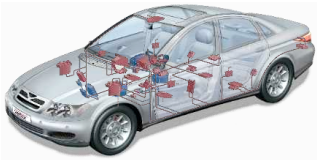


Brief tour of Real-Time Embedded Networks

Luís Almeida

lda@det.ua.pt

Electronic Systems Lab-IEETA / DET
Universidade de Aveiro
Aveiro, Portugal

Course Aims

- ✓ Briefly analyze the concepts behind the techniques and protocols to support **timeliness** in the **network**
- ✓ Analyze the network impact on **Distributed Computer Control Systems**
- ✓ Focuses on **Embedded Control**

Course Outline

- ✓ **Basic concepts of communication**
- ✓ **Temporal control of the communication**

Bibliography

- Richard Zurawski (ed). ***The Industrial Communication Systems Handbook***. CRC Press, 2005.
  - P. Pedreiras, L. Almeida. **Approaches to Enforce Real-Time Behavior in Ethernet**.
- B. Bouyssounouse, J. Sifakis (eds.) ***Embedded Systems Design, The ARTIST Roadmap for Research and Development.*** LNCS 3436, Springer 2005.
- P. Veríssimo, L. Rodrigues. ***Distributed Systems for System Architects***. Kluwer Academic Publishers, 2001.
- J. Liu. ***Real-Time Systems***. Prentice-Hall, 2000.
- Krishna and Shin. ***Real-Time Systems***. McGraw Hill, 1997.
- Kopetz H.. ***Real-Time Systems: Design Principles for Distributed Embedded Applications***. Kluwer Academic Publishers, 1997**.**

Other suggested reading

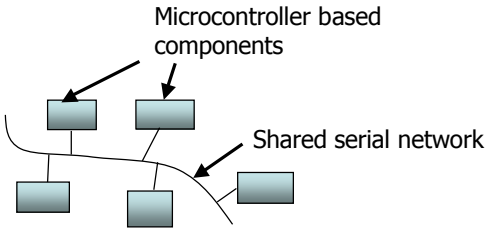
- O. Redell, J. Elkhoury, M. Törngren. **The AIDA tool-set for design and implementation analysis of distributed real-time control systems**. Microprocessors and Microsystems, 28(4):163-182, May 2004.
- P. Koopman. **Critical Embedded Automotive Networks.** *IEEE Micro*, IEEE Press, July/August 2002.
- Thomesse J.-P.. **A Review of the Fieldbuses.** *Annual Reviews in Control*, 22:35-45, 1998.
- M. Törngren. **Fundamentals of implementing Real-time Control applications in Distributed Computer Systems**. *Journal of Real-time Systems,* 14:219-250. Kluwer Academic Publishers, 1998.
- Malcolm N. and W. Zhao. **Hard Real-Time Communication in Multiple-Access Networks.** *Journal of Real-Time Systems*, 8(1): 35-78, 1995.
- Tindell K., A. Burns and J. Wellings. **Analysis of Hard Real-Time Communication.** *The Journal of Real-Time Systems.* 9:147-171, Kluwer Academic Press. 1995.
- Cruz R.L., **A calculus for network delay, part i: Network elements in isolation**. *IEEE Trans. Information Theory*, 37(1):114:131, January 1991.
- Cruz R.L., **A calculus for network delay, part ii: Network analysis**. *IEEE Trans. Inform. Theory, 37(1):*132:141, January 1991.

Part 1
Basic concepts of communication

Towards distribution

Nowadays, complex embedded systems are **distributed**, with a **network** connecting several active components

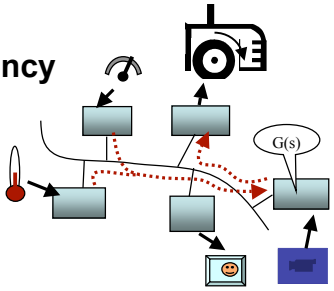
- ✓ Cars, trains, planes, industrial machinery ...



