Real-Time Networks and Networked Control Systems

Background

Real-Time Networks

Protocol Stack

Automotive Network Examples

– LIN, CAN, FlexRay

Protocols

– TTP, Event-triggered

Networked Control Systems – Industrial Networks

- Profibus, Profinet

Industrial control systems (ICS), SCADA systems,

Cyber security in ICS

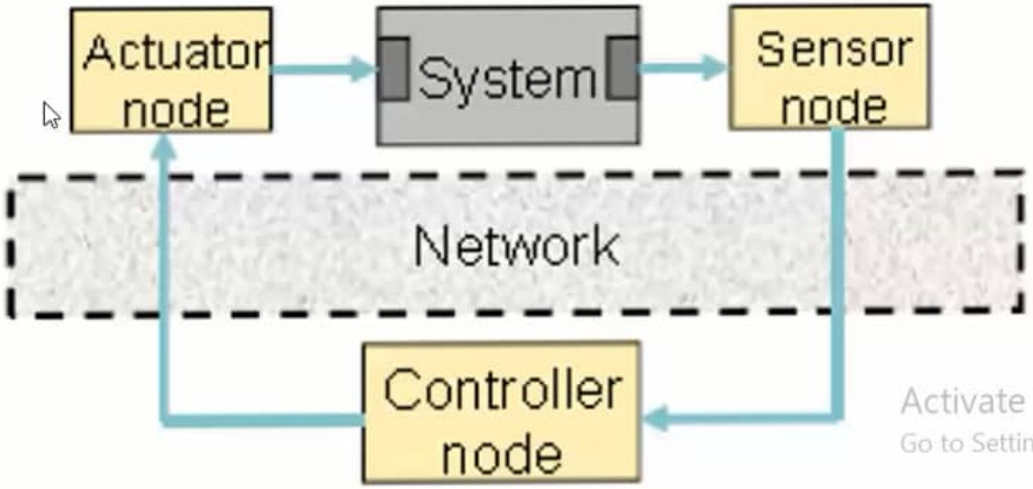
Background

Distributed architectures are pervasive in many application fields:

1. – Industrial automation (manufacturing, processing,…)
2. – Transportation units and systems (avionic, automobile, trucks, rail, …)
3. – Multimedia systems (remote surveillance, industrial monitoring, video on demand, …)

In many cases there are **critical timeliness** and strict **safety** Requirements (e.g., in 1 and 2)

It is increasingly common that control loops are closed over networks (= **networked control**)



Background

Motivations for distributed architectures:

- Processing closer to data source /sink
  - ”Intelligent” sensors and actuators
- Dependability
  - Error-containment within nodes
- Composability
  - System composition by integrating subsystems
- Scalability
  - Easy addition of new nodes with new or replicated functionality
- Maintainability
  - Modularity and easy node replacement
  - Simplification of the cabling

Background

Today there are many different networks with real-time capabilities aiming at different application domains, e.g.

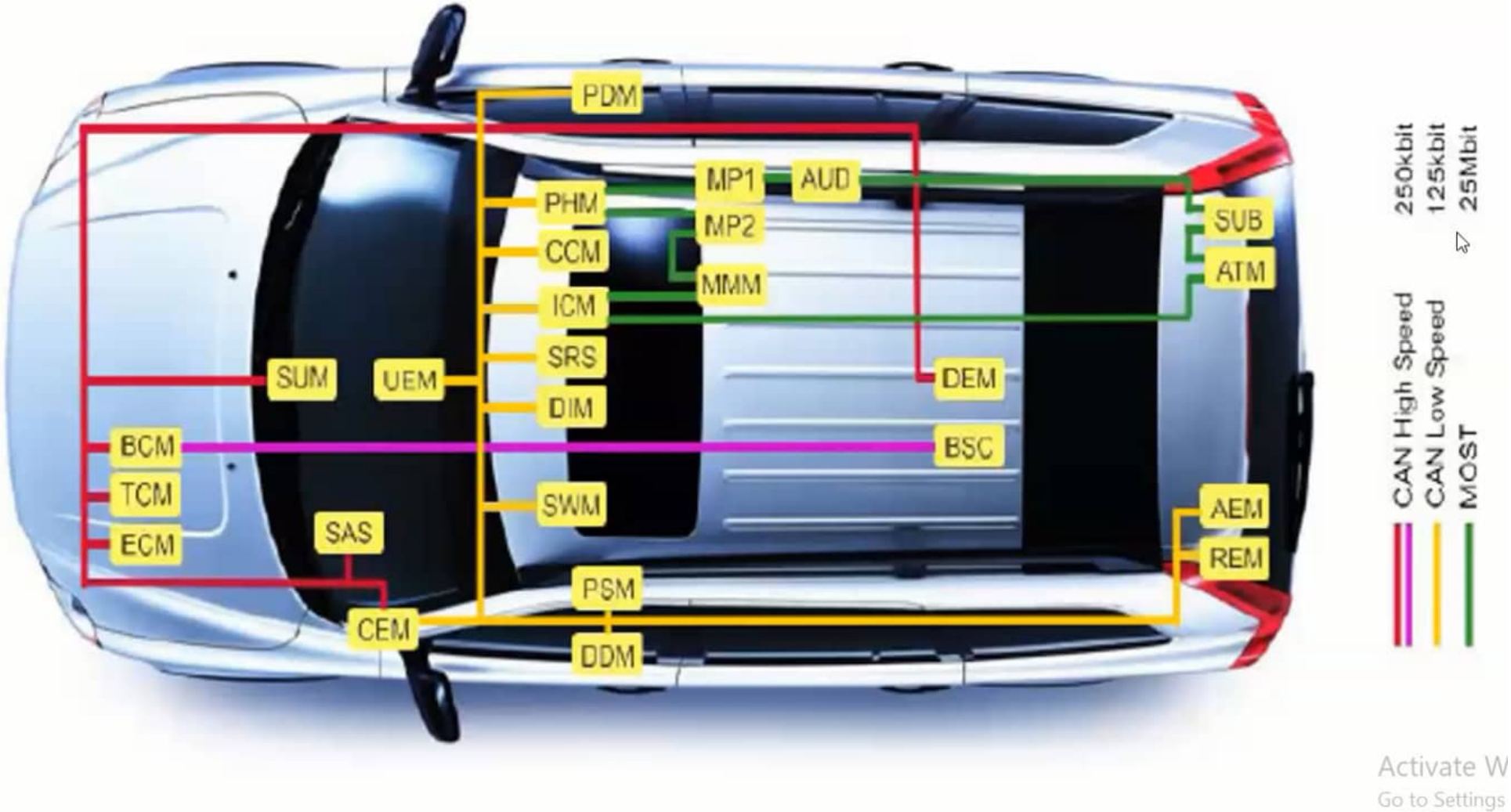
- ATINC629, SwiftNet, SAFEbus – avionics
- WorldFIP, TCN – rail (trains)
- LIN, CAN, TT-CAN, FlexRay – automotive (cars)
- ProfiBus, Profinet, WorldFIP, P-Net, DeviceNet, Ethernet, Industrial Ethernet (IE) – industrial automation
- Firewire, USB, MOST – multimedia

Automotive - VW Phaeton 2003-2016

- 11,136 electrical parts
- 61 ECUs (Electronic Controller Units == CPUs)
- Optical bus for high bandwidth infotainment data
- 35 ECUs connected by 3 CAN-buses sharing
  - 2500 signals
  - In 250 CAN messages

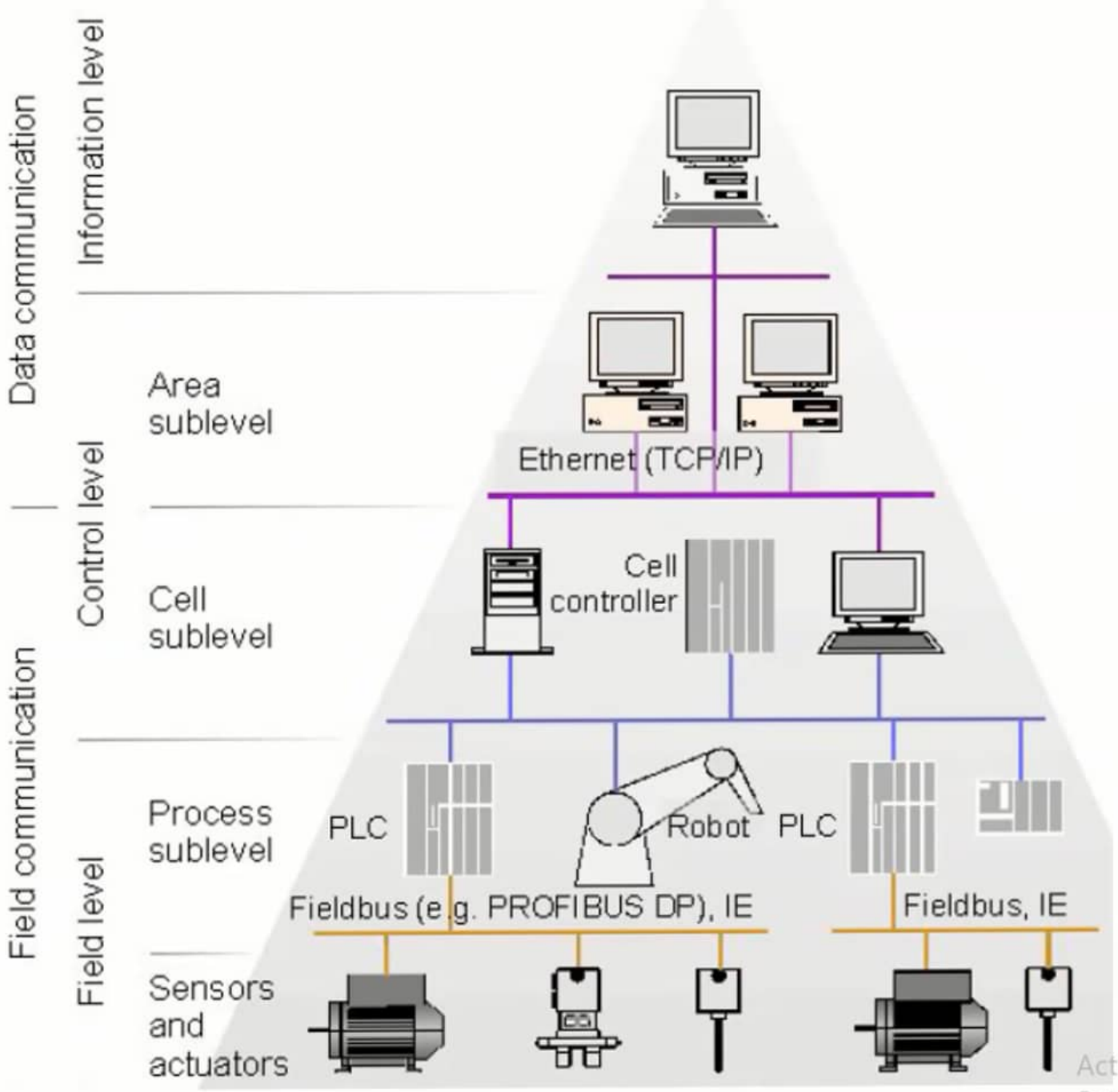
The VW Phaeton

Volvo XC 90 network topology



Industrial Networks:

Let us start with a brief look at Networks for Manufacturing and Process Automation



Hierarchy of an industrial automation system

Industrial Networks:

Industrial network protocols are deployed throughout a typical Industrial Control System (ICS) network architecture spanning wide-area networks, business networks, plant networks, supervisory networks, and fieldbus networks.

