determines the message priority.

Token Bus-Profibus

In a token-bus system, the right to transmit is contained in a special control message, the token. The token owner is allowed to transmit.
Two time parameters determine the response in a token system: the token hold time, and the token rotation time.
A serious error in a token system is the loss of the token. In this case network traffic is disrupted until some other node detects the “silence” by monitoring a time-out.
Example: Profibus used for process automation.

Minislotting – ARINC 629

With minislotting, time is partitioned into a sequence of minislots, each longer than the length of the propagation delay of the channel. Every node is assigned a unique number of minislots that must elapse, with silence on the channel, before it is allowed to transmit.

Example: ARINC 629 protocol, used in Boeing 777 aircraft.

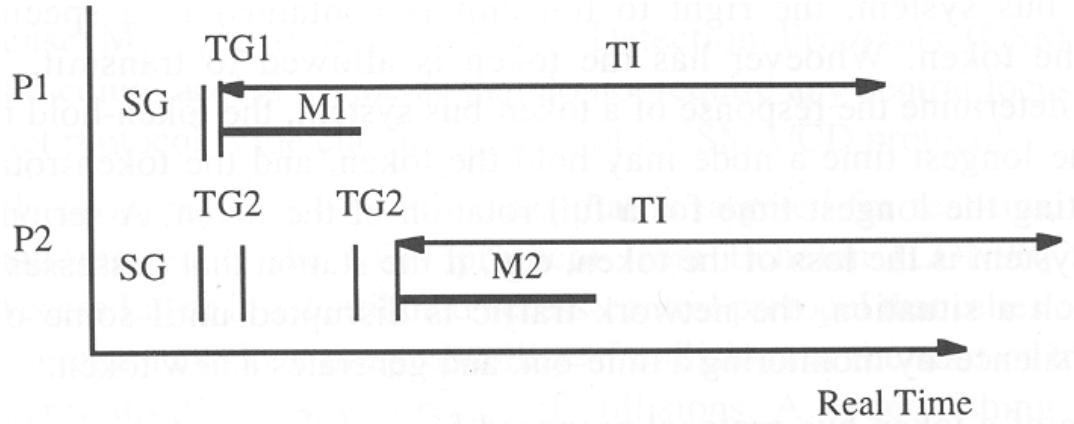
7.5 Media-Access Protocols

In ARINC 629, medium access is controlled by three time-out parameters:

- Synchronization gap SG, controlling entrance to a “distributed waiting room”
- Terminal gap TG controlling access to the bus
- Transmit interval TI disabling a host from monopolizing the channel

SG and TI are identical for all nodes. The following relationships hold:

$SG > \max\{TG_i\}$ for all processes i
$TI > SG$



Example: two processes P1 and P2 want to transmit a message. TG1 is shorter than TG2. Initially, both processes wait for an interval of silence longer than SG, the admit time-out for the waiting room.

After they have entered the waiting room, both processes wait for another period of silence corresponding to their individual terminal gaps. Since all TGs are different, the process with the shorter TG starts transmitting if the bus is idle at the moment its time-out has elapsed.

At the start of transmission, P1 sets its time-out TI to block any further sending activity in this epoch by node P1, to avoid monopolizing the channel.

As soon as P1 has started transmitting, P2 backs off until P1 has finished (after M1 ends). Then another terminal gap for P2 is started, and at the end of this terminal gap, in case the bus is idle, M2 is transmitted.

All nodes that must send a message in this epoch complete their sending activity before any other node may start a new epoch, because $SG > \max\{TG_i\}$.
Typical values for the time-out parameters on a 2 MBit/s channel are:

- TG (determined by propagation delay): 4-128 µs
- SG: 0.5-64 ms

Central Master - FIP

A central master protocol relies on a central master for bus access control. In case of master node failure, another node takes over.

Example: FIP protocol.
In a FIP system, the central master (call bus arbitrator in FIP) possesses a list with names and periods of all messages in the entire system. The master periodically broadcasts the names of a variable from this list on the bus. The node that produces that variable responds with a broadcast of the contents of that variable.

All other nodes listen to the broadcast and accept the content of that variable if needed.
Free time after being polled by the master can be used by the polled node for sporadic messages.