Distribution and control

Also, the number of **automatic control loops** within such systems **increased dramatically** motivated by

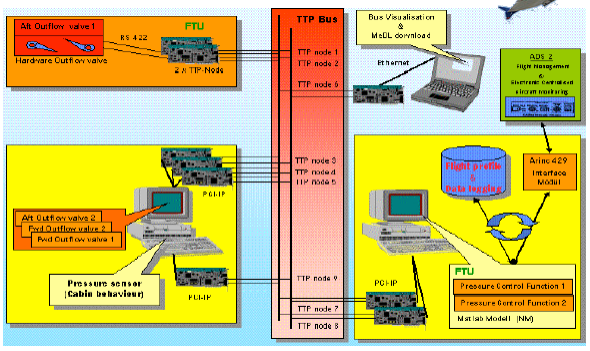
- ✓ Increased **safety**
- ✓ Increased **energy efficiency**
- ✓ New **functionality**
- ✓ More **confort**
- ✓ ...



These are called

Distributed Computer Control Systems (DCCS)

Avionics



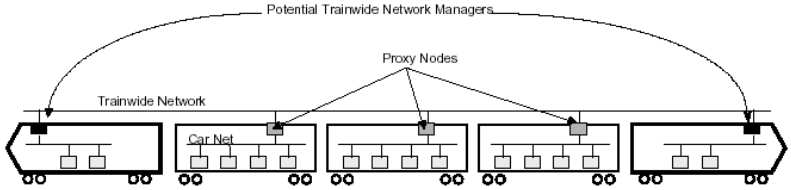
the SETTA project

(proposed using TTP/C in the Airbus A380)

Train control

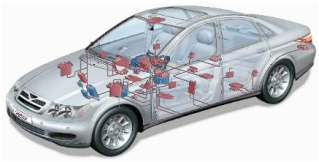
Fast trains (Thalys, Korean project)

Automatic trains (Frankfurt Airport)

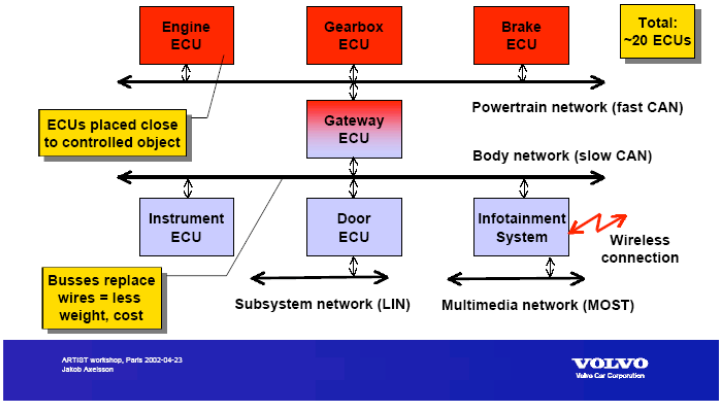


(LONworks applications)

Automotive



Typical Vehicle Electronic Architecture



Automotive

VW Phaeton:

→ **11.136 electrical** parts in total

→ **61 ECUs** in total

→ external diagnosis for 31 ECUs via serial communication

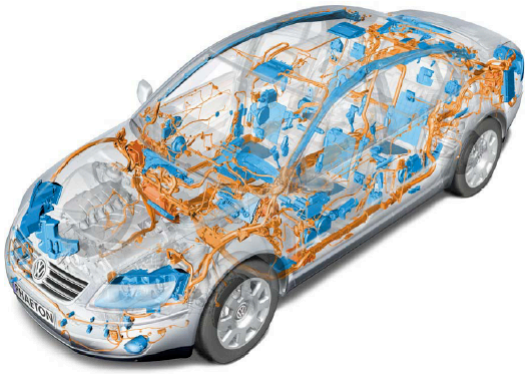
→ **optical bus** for high bandwidth Infotainment-data

→ sub-networks based on proprietary serial bus

→ **35 ECUs** connected by **3 CAN-busses** sharing:

→ appr. **2500 signals**

→ in **250 CAN messages**

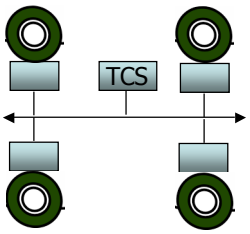


The VW Phaeton

Adapted from (Loehold, WFCS2004)

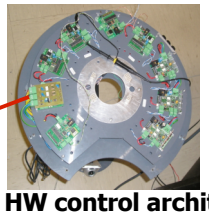
Automotive and control

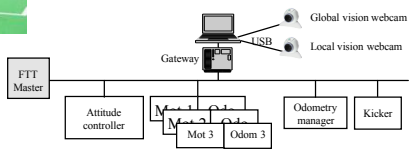
Bosch traction control system (TCS)