Most of the protocols have the ability to perform several functions across multiple network zones, and so are referred to generically as **industrial protocols**.

Industrial protocols are real-time communications protocols, developed to Interconnect the systems, interfaces, and instruments that make up an industrial control system.

Industrial Networks:

Many industrial protocols were designed initially to communicate serially over RS-232/485 physical connections at low speeds (typ. 9.6 kbps to 38.4 kbps), but have since evolved to operate over Ethernet networks (IEEE 802.3) using routable protocols, such as TCP/IP and UDP/IP.

These protocols can be divided into two common categories:

fieldbus and **backend** protocols.

Fieldbus is used to represent a broad category of protocols that are commonly found in process and control: commonly deployed to connect process-connected devices (e.g. sensors) to basic control devices (e.g. programmable logic controller or PLC), and control devices to supervisory systems (e.g. ICS server, human–machine interface or HMI, historian).;

Industrial Networks:

Backend protocols are those protocols that are commonly deployed on or above supervisory networks, and are used to provide efficient system-to-system communication, as opposed to data access.

Examples of backend protocols include connecting a historian to an ICS server, connecting an ICS from one supplier to another supplier's systems, or connecting two ICS operation control centers.

Industrial control systems cyber security

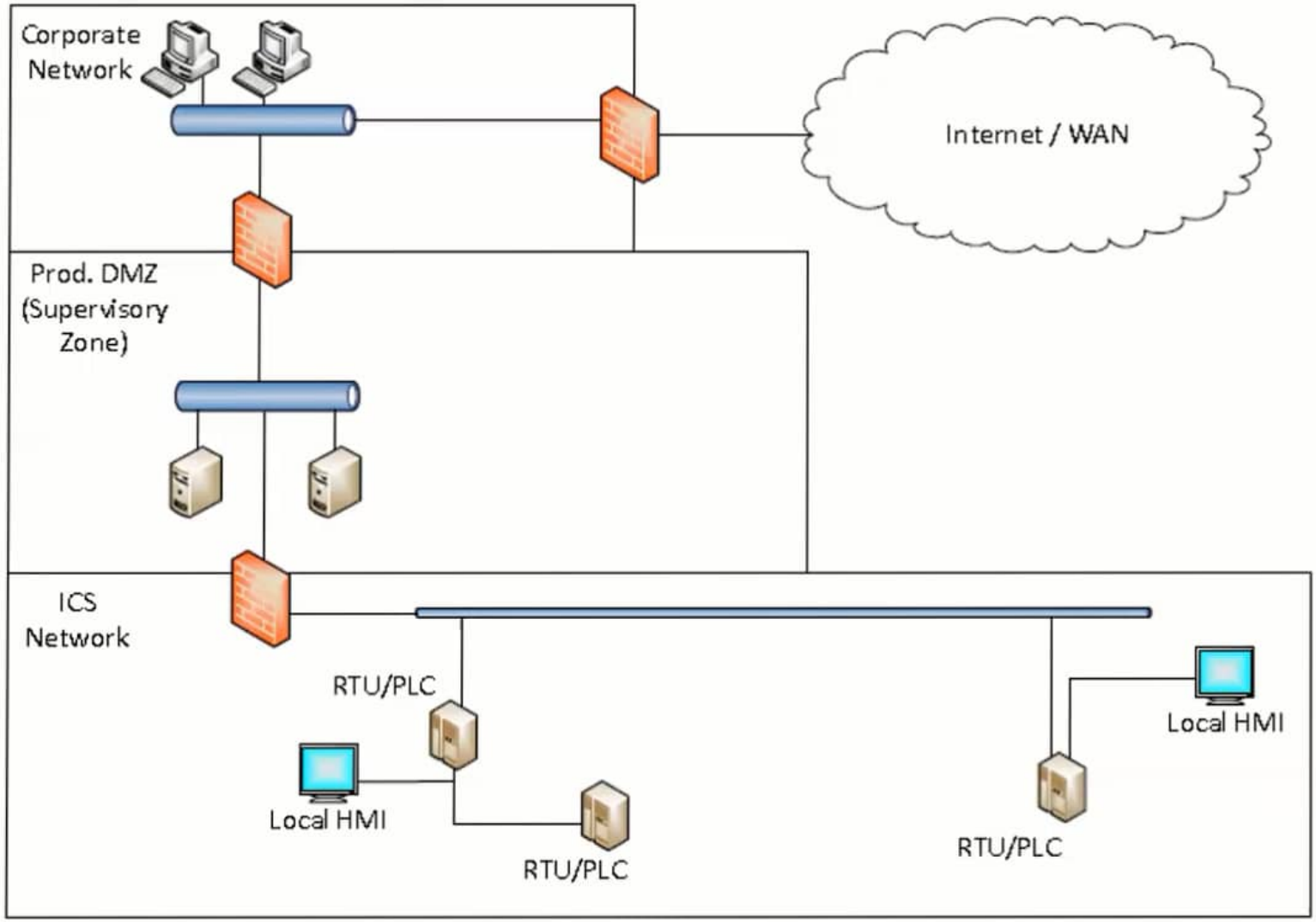


Figure 1. Simplified example network structure of industrial site.

ra off Hence these are also Cyber Physical Systems (CPS)

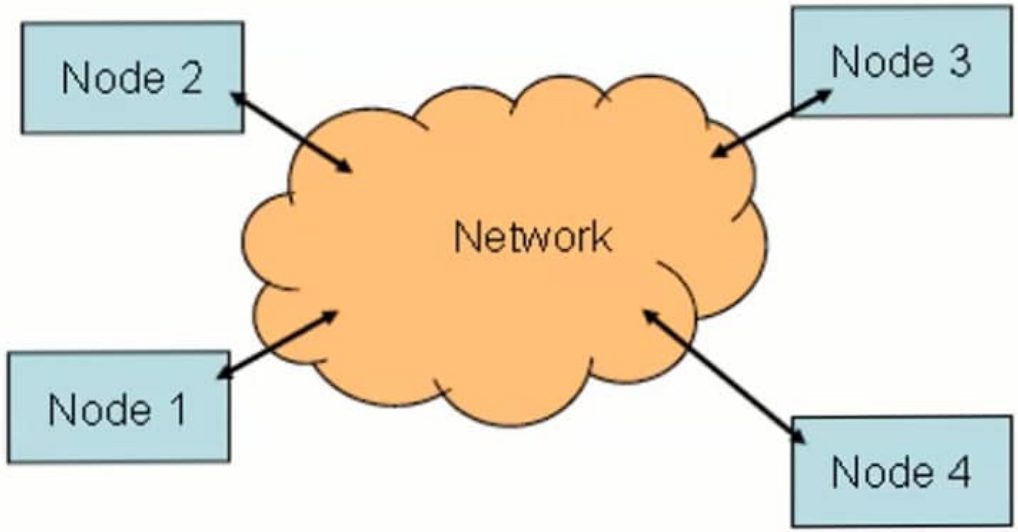
Service requirements

Typical service requirements in real-time networks:

- Efficient transmission of **short data** (few bytes)
- **Periodic** transmission (control, monitoring) with **short periods** (ms), **low latency** and **small jitter**
- Fast transmission of **aperiodic requests** (alarms, commands, …)
- Transmission of **non-real-time data** (configuration information, log data, …)
- Multicasting as well as unicasting (peer to peer)