TDMA - TTP

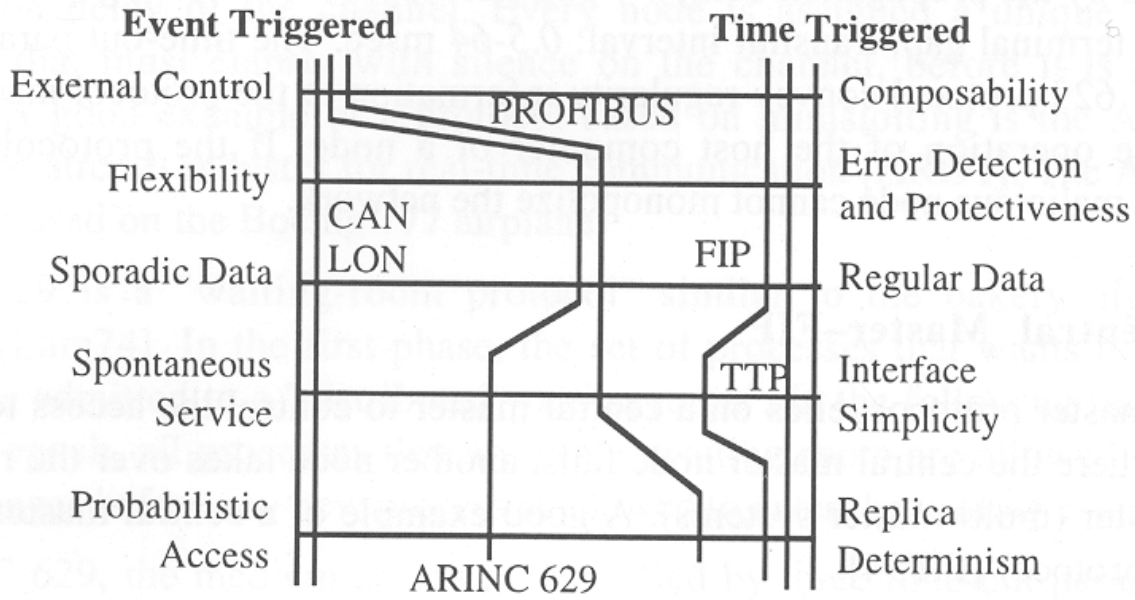
In a Time Division Multiple Access (TDMA) protocol the right to transmit a frame is controlled by the progression of real time. This presupposes that a global time base is available at all nodes.

In a TDMA system, the total channel capacity is statically divided into a number of slots. A unique sending slot is assigned to each node.

The sequence of sending slots within an ensemble of nodes is called a TDMA round. A node can send one frame per round. If there is no data to be sent, an empty frame is transmitted. The sequence of all different TDMA rounds is called a cluster cycle. The length of the cluster cycle determines the periodicity of the TDMA system.

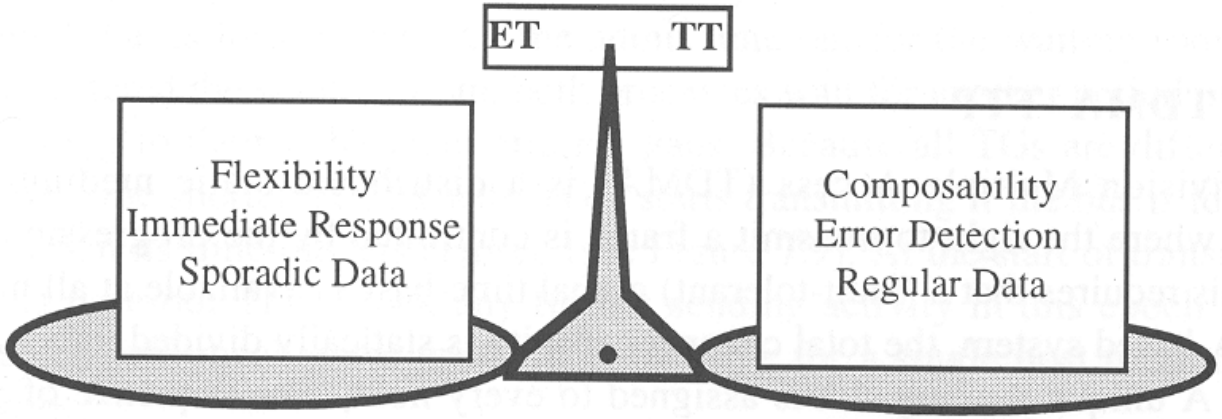
Comparison of the Protocols

Each of the protocols mentioned has its specific advantages and disadvantages. There is not a single real-time communication protocol that is able to meet all requirements listed in section 7.1.