Autonomous robotics

The CAMBADA robotic soccer team





→ **9 ECUs** connected by **1 CAN-bus** conveying:

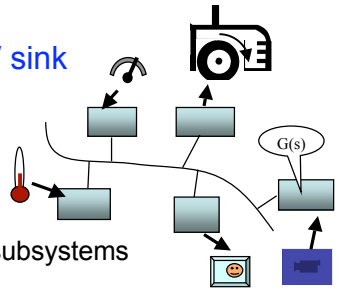
→ **9 periodic** messages

→ **12 aperiodic** messages

HW control architectures for mobile robots

Why distributed architectures

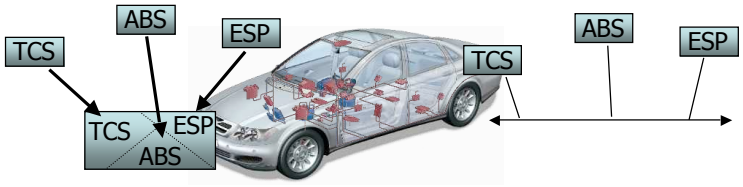
- ✓Processing closer to data source / sink
  - ✓ Intelligent sensors and actuators
- ✓Dependability
  - ✓ Error-containment within nodes
- ✓Composability
  - ✓ System composition by integrating subsystems
- ✓Scalability
  - ✓ Easy addition of new nodes with new or replicated functionality or to increase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



Towards integration

There is also a trend to **increase integration** among subsystems as a way to

- ✓ **Improve performance** by coupling subsystems
- ✓ **Reduce** number of active **components** and **costs**
- ✓ **Manage complexity**



However, integration is a source of interference!

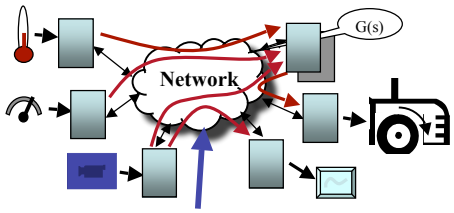
Network-centric trend

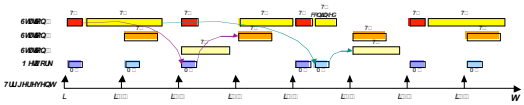
Higher integration and distribution lead to a **stronger impact of the network** on the global system properties:

- ✓ Composability, timeliness, flexibility, dependability...

And may impact on the **quality of the network services**

Communication requirements must be considered together with execution requirements at early phases of system design



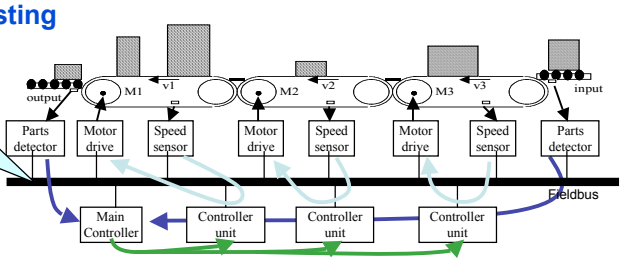


Bandwidth:
Limited shared resource

Typical DCCS service requirements

- ✓ In typical DCCS the network must support
  - ✓ Efficient transmission of **short data** (few bytes)
  - ✓ **Periodic** transmission (*monitoring, feedback control*) with **short periods** (ms) and **low jitter**
  - ✓ **Fast** transmission (ms) of **aperiodic** requests (*alarms*)
  - ✓ Transmission of **non-real-time data** (*configuration, logs*)
  - ✓ **Multicasting**

•Short data
•Periodic & aperiodic,
•Single broadcast domain



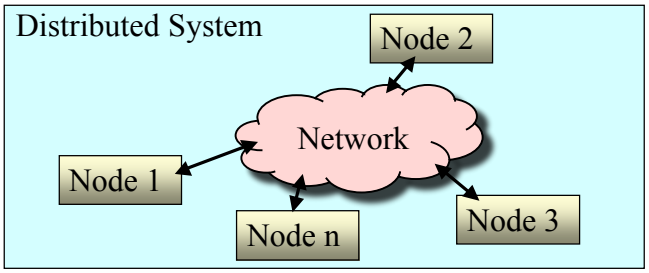
How difficult is it meeting these requirements?