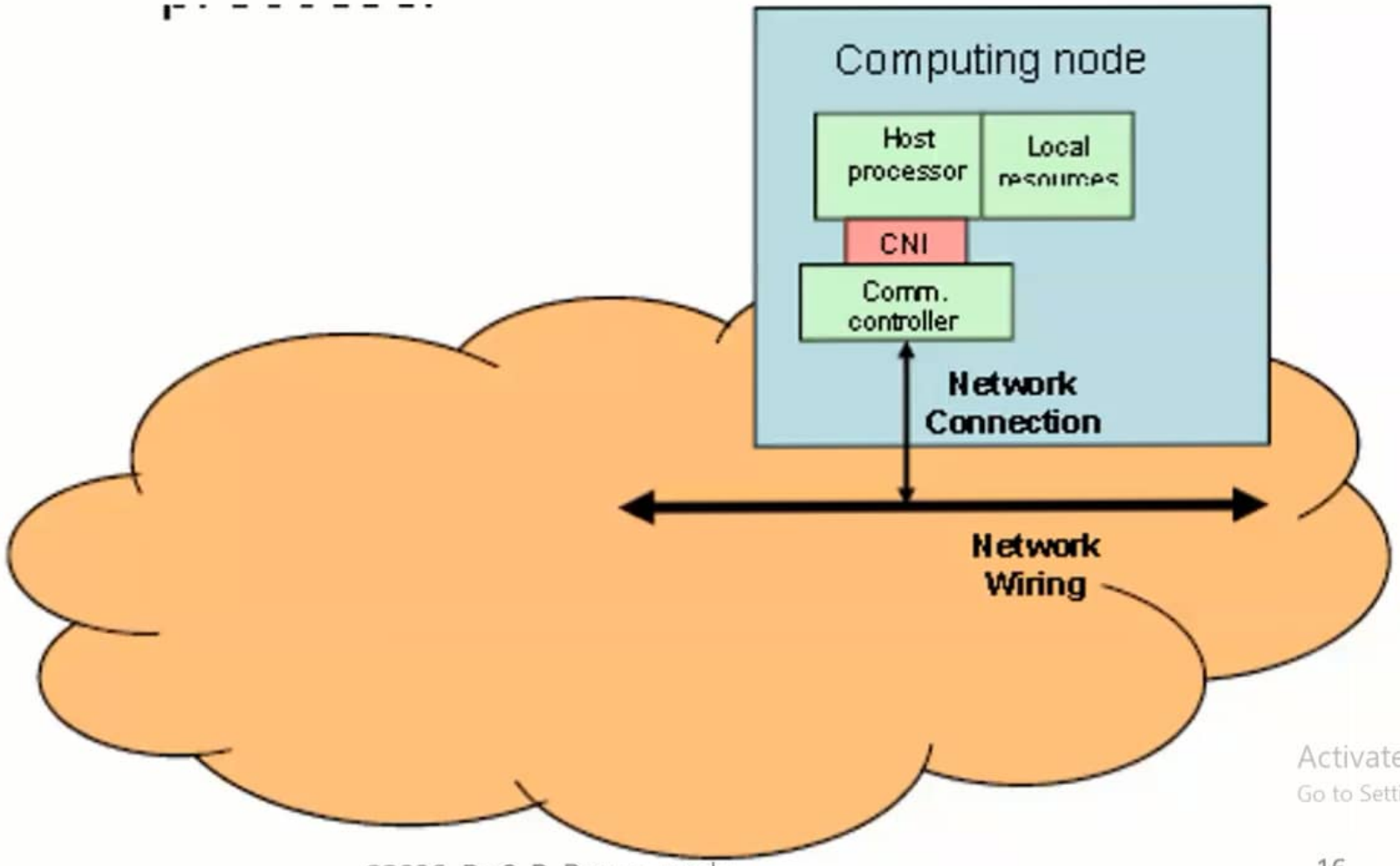
The Network in a Distributed System

- The network is a fundamental component in a distributed system supporting all the interactions among the nodes
- Hence, it is also a critical resource since loss of communication results in the loss of all global system services



Network Interfaces

• The network extends up to the **Communication Network Interface (CNI)** that is the interface between the communication systems and the node host processor.



Messages & Transactions

- Interactions are supported by **message passing**
- A message is a **unit of information** that should be transferred at a given time from a sender to one or more receivers
- Contains both the **data** and the **control information** that is relevant for the proper transmission of the data (e.g., sender, destination, checksum, …)
- A network **transaction** is the sequence of actions within the communication systems required to transfer the message
- Might include messages containing only control information, i.e., **control messages**
- Some networks automatically break large messages into smaller packets (**fragmentation/reassembly**)
- A **packet** is the smallest unit of information that is transmitted
- The **data efficiency** of the network is the ratio between the time to transmit **effective data** bits and the total duration of the transaction

Timing related Measures

- Typical timing related measures concerning the temporal behaviour of the network:

– **Network induced delay** – extra delay caused by the transmission of data over the network. Some applications, e.g., control, are very sensitive to this

– **Delay jitter** – variations in the network induced delay. Some applications, e.g., multimedia streaming, are very sensitive to this, but not so sensitive to the delay.

– **Buffer requirements** – when the instantaneous transmission from a node is larger than the capacity of the network to dispatch, the traffic must be stored in buffers. Too small buffers lead to packet losses

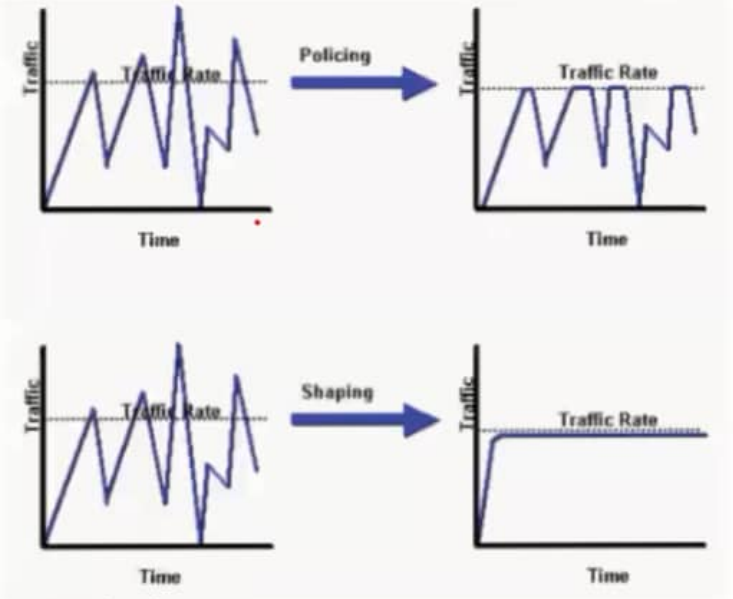
– **Packet loss probability** – packet losses can in addition to the above also be caused by unreliable network media, one example of which is wireless networks.

Timing related Measures, cont.

– **Throughput (bandwidth)** – amount of data, or packets, that the network dispatches per unit of time (bit/s and packet/s)

– **Arrival/Departure rate** – rate at which data arrives at/from the network

– **Burstiness** – measure of the traffic submitted to the network in a short interval of time. Bursts may have a negative impact on the real-time performance of the network and impose high buffering requirements. **Traffic shaping** can be used to control the characteristics of the traffic generated by a node.



Shaping implies the existence of a queue and of sufficient memory to buffer delayed packets, while policing does not.

Queueing is an outbound concept; packets going out an interface get queued and can be shaped.

Only policing can be applied to inbound traffic on an interface.

Real-Time Messages

- Real-time messages can have **event** or **state semantics**
- **Events** are perceived changes in the system state. All events are significant for the state consistency across sender and receiver
- **Event messages** must be queued at the receiver and removed upon reading. **Correct order** in delivery must be enforced.
- **State messages** (containing state data) can be read many times and the values of the previous message concerning the same real-time entity can be overwritten by the new values.

Real-Time Messages

Understanding how industrial networks operate requires a basic understanding of the underlying communications protocols that are used, where they are used, and why.

There are many highly specialized protocols used for industrial automation and control, most of which are designed for efficiency and reliability to support the economic and operational requirements of large industrial control system (ICS) architectures.

Industrial protocols are designed for real-time operation to support precision operations involving deterministic communication of both monitoring and control data.

This means that most industrial protocols forgo any feature or function that is not absolutely necessary for the sake of efficiency.

More unfortunate is that this often includes the absence of even basic security features, such as authentication or encryption, both of which require additional overhead.

Real-Time Messages

To further complicate matters, many of these protocols have been modified to run over Ethernet and Internet Protocol (IP) networks as suppliers moved away from proprietary networks and networking hardware and leveraged commercial off-the-shelf (COTS) technologies.

This, however, has now left these “fragile” protocols potentially vulnerable to cyber-attack.

A Brief Look at Automotive Networks

In an automobile of the present day, the various control modules of a vehicle such as Engine Control Modules, Transmission Control Modules and Body Control Modules etc. usually communicate with each other, in real time, helping in the operation of the vehicle.

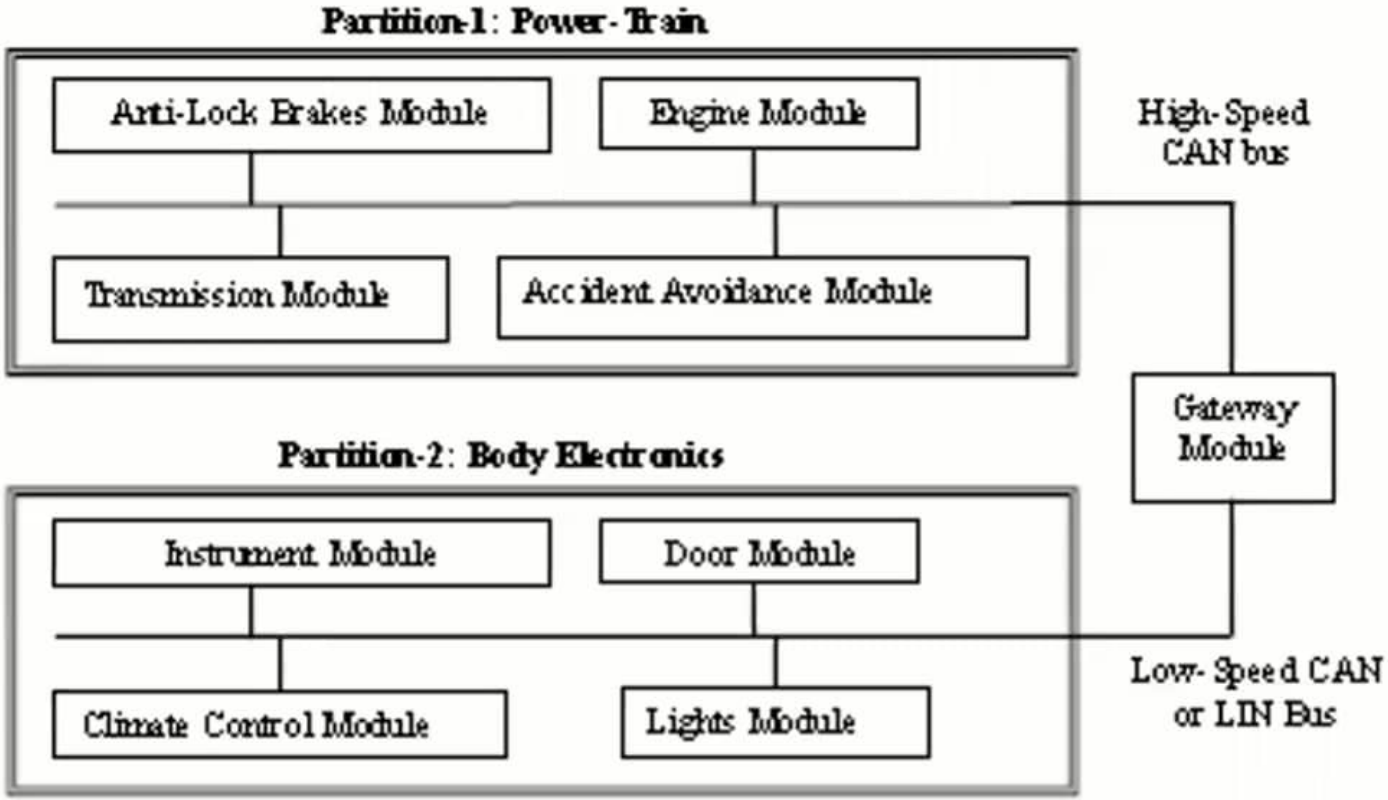
The corresponding number of electronic components is increasing exponentially: the number of electronic sensors and actuators are very high so that point-to-point communication is not possible, the main reasons being the large number of wires needed to connect all the components, non-availability of space and the fact that in case of a failure the fault detection will be extremely difficult.