The following figure illustrates some of the tradeoffs to be made when designing real-time communication protocols.

7.5 Media-Access Protocols



One the one side, event-triggered protocols provide flexibility, brief response times, and support well transmission of sporadic data.

On the other side, time-triggered protocols favour composability, facilitate error detection and are more robust in the case of regular, periodic data.

The CAN protocol is a stringent implementation of the event-triggered family, while the Time-Triggered Protocol and FlexRay are positioned clearly in the time-triggered family of protocols.

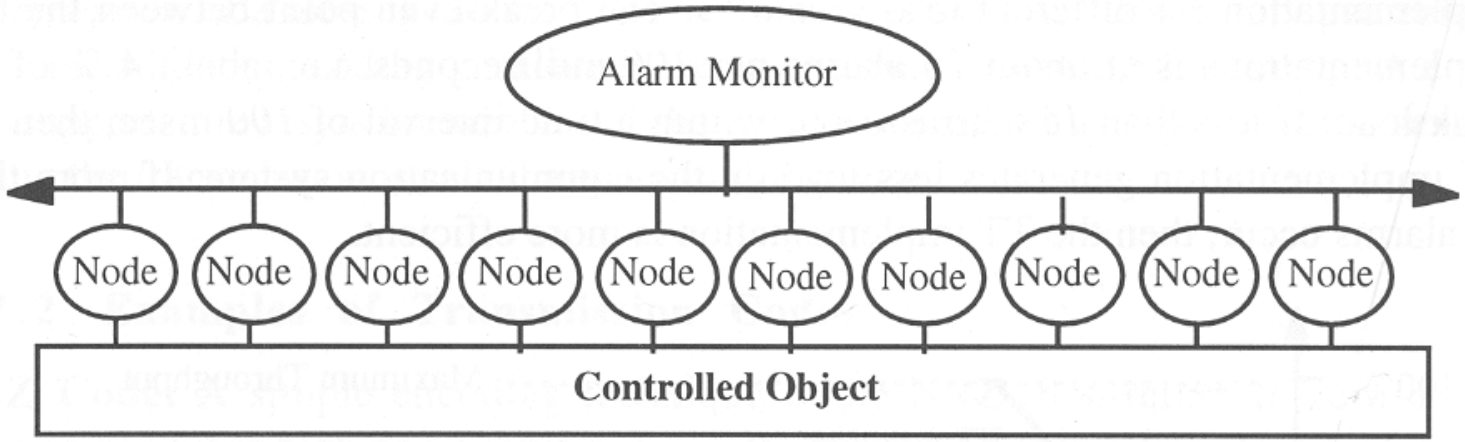
ARINC 629 and Profibus are placed in between these two corners of the design space.

7.6 Performance Comparison: ET versus TT

Performance of ET protocol is superior to that of TT protocols if the environment requires the exchange of many sporadic messages with unknown request times.
In an environment with many periodic messages to be exchanged, the performance of a TT protocol is superior to that of an ET protocol. In the following sections this is illustrated by a (biased) example.

Problem specification

Cluster with ten interface nodes connected to a controlled object. Nodes monitor and process alarms, and convey them to an Alarm monitor node which displays the alarms to an operator. Each interface node observes 40 binary alarms. The operator must be informed within 100 ms after an alarm signal has been raised. The bandwidth of the communication channel is 100 kbit/s.



ET and TT Solutions

7.6 Performance Comparison: ET versus TT

For both solutions we use the same basic protocol, the CAN protocol. The first implementation is event-triggered, the second time-triggered.

Event-triggered implementation: message is being sent to the operator as soon as the alarm is recognized. The event message contains the name of the alarm that is encoded into a CAN message with a data field of one byte (40 binary alarms would actually only require 6 bit).

CAN overhead is 44 bit, and we choose an intermessage gap of 4 bit. Thus, the total alarm message is 56 bit long.

With a bandwidth of 100 kbit/s, about 180 messages can be transported within the latency interval of 100 ms. This is less than the 400 messages that could occur simultaneously.