How to build networks that meet them?

How to verify that the requirements are met?

The network in a DS

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



But what is a network?

- **Physical infrastructure** ?
  - Wiring
  - Connectors
  - Networking equipment

OR

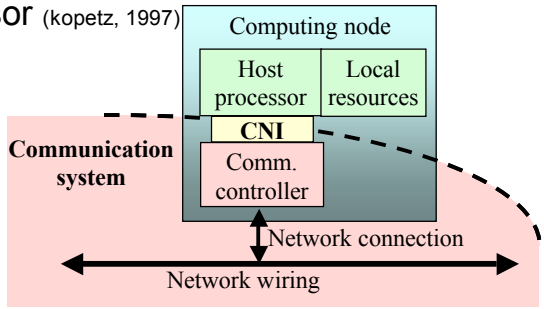
- **Communication system** ?
  - Physical infrastructure **plus**
  - Network access interfaces
  - Protocol stack

Preferred definition for "networkers"

Supports a system-wide vision that facilitates reasoning about the interplay between all communicating nodes.

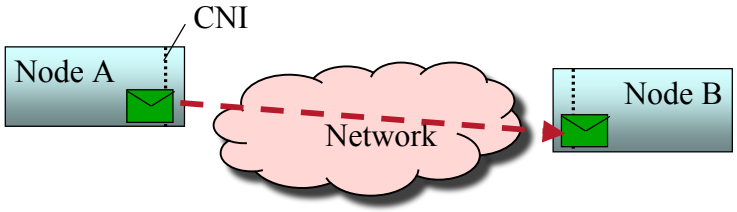
Network interfaces

- The network extends up to the **Communication Network Interface (CNI)** that establishes the frontier between communication system and the node host processor (kopetz, 1997)



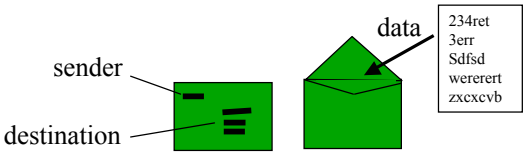
Purpose of the network

- To deliver **communication services** to the requesting nodes (i.e., to transport messages from the CNI of a sender node to the CNI of a receiver node) reliably, securely, efficiently and (for a real-time network) **timely**



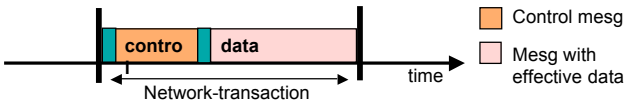
Messages

- **Interactions** are supported by **message passing**
- A **message** is a **unit of information** that is to be transferred, at a given time, from a sender process to one or more receiver processes
- Contains both the respective **data** as well as the **control** information that is relevant for the proper transmission of the data (e.g. sender, destination, contents).



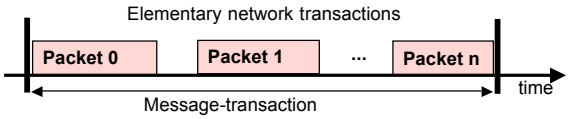
Transactions

- A **network transaction** is the **sequence of actions** within the communication system required to accomplish the **effective transfer** of a message's data.
- A transaction might include the transfer of messages carrying protocol control information, only. These are referred to as **control messages**.
- A multi-message transaction is **atomic** when all its messages must be transmitted without interruption.



Transactions

- ✓ Many networks automatically break large messages in smaller packets (fragmentation/reassembly).
- ✓ In that case, a **message transaction** includes several **elementary network transactions** that correspond to the transfer of the respective packets.
- ✓ A **packet** is the smallest unit of information that is transmitted without interruption (when there is no risk of confusion we will use **message** and **packet** interchangeably).