A Brief Look at Automotive Networks ctd.

All these factors lead to the development of a distributed system in which all the components are connected to various buses and the devices communicate with each other using standardised automotive protocols.
A serial bus can replace all the dedicated point-to-point wiring between modules.

Since different functions of a vehicle need different data rates, such as powertrain needs higher data rate than body electronics, the use of a single serial bus for the entire vehicle may not be the best choice to design the in-vehicle communication network. As high-speed electronic components are more expensive than low-speed electronic components, it would be desirable and cost effective to partition the serial bus into two buses: a high-speed serial bus and a low-speed serial bus.

A Brief Look at Automotive Networks ctd.



An in-vehicle networking system with two partitions

LIN, CAN and FlexRay are some of the protocols used in automobiles today for communicating between the various components.

A gateway is designed to transfer messages between different communication protocols.

A Brief Look at Automotive Networks ctd.

Then there is a need for techniques to simplify the development of applications utilising multiple ECUs.

For this automobile manufacturers are developing the Automotive Open Systems Architecture (AUTOSAR) to allow software components to communicate irrespective of the ECU or bus technology (FlexRay, CAN) that is in use.

In this scenario formal modelling and analysis of communication protocols, or protocol verification is important in system design stages as functional errors discovered during testing and usage are expensive to fix. More importantly in vehicles, unexpected behaviour in communications systems may lead to fatalities.

A Brief Look at Automotive Networks ctd.

- there is no global synchronization among the stations on a CAN network, each station possesses its own local clock

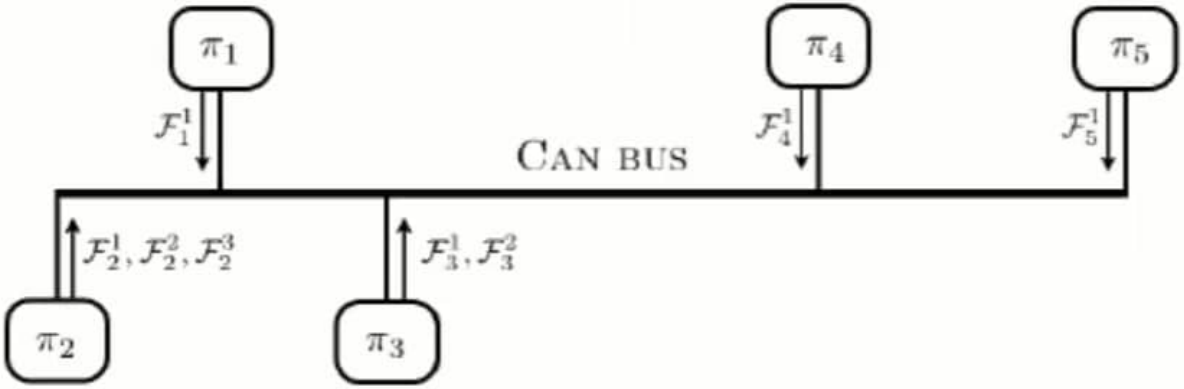


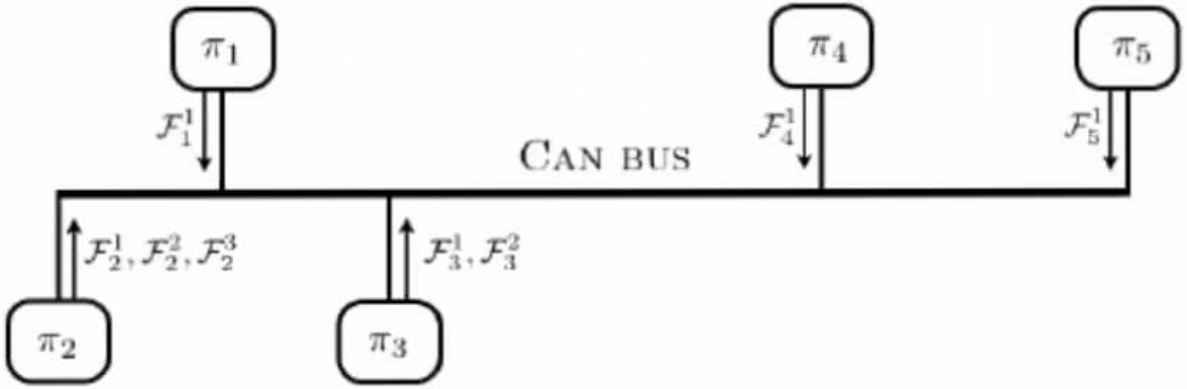
Figure 1. A CAN Communication system where 5 nodes are connected to a CAN bus. $\{\mathcal{F}_1^1, \mathcal{F}_1^2, \mathcal{F}_2^2, \mathcal{F}_3^2, \mathcal{F}_1^3, \mathcal{F}_2^3, \mathcal{F}_1^4, \mathcal{F}_1^5\}$ is the set of frames transmitted on the network, and $\{\mathcal{F}_2^1, \mathcal{F}_2^2, \mathcal{F}_2^3\}$ is the set of frames generated by node π_2.

Typical Design Problem: Reducing Worst Case Response Time: response time of a frame instance is the time elapsed between its activation time and its reception by all the targeted nodes. Accordingly, the worst case response time of a frame is the maximum amongst the response times of all the frame instances.

Typical Solution Strategy: Desynchronizing streams of frames through the means of offsets is a common practice in automotive CAN networks.

This **traffic shaping** strategy is very beneficial in terms of reducing response times especially at high load levels.

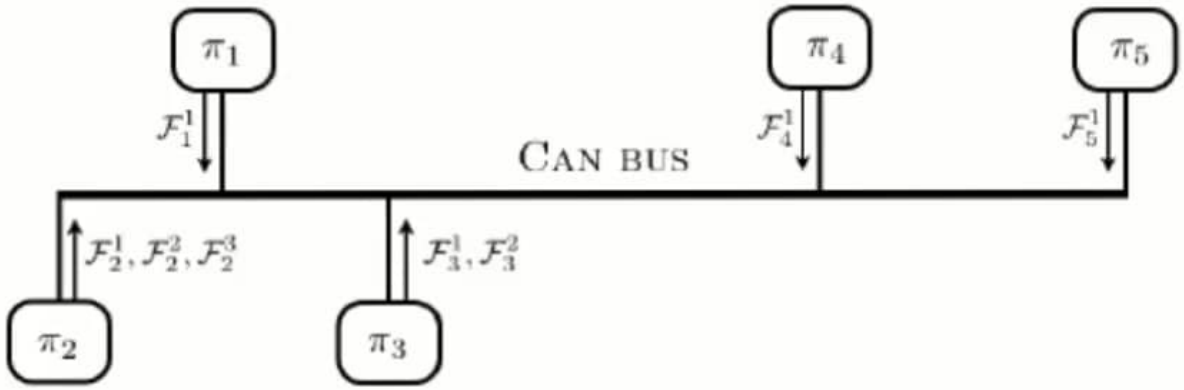
A Brief Look at Automotive Networks ctd.

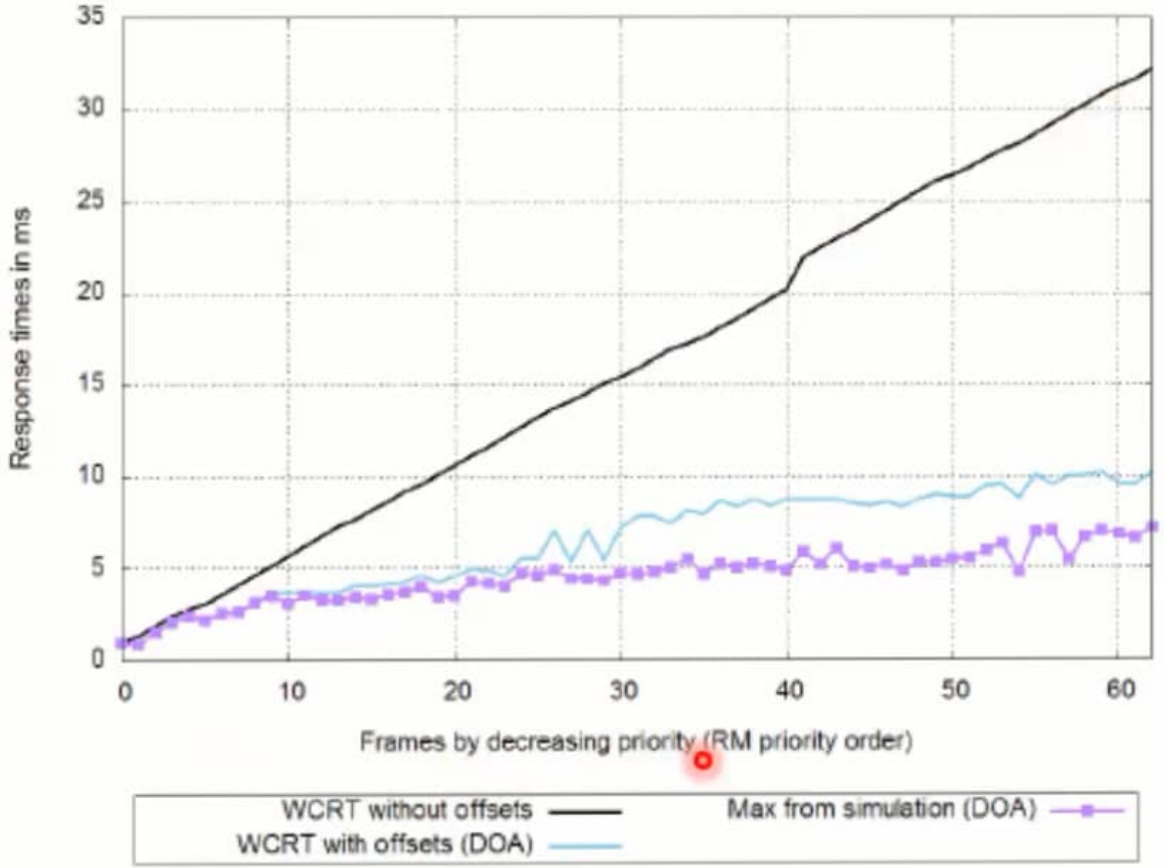


Scheduling frames with offsets. Transmitting frames with offsets means that the first instance of a stream of periodic frames is released with a delay, called the offset, with regard to a reference point which is the first point in time at which the station becomes ready to transmit. Subsequent frames of the streams are then sent periodically, with the first transmission as the time origin. Since there is no global synchronization among the stations on a CAN network, each station possesses its own local clock and the de-synchronization between the streams of frames remain local to each station.