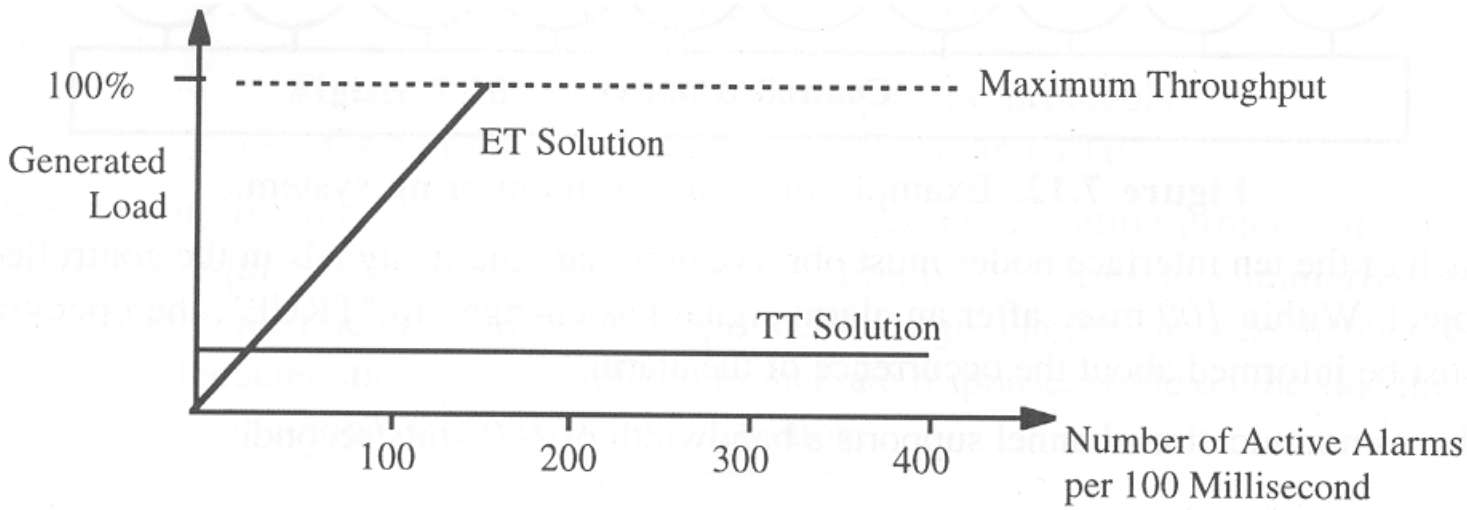
Remark: we could of course summarize the 40 bit into a single message of 5 byte, so if they would occur all at the same time, they would be sent simultaneously. But we want to make the TT protocol look good.

Time-triggered implementation: message is being sent to the operator in a periodic state message every 100 ms (our permissible latency interval). The state message is the same as the event message described above, except that we reserve 5 data bytes to accommodate all 40 alarms in a single message. With all other parameters the same, the message length is 88 bit. With the given bandwidth of 100 kbit/s, about 110 messages can be transported in the given latency interval of 100 ms. Because we need at most 10 periodic state messages to cover the peak load scenario, the TT implementation requires less than 10% of the available bandwidth. In addition, the TT implementation provides an error detection capability.

Comparison of the Solutions

7.6 Performance Comparison: ET versus TT

A comparison of the generated load for the ET protocol and TT protocol solutions for different load scenarios illustrates a break-even point of about 16 alarms per 100 ms.



For less than 16 alarms per 100 ms, the ET implementation is more efficient than the TT implementation. At about 180 alarms per 100 ms the ET implementation loads the communication channel 100%. The load of the TT solution is about 10% in all situations.

7.7 The Physical Layer

The physical layer specifies:

- Transmission codes (coding of bit pattern on physical channel)
- Transmission medium
- Transmission speed
- Physical shape of bit cells

Protocol design is influenced by physical layer properties, and vice versa.

Example: CAN protocol presupposes that every bit cell stays long enough on the channel such that priority arbitration can be performed. This limits the speed of the network to a bit cell size longer than the propagation delay.

Properties of Transmission Codes

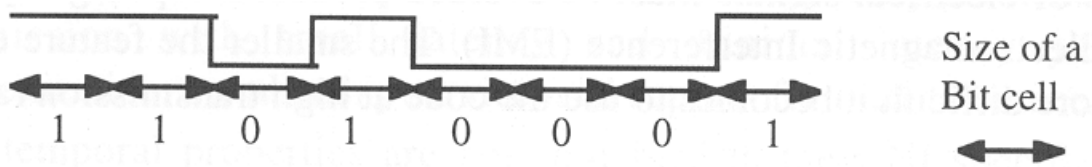
In asynchronous communication, the receiver synchronizes its receiving logic with that of the sender only at the beginning of a new message. Clock drift limits the message length, for example to about 10 bit in a UART device using a low cost resonator (drift rate 10^{-2} s/s).