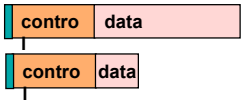
Transactions

- ✓ The **data efficiency** of the network protocol can be defined as the **ratio** between the time to transmit **effective data** bits and the **total duration** of the respective transaction.

$$\text{Data\_eff} = \frac{\text{data tx\_time}}{\text{transaction duration}}$$

- ✓ In general:

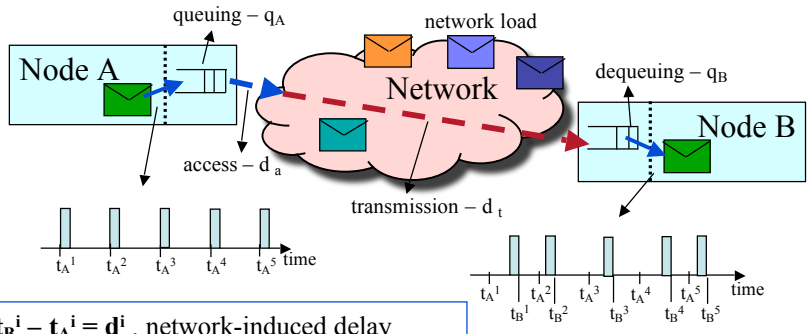
  The **shorter** the data per transaction, the **lower** the efficiency is



Timing figures

- Typical figures of merit about the temporal behavior of the network:
  - **Network induced delay** – extra delay caused by the transmission of data over the network. Some applications (e.g. control) are particularly sensitive to this delay
  - **Delay jitter** – variation affecting the network induced delay. Some applications (e.g. streaming) are not much affected by the network delay but are highly affected by delay jitter
  - **Buffer requirements** – when the instantaneous transmission from a node is larger than the capacity of the network to dispatch it, the traffic must be held in buffers. Too few buffers lead to packet losses

Timing figures



$\mathbf{t_B^i - t_A^i = d^i}$, network-induced delay

$\mathbf{d^i = q_A^i + d_a^i + d_t^i + q_B^i}$, delay components

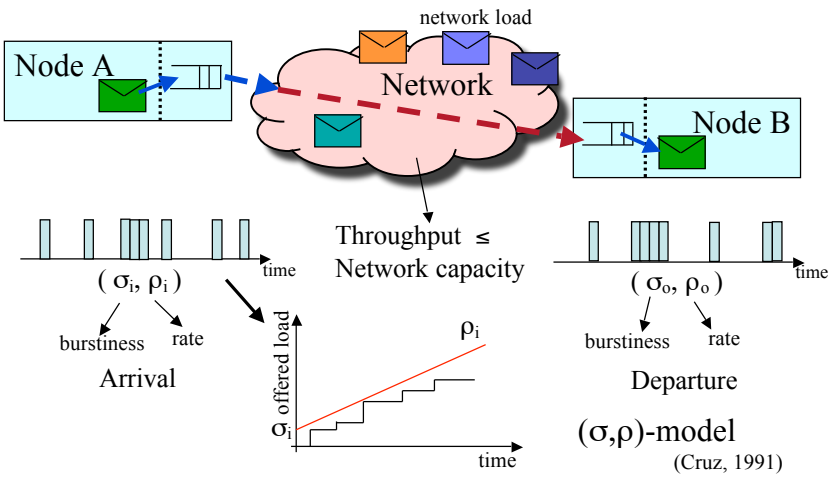
$\mathbf{d^i - d^{i-1} = j^i}$, delay jitter

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

Timing figures

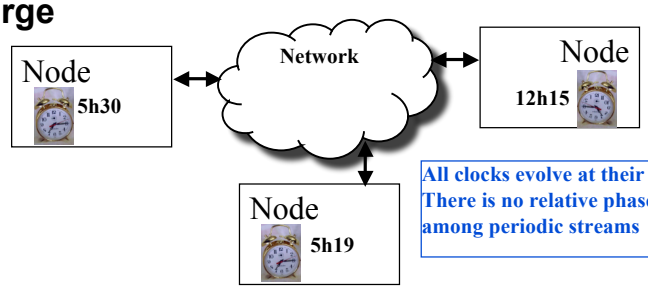
- Other relevant figures of merit:
  - **Throughput** – amount of data, or packets, that the network dispatches per unit of time (bit/s and packets/s).
  - **Arrival / departure rate** – rate at which data arrives at/from the network (bit/s and packets/s).
  - **Burstiness** – measure of the load submitted to the network in a short interval of time. Bursts have a profound impact on the real-time performance of the network and impose high buffering requirements.
    File transfers are a frequent cause of bursts.