Offsets are efficient because they allow the workload to be spread over time and thus to reduce peak load.

A Brief Look at Automotive Networks ctd.





Example of available tools (commercial)

NETCAR-Analyzer is a timing verification tool for CAN networks based on worst-case schedulability analysis. It enables the designer of CAN networks to optimize the bandwidth usage and make sure that the constraints on communication latencies are met.

Test Scenario used

- Each network studied runs at 250kbit/s and is made up of 8 to 10 nodes.
- The periods of the frames are chosen from the set {20; 50; 100; 200; 500} ms and the size of the data payload in the frames is between 1 and 8 bytes.
- The priority assignment for the CAN identifier is Rate-Monotonic.
- The total load on each network is between 38% and 42%.
- As is often the case in automotive networks, they have assumed that there is one station (e.g., a gateway) that generates more load than the others. Here this station transmits 20% of the total traffic.
- The offset assignment algorithm that is used in this study is Dissimilar Offset Assignment (DOA).
- The experiments were done with and without frame queuing jitter; in the former case 10% of the frames are assigned a random jitter in emission less than their period (integer values in ms).

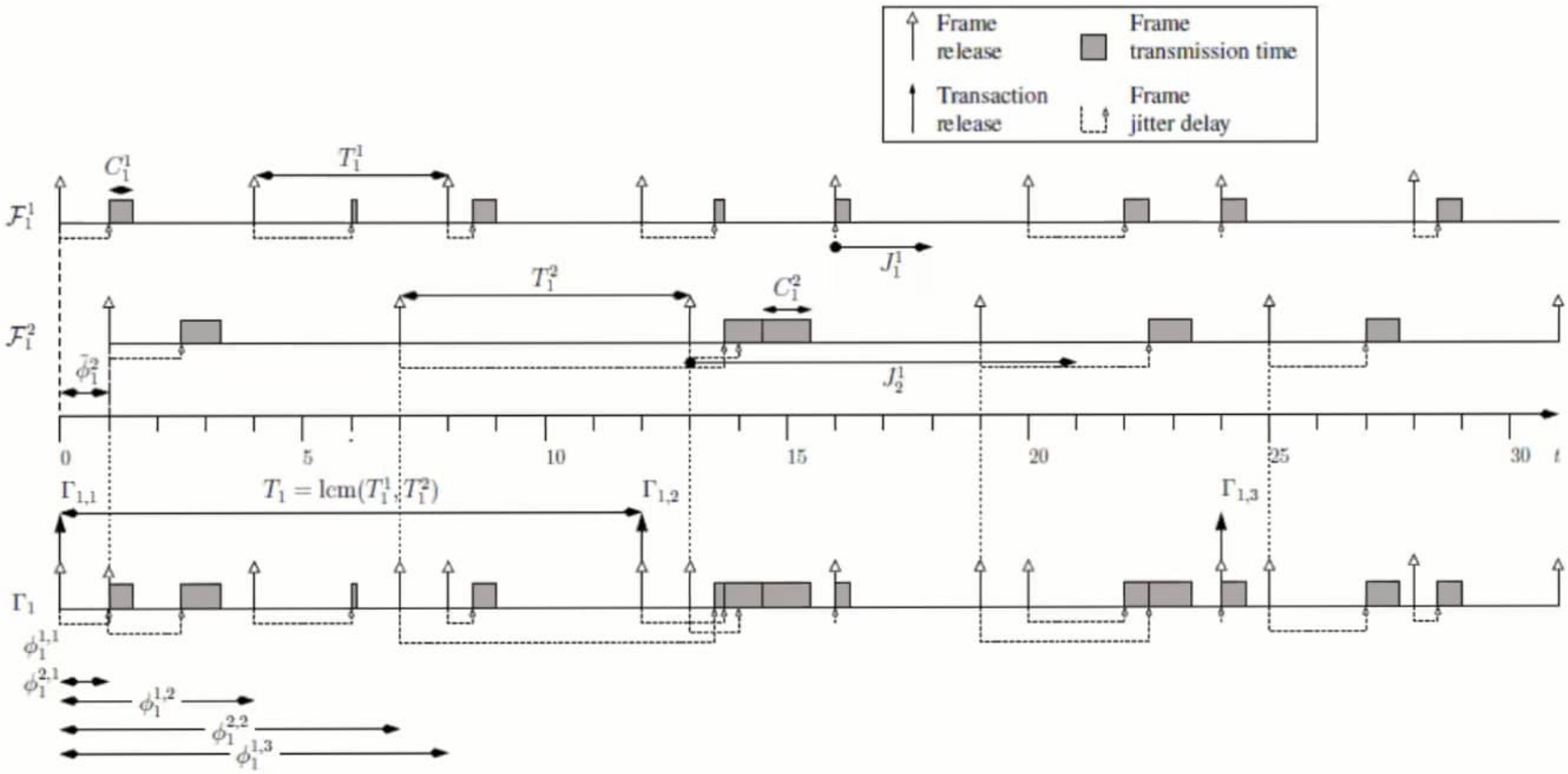


Figure 2. Workload induced by frames $\mathcal{F}_1^1$ and $\mathcal{F}_1^2$ generated by node π_1, and mapping of these frames into a single transaction Γ_1. The periods of frames $\mathcal{F}_1^1$ and $\mathcal{F}_1^2$ are respectively 4 and 6, the maximum jitters are 2 and 7 respectively, and the offsets are 0 and 1 respectively.

Reference: Yomsi, et al (2012), Controller Area Network (CAN): Response Time Analysis with Offsets
Article · May 2012; DOI: 10.1109/WFCS.2012.6242539

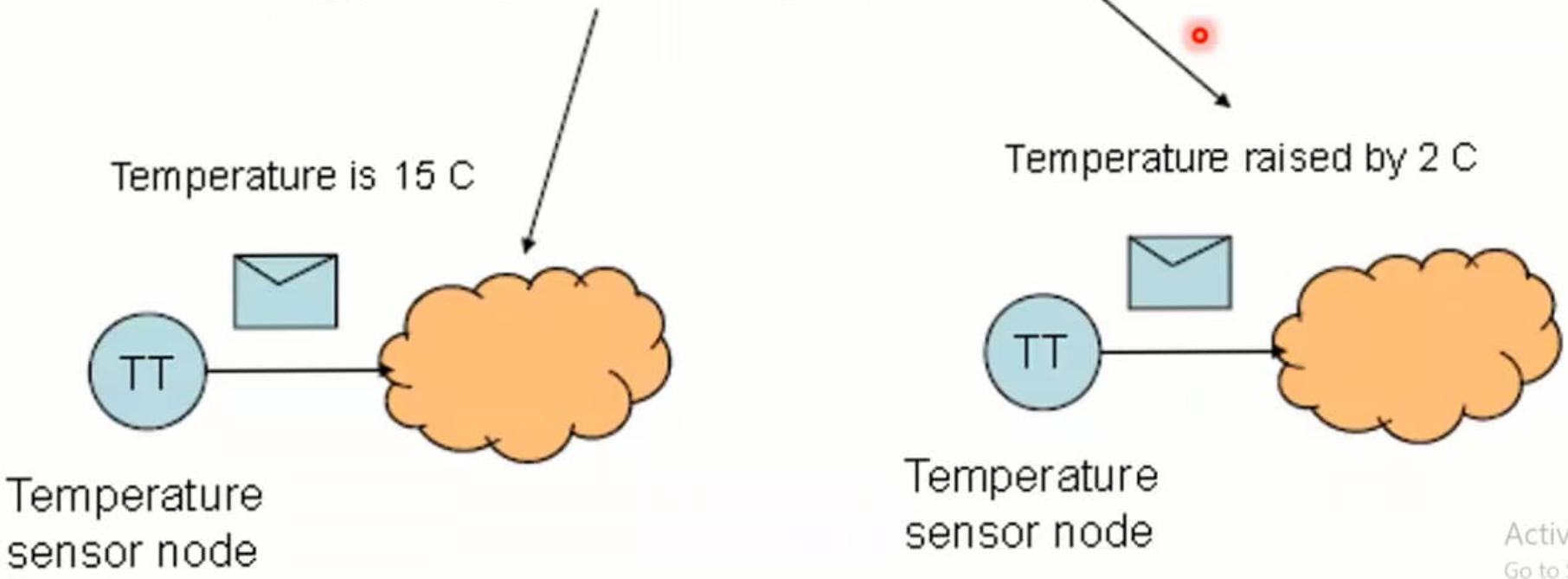
Event or Time Triggering

• According to the type of message (event or state) conveyed by the network, it can be