In synchronous communication, the receiver synchronizes it's receive logic during the reception of a message to the ticks of the sender's clock. This requires frequent transitions in the bit stream which are enforced by proper encoding. A code that supports resynchronization of the receiver's logic to the sender's clock during transmission is called a synchronizing code.

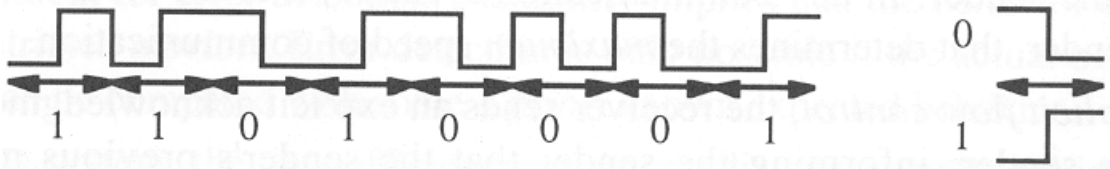
Examples of Transmission Codes

Points to Remember

NRZ Code: (non-return-to-zero code), “1” bit is high, and “0” bit is low. If all data bits are “1” or “0”, there will be no signal transitions in the bit stream. Therefore, this is a non-synchronizing code. This code cannot be used in a synchronous environment.

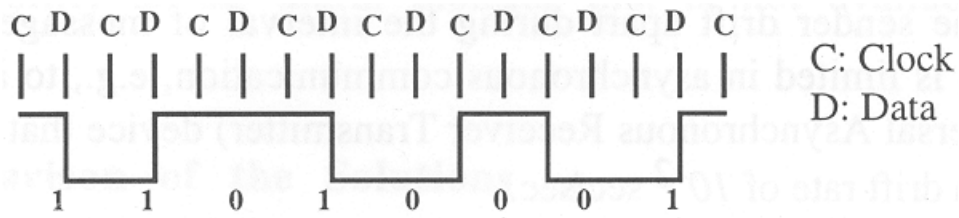


Manchester Code: this code has a synchronizing edge in every bit cell of the transmitted signal. The Manchester code encodes a “0” as a high/low bit cell and a “1” as a low/high bit cell. This is a synchronizing code, but it has the disadvantage that each bit requires two feature elements to encode (0-1 for “0” and 1-0 for 1). Thus, the actual “bit rate” on the bus is twice the payload bit rate.



Modified Frequency Modulation (MFM): has a feature size of one bit cell, and is also synchronizing. It distinguishes between a data point and a clock point. A “0” is encoded by no signal change at a data point, a “1” requires a signal change at a data point. If there are more than two “0” in sequence, the encoding rules require a signal change at clock points.

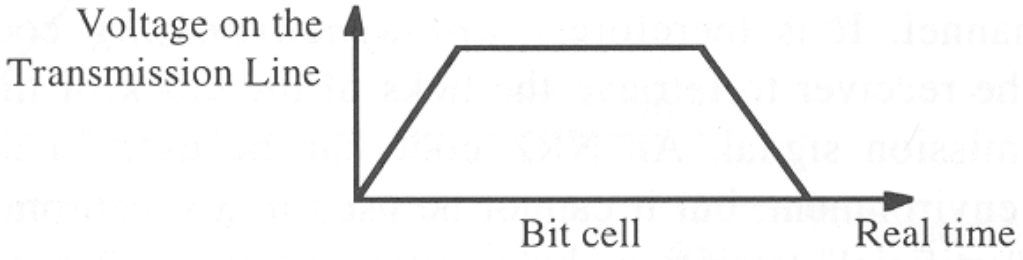
Example: encoding the bit sequence of “1101 0001” in MFM:



Signal Shape

The physical form of the feature element influences the electromagnetic emission (or electromagnetic interference, EMI).

Steep edges must be avoided, since they cause significant spectral components at high frequencies and increase EMI.



Points to Remember

Different protocols in the car:

CAN

Flex Ray

LIN

MOST

What is the criteria for choosing which communication should be used in which system.

1. Topology
2. Bitrate
3. Data correctness
4. Error Handling
5. Bus protocol
6. Backward compatibility
7. Gateway
8. Hardware considerations
9. Propagation delay
10. Bandwidth
11. Fault handling on hardware

Points to Remember

12. Medium of communication
13. External disturbance – EMI and ESD
14. Message encoding
15. Spectrum.
16. Protocol Latency