Timing figures



Time across a network

- ✓ As opposed to a centralized system, in a distributed system **each node has its own clock**
- ✓ Without specific support, there is no explicit coherent notion of time across a DS
- ✓ Worse, due to **drift**, clocks tend to **permanently diverge**



All clocks evolve at their own pace! There is no relative phase control among periodic streams

Time across a network

- ✓ However, a **coherent notion of time** can be very important for several applications to:
  - **Carry out actions at desired time instants**
    e.g. synchronous data acquisition, synchronous actuation
  - **Time-stamp data and events**
    e.g. establish causal relationships that led to a system failure
  - **Compute the age of data**
  - etc.

But how to **synchronize the clocks** across the network?

Synchronizing clocks

- ✓ Clocks can be synchronized:
  - ✓ **Externally** – an external clock source sends a time update regularly (e.g. GPS)
  - ✓ **Internally** – nodes exchange messages to come up with a global clock value
    - ✓ **Master-Slave** – A special node (time master) spreads its own clock to all other nodes
    - ✓ **Distributed** – all nodes perform a similar role and synchronize their clocks using the clocks of the others, for example, using an average (e.g. FTA, Fault-Tolerant Average)
    - ✓ ...

Clock synchronization requirements

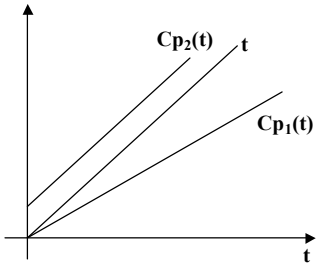
- Example: Substation Automation

  Class(Sync. Accuracy): T1(1ms); T2(0.1ms); T3(25μs); T4(4μs); T5(1μs)
- Additional requirements for DCCS:
  - Must be available on different networking technologies
  - Minimal administration is highly desirable,
  - The technology must be capable of implementation on low cost and low-end devices,
  - The required network and computing resources should be minimal.

Clock synchronization

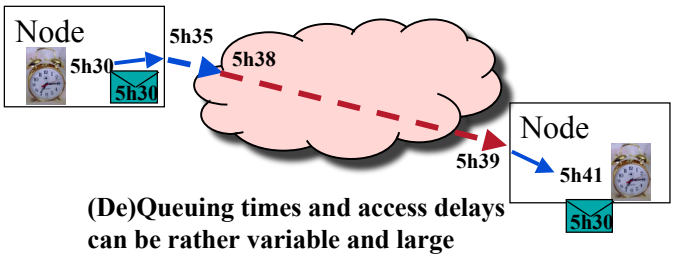
Few definitions ($Cp_i(t)$ is the clock of node *i* at instant *t*)

- **Accuracy α**
  $|Cp_i(t) - t| \leq \alpha$ for all *i* and *t*
- **Precision δ**
  $|Cp_i(t) - Cp_j(t)| \leq \delta$ for all *i, j, t*
- **Offset**
  Difference between Cp(t) and t
- **Drift**
  Difference in growing rate between Cp(t) and t



Clock synchronization

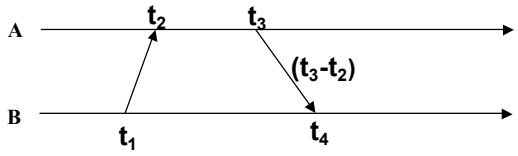
- Uncertainties in **network-induced delay** lead to limitations in the achievable **precision**



(De)Queuing times and access delays can be rather variable and large

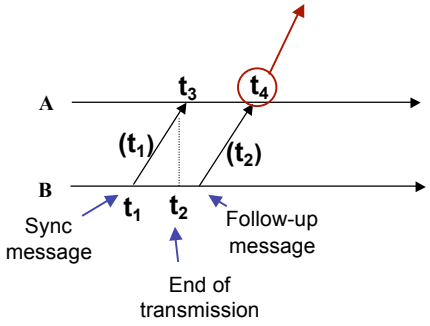
Clock synchronization

- Methods to achieve synchronization
  - **Measuring the round-trip delay** (used by NTP)
  - Network-induced delay estimated from **$((t_4-t_1)-(t_3-t_2))/2$** on node A