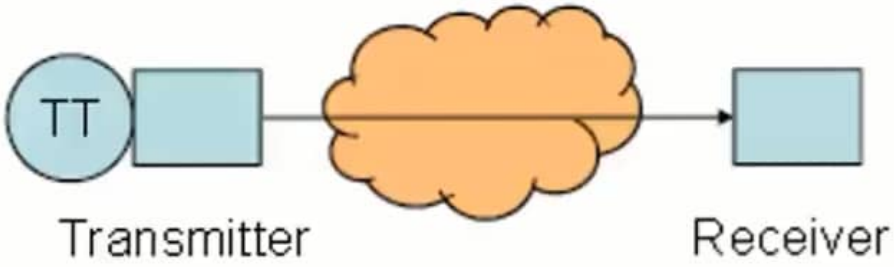
– **Event-triggered (event messages)**

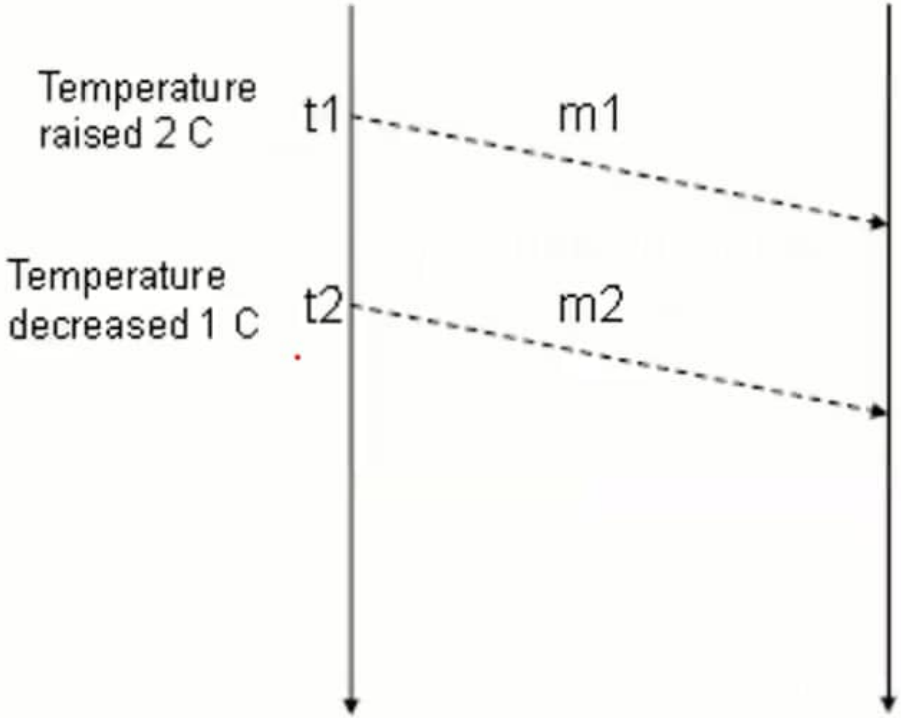
or

– **Time-triggered (state messages)**



Event-Triggered Network





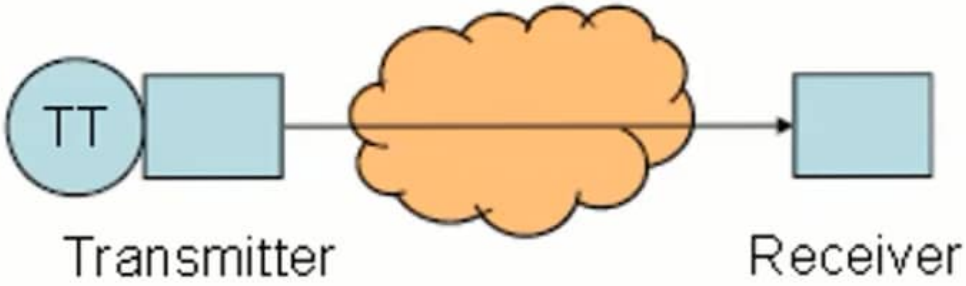
What if m1 is lost?

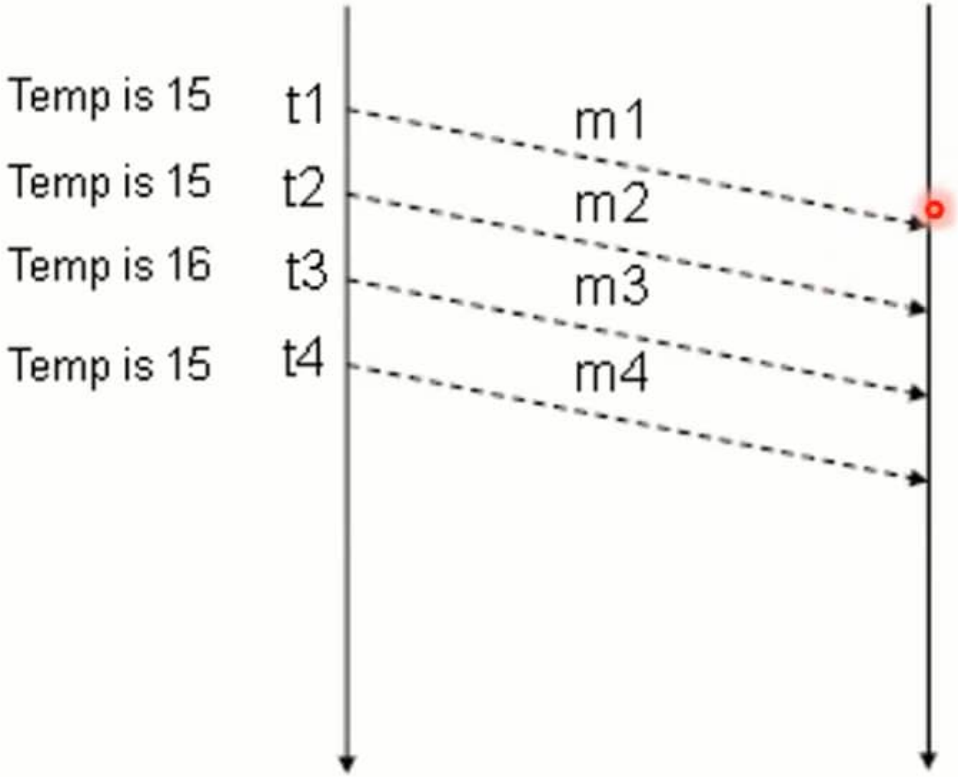
Lost for ever!

Time-triggered Network

- There is a notion of **network time**
  - All clocks are globally synchronized
- Transactions carrying **state data** are triggered at **predefined time instants**
- Receivers have a **periodic refresh** of the system state
- The submitted communication load is well determined

Time-Triggered Network





Loosing a message gives inconsistency only until next message arrives

Event vs Time Triggering

Time-triggered networks:

- Are more deterministic
  - Transmission instants are predefined
  - Fault-tolerance mechanisms are easier to design
- Are less flexible in reacting to errors
  - Retransmissions are often not possible because the traffic schedule is fixed
  - A lost message is not recovered until the next period of the message stream
- Are less flexible with respect to changes
  - Everything must be known a priori and very little can be changed dynamically (cp. static cyclic CPU scheduling)
- The communication protocols are often quite complex

Event vs Time Triggering ctd.

Event-Triggered Networks:

- Low level of determinism
  - Events can occur at any time
- More complex fault-tolerance schemes
- Very flexible with respect to errors
  - Retransmissions can be carried out immediately
- The communication protocols are normally quite simple

An illustrative comparison

- **A TT-network**
  - **The train system!**
    - You can optimize your schedule to be there just in time
    - The train will go anyway at the predefined time!
    - If the train is delayed you may lose a connection
    - Schedule of trains optimized for good use of the track
    - But how to optimize the time to travel for many people and with hops?
    - If you do not synchronize with the train you may have to wait for the next…

- **An ET-network**
  - **The roads system with private cars!**
    - You go when you want
    - Might take less time if there is few traffic
    - But strong congestions can occur!…

Deciding Factors

In real-time processing the urgency of messages to be exchanged over the network can differ greatly: a rapidly changing value (e.g. engine load) has to be transmitted more frequently and therefore with less delays than other values (e.g. engine temperature) which change relatively slowly.

Hence, the priority at which a message is transmitted compared with another less urgent message could be specified beforehand.

Another way of looking at situations is as follows: in formulating the requirements of distributed control systems two different situations should be distinguished:

1.) Occurrence of critical situations, like for instance, the exceeding of a critical temperature, and
2.) the regular operation of the control system.

The former situation requires an "as fast as possible" reaction to the asynchronous event in order to start emergency mechanisms.

Hence it is necessary to compare event- and time-triggered communication in relation to their ability to react to asynchronous events.

Comparison of ET and TT-networks in relation to their Reaction to Asynchronous External Events

Test scenario: cooperative communication between two control units A and B (figure 1)

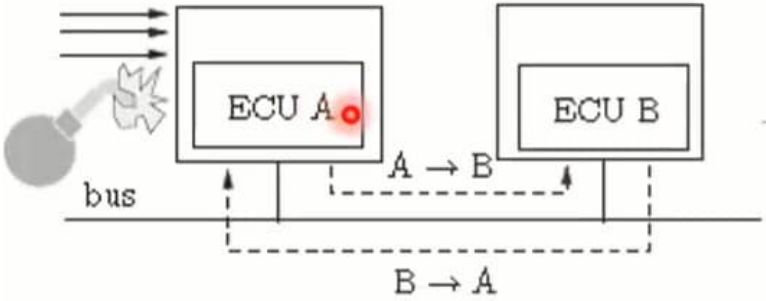


Figure 1

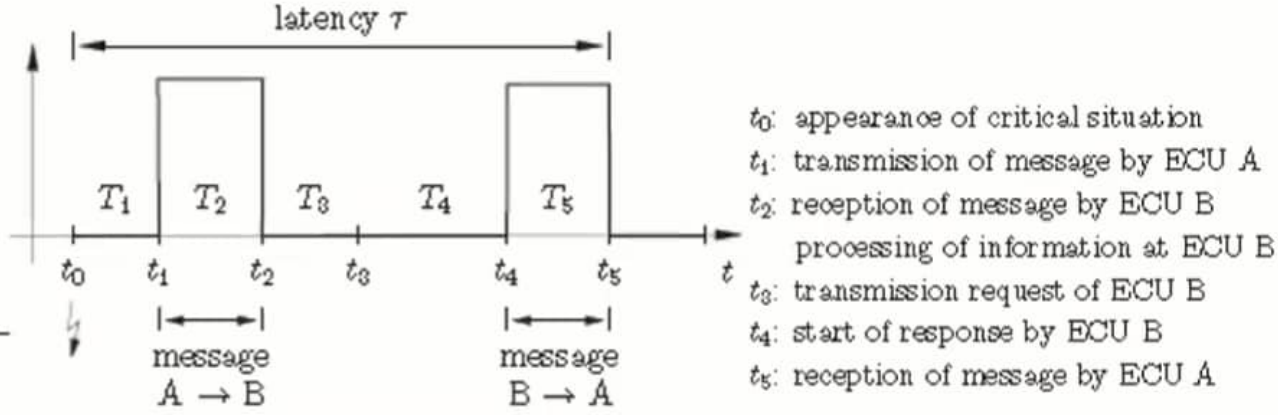
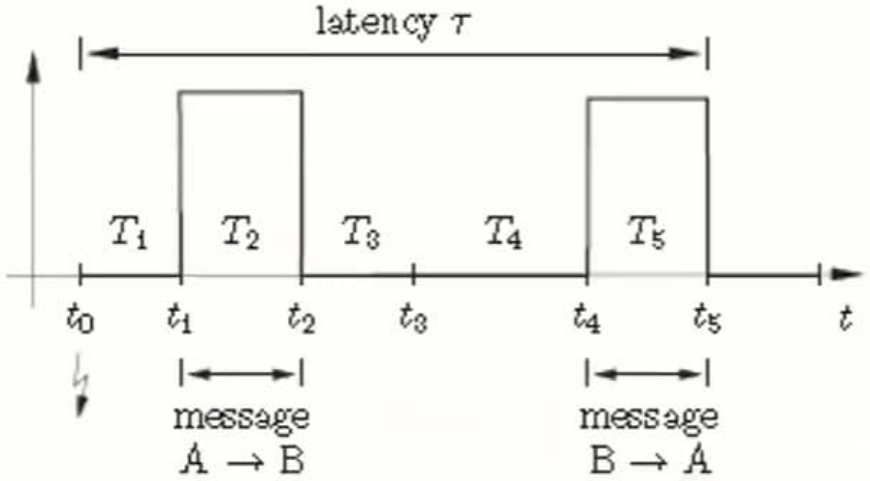


Figure 2

A critical situation occurs at an arbitrary instant of time and corresponding sensor signals reach control unit A (ECU A). Now ECU A informs ECU B about the critical situation and waits for its reply. Thus in total the cycle A -> B -> A is examined. Figure 2 illustrates the explained scenario along the timeline.

Reasons for the latency $T1 = t1 - t0$ are manifold. On the one hand there is a time demand for the computation at ECU A. On the other hand one has to wait for the permission to access the bus.

Comparison of ET and TT-networks: Test scenario ctd.



The duration *T*2 is the transmission time on the bus which depends on the data rate and the length of the message.

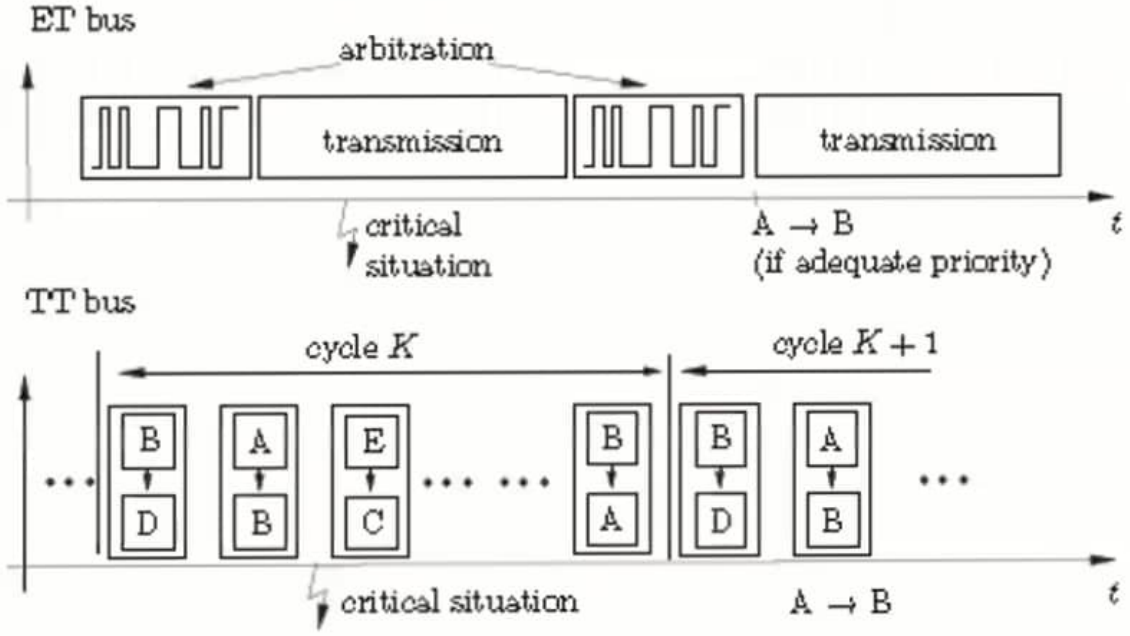
The reception of the message by ECU B is finished at *t*2. The information processing takes place until *t*3. Afterwards, a response for A is required. The permission to access the bus is received at *t*4. The scenario ends at *t*5 when ECU A receives the response from ECU B. Since external (environmental) events occur asynchronously, the time at which the critical situation appears is not known in advance. Therefore, *T*1 and *T*4 are quite susceptible for jitter.

Comparison of ET and TT-networks: Test scenario ctd.

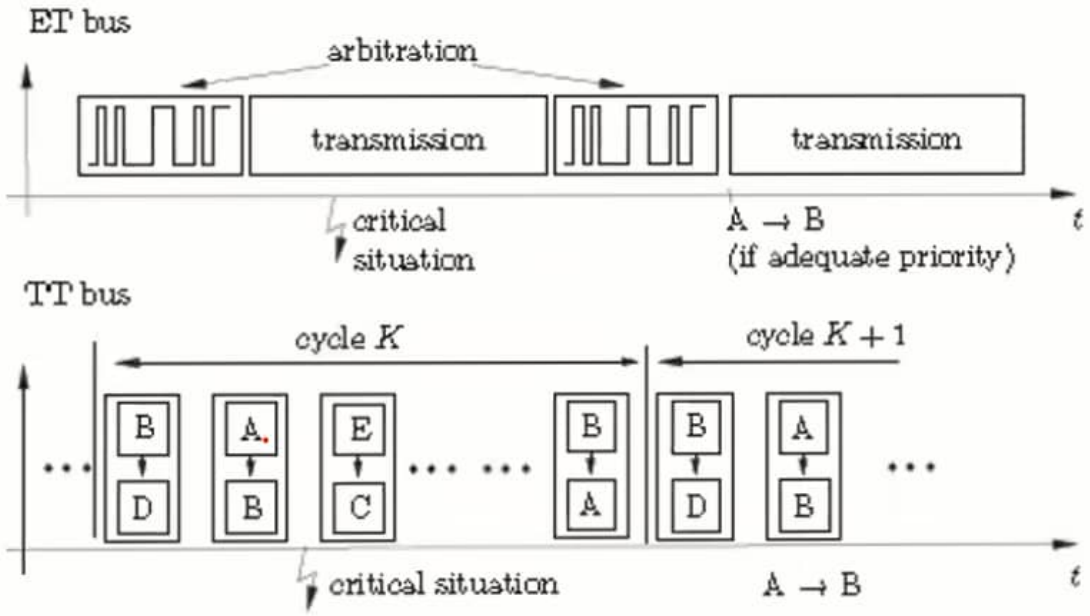
the overall latency τ is composed of the latencies $T1$, …, $T5$. Particularly, the latencies $T1$ and $T4$ depend on the bus concept. Therefore, a more detailed examination is carried out for $T1$ and $T4$.

Figure 3 illustrates qualitatively the behavior along the timeline for an event-triggered and a time-triggered bus concept.

Figure 3



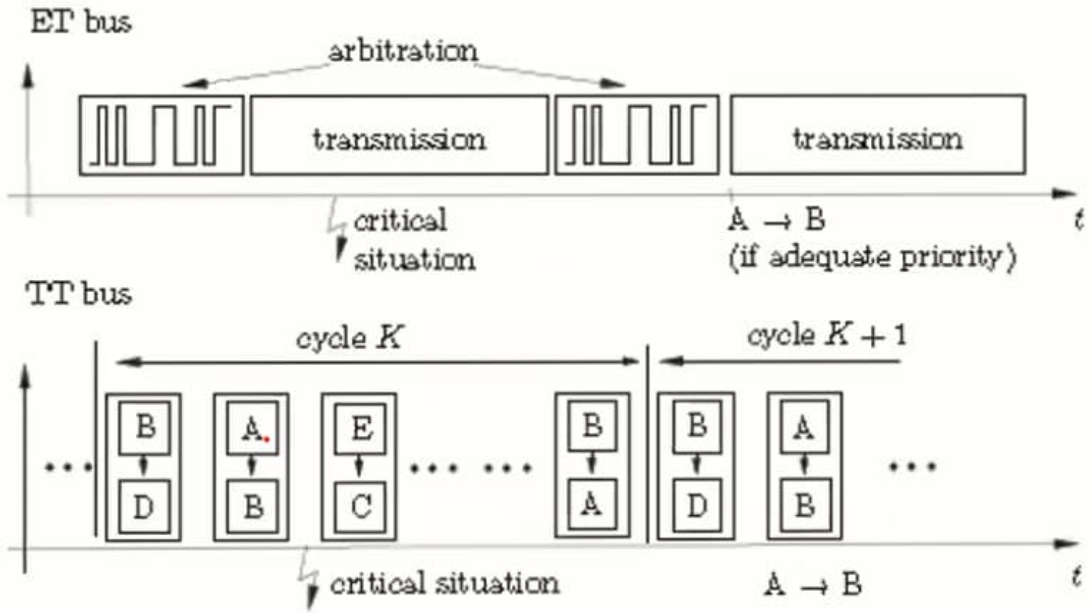
Comparison of ET and TT-networks: Test scenario ctd.



The upper part shows a situation for CAN (an ET-bus), where the critical situation occurs at an arbitrary instant of time. Particularly, the bus can be occupied if a transmission is currently in progress. Then ECU A has to wait for the next arbitration and receives in the best case the permission to the bus in this next arbitration. The cycle A -> B -> A then starts. For this situation to happen it is assumed that the message of ECU A possesses the highest priority compared to all other messages during the arbitration. Summarising, the following ifactors are of importance:

- *bus work load and message priority*
- *maximum length of message and data rate*

Comparison of ET and TT-networks: Test scenario ctd.



The lower part of figure 3 shows the qualitative behavior of a time-triggered bus (FlexRay or TTCAN, for instance) when reacting to an asynchronous event. Here the time instant at which the message is transmitted in the cycle is well defined. In the worst case, after the occurrence of the critical situation one has to wait an entire cycle, if the respective time slot has just passed. After this idle time it is guaranteed that the transmission will take place. Therefore, for the inspected scenario at least a guaranteed upper bound can be given. Summarising, the following factors are of importance:

- *cycle structure, cycle time*
- *position and counts within cycle*
- *data rate*

Comparison of ET and TT-networks: Test scenario ctd.

To evaluate the real-time performance on the basis of the scenario above, usually three questions arise:

1.) Does the system react to all critical situations?

2.) Of what magnitude is the average delay τ of the system?

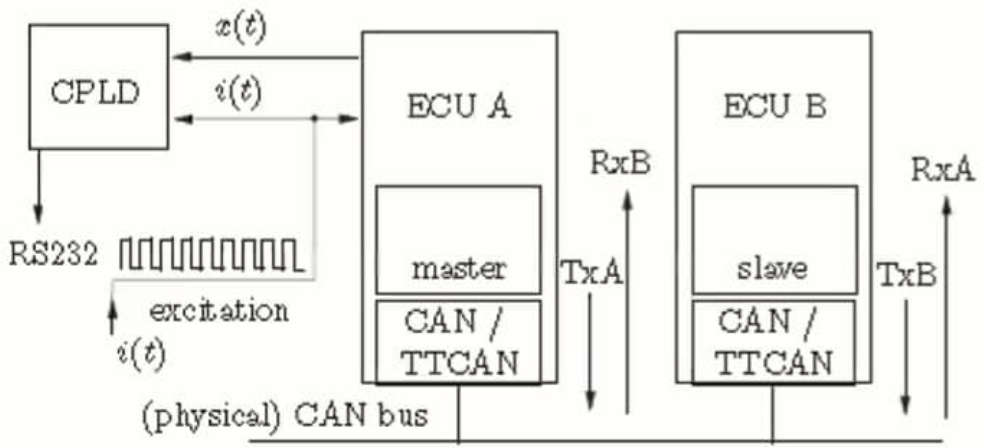
3.) How reliable is the system's response? I.e., of what magnitude is the jitter?