Clock synchronization

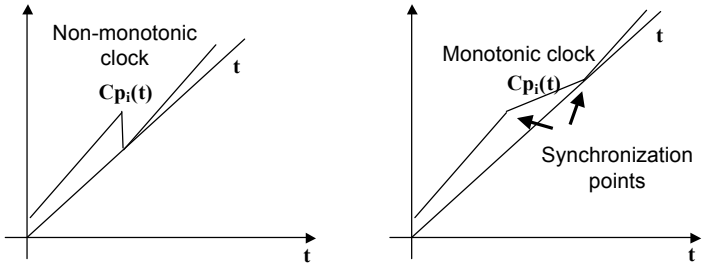
- Methods to achieve synchronization
  - **Follow up messages** (used by IEEE1588) (master-slave)
  - Network-induced delay estimated from **(t_2-t_1)** on node B



IEEE1588 also uses the round-trip delay for the slave to correct the offset.

Clock synchronization

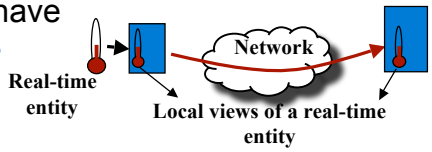
- An important point is that **clocks must be monotonic**
  $t_1 < t_2 \Rightarrow Cp_i(t_1) < Cp_j(t_2)$ for all i, t_1, t_2
- Therefore, when applying a correction to a local clock care must be taken to keep the monotonicity property



Part 2
Temporal control of communication

Real-time messages

- A message related to a *real-time entity* (e.g. a sensor) is a **real-time message**.
- Real-time messages must be transmitted within precise **time-bounds** to assure **coherence** between senders and receivers concerning their **local views** of the respective real-time entities
- Real-time messages can have **event** or **state semantics**

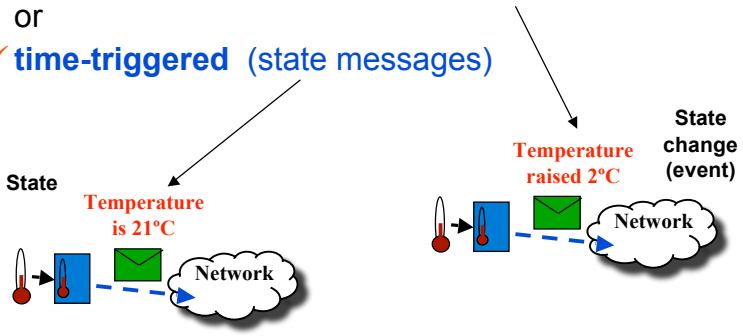


Real-time messages

- **Events** are perceived changes in the system state. All are significant for **state consistency** across sender and receiver
- **External events** refer to the environment outside the computing system (normally asynchronous)
- **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 overwrite the values of previous messages concerning the same real-time entity.

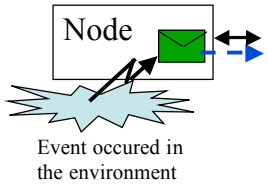
Event or Time triggering

- According to the type of messages (event or state) conveyed by the network, it can be
  - **event-triggered** (event messages)
    or
  - **time-triggered** (state messages)

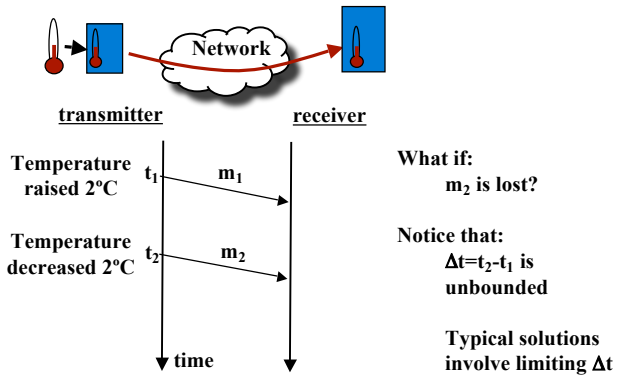


Event triggered network

- Transactions, carrying event data, are **triggered** upon **event occurrence**.
- Consequently, transmission instants are generally asynchronous wrt the nodes or network.
- Receivers know about **system state** by means of the **received events**.
- The submitted communication load (number of *simultaneous* message transmission requests) may be very high (**event showers**).