To all three questions, the notion of *'Distinctness of Reaction'* (*DoR*) has been developed to give a quantitative answer. The method is based on an orthogonal Walsh correlation and yields a reliability measure given by the average latency response time and the jitter when reacting to asynchronous external events.

The underlying logic is that a CPS performing time-critical tasks could be described by a frequency response locus like an electronic device which reacts with a delay to a periodical external signal.

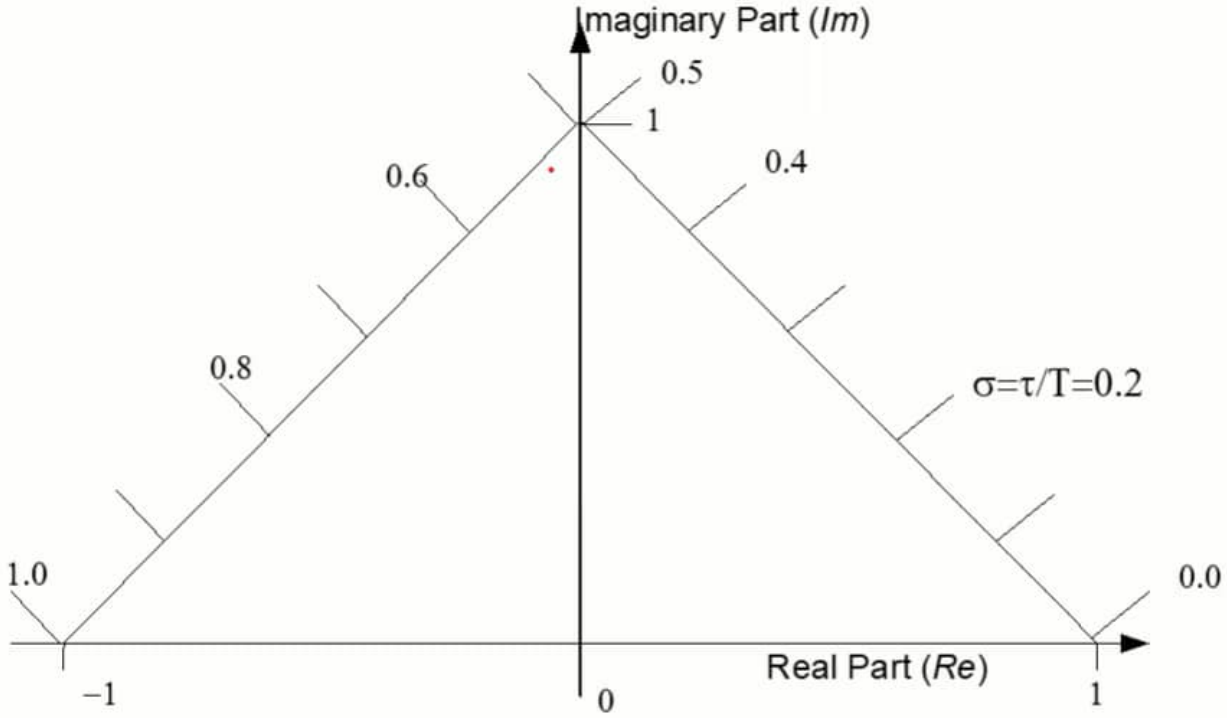
Comparison of ET and TT-networks: Test scenario ctd.

In order to measure the *DoR*, the communication system is excited by a square wave signal $i(t)$ with an adjustable frequency as shown in figure 4. This excitation simulates the occurrence of the critical situation. After the described cycle A -> B -> A the system reacts in a predefined manner with its response $x(t)$. The signals $i(t)$ and $x(t)$ can be processed by a digital circuit, implemented on a CPLD (Complex Programmable Logic Device) which allows to quantify the *DoR*. The *DoR* is a measure for the jitter in the system's response and takes on values from 100% (no jitter) to 0% (at least sporadic loss of excitations).



The *DoR* determines the amplitude response part of the frequency response and the phase response part is determined by the average response time τ which is a direct measure of the average latency of the system's response.

Frequency Response locus of the Walsh Correlation for Different Frequencies for an ideal system with constant latency



Frequency Response locus of *DoR* for an ideal system with constant latency τ

For an ideal system that responds to every pulse with a constant delay τ is shown above. This can be used to estimate the *DoR* by observing that for the frequency f = 0.5/τ, Re(DoR) = 0 and by delaying the incoming signal for a certain *dead time* T_d before it is fed into an input of the test system that point shifts to f = 0.5/(τ + T_d). Thus *DoR* is estimated by adjusting the right frequency for a chosen T_d.

Definition of *'Distinctness of Reaction'* (*DoR)*

Definition of *DoR*

$$DoR = \lim_{n\to\infty} \frac{1}{n*T} \int_0^{n*T} x(t)w(t-\tau)dt \qquad (1)$$

Where

$x(t)$ is the output signal of the network analysed,

$w(t-\tau)$ is the Walsh fundamental wave: the signal used to excite the test system, only delayed by a time τ (TT or ET network segment here),

τ is the average time taken for the cycle described cycle A -> B -> A (average latency) and

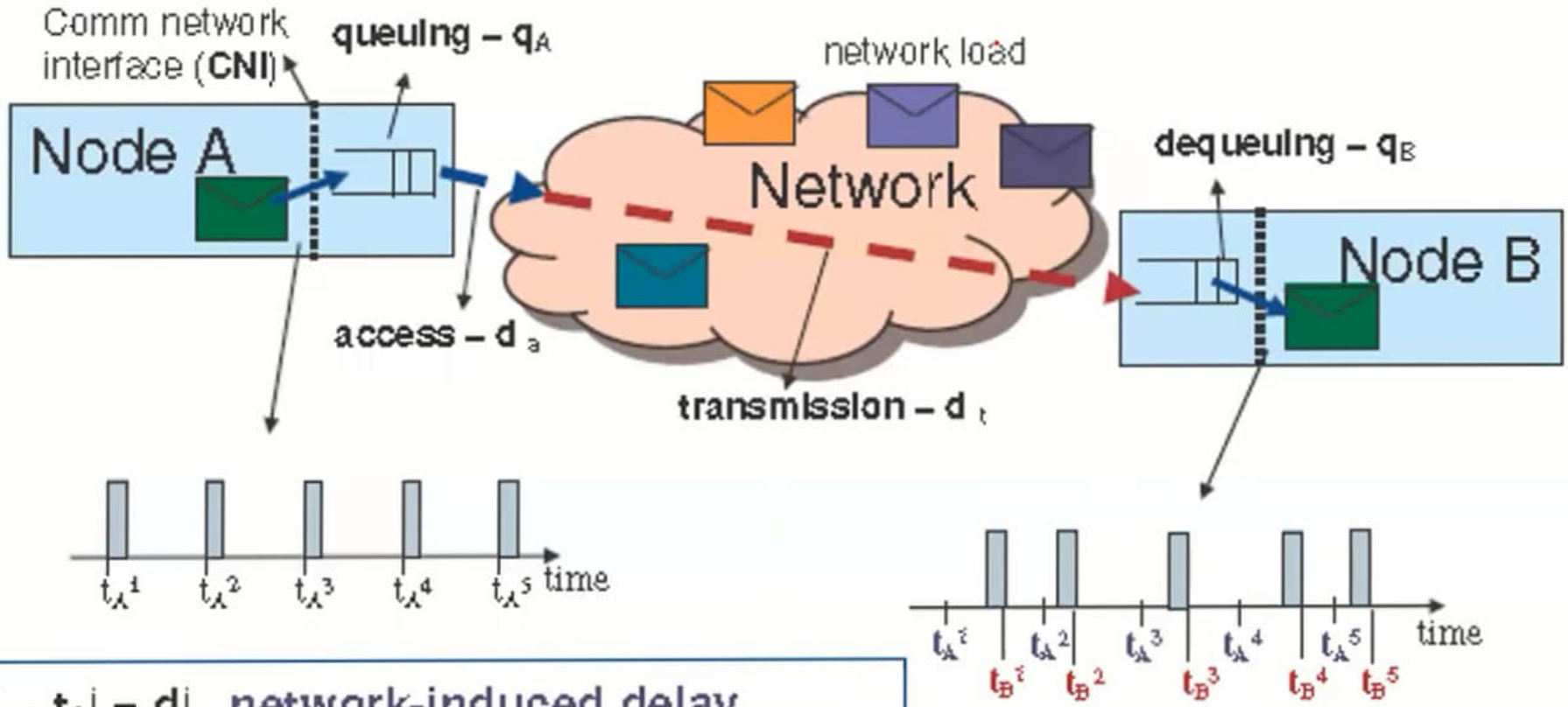
T is the period of the input pulse (input frequency, $f = 1/T$).

For an ideal network without jitter, $x(t)$ equals $w(t-\tau)$, so the *DoR* will be 1 in this case. For all other system it will take a value between 0 and 1.

Any CPS has a cut-off frequency, which means that it cannot react to frequencies above. So the input pulse must have a frequency below this cut-off value.

Theory and experimental setups for computing average latency τ,and DoR have been discussed in recent research literature. However these details are quite involved and go out of the scope of the present course.

Network delay and delay jitter



$t_B^i - t_A^i = d^i$, network-induced delay

$d^i = q_A^i + d_a^i + d_t^i + q_B^i$, delay components

$d^i - d^{i-1} = j^i$, delay jitter

Reception instants may suffer irregular delays due to interferences from the network load, queuing policies and processor load

Comparison of ET and TT-networks In Summary

It suffices to summarise here that event-triggered bus concepts are more efficient for small bus loads, since they generate lower latencies and less jitter when reacting to asynchronous events. Then, a bus load or a load on the micro controller can arbitrarily worsen the behaviour of the CAN bus (which is event-triggered).

For time-triggered buses the result is almost independent of the actual load and the *DoR* as well as the skew (defined as 'average skew' $s = -\tau/T$) show a linear characteristic with respect to the frequency of the excitation, f. Thus in a sense, time-triggered buses (FlexRay and TTCAN) are deterministic (compared to CAN), since the limit of the (worst case) time behaviour can easily be determined in advance.