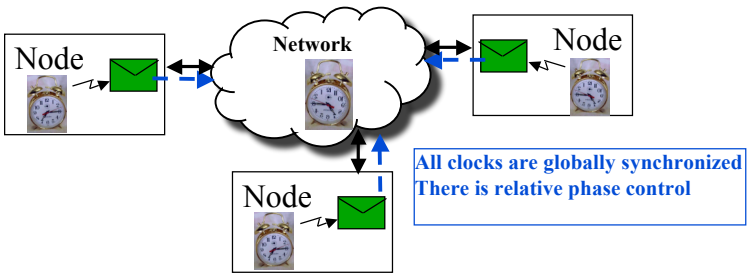


Event triggered network

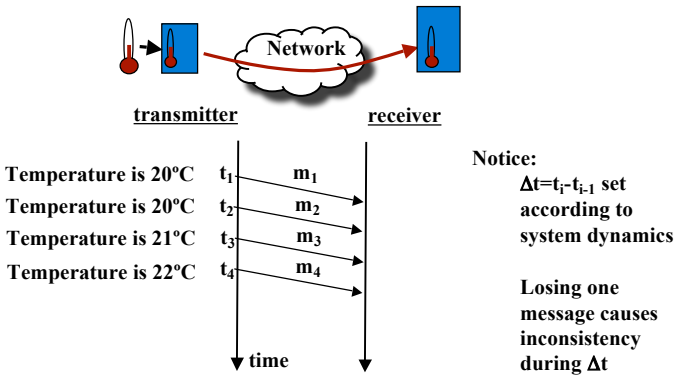


Time triggered network

- ✓ There is a notion of **network time** (explicit or implicit)
- ✓ 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



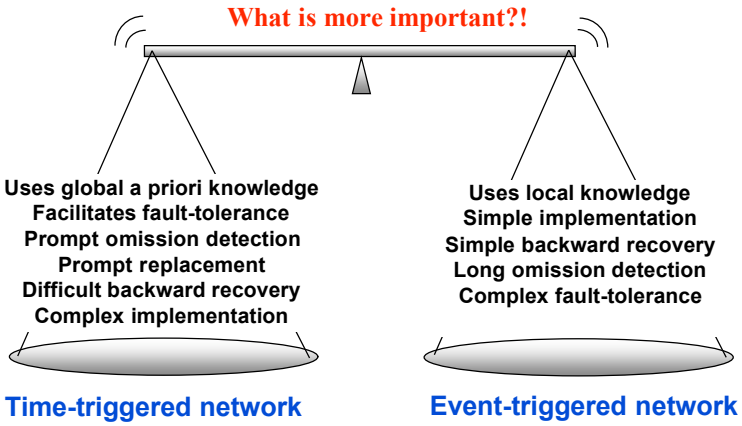
Event vs time triggering

- **Time-triggered network**
  - There is **more knowledge about the future**
    - Transmission instants are predefined
  - It is **easier to design fault-tolerance mechanisms**
    - Low omission detection latency
    - Simple management of spatial replication
  - However, there is **less flexibility to react** to errors
    - Retransmissions are often not possible because the traffic schedule is fixed
    - Without spatial replication, **latency to recover** from an omission is **normally long** (about one period of the message stream)
  - The communication **protocol is substantially complex** because of the amount of global a priori knowledge

Event vs time triggering

- **Event-triggered network**
  - There is **few knowledge about the future**
    - Events can occur at any time
  - More complex **fault-tolerance mechanisms**
    - Use of failure detectors, e.g. using **heartbeats** (maximum inter-transmission or silence time)
    - Omission detection takes about one heartbeat period
    - Replication is harder to manage (replicas may arrive at different instants in each channel)
  - However, there is **high flexibility to react** to errors
    - Retransmissions can usually be carried out promptly (or at least tried…)
    - Without spatial replication, **latency to recover** from an omission can be **very short** (time for one retransmission)
  - The communication **protocol is normally simple** to deploy, using local information, only, with few or no a priori knowledge

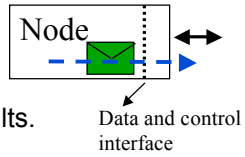
Event vs time triggering



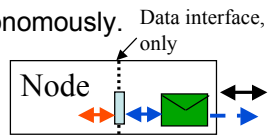
What about both?

Transmission control

- Determines **who** triggers network transactions, application or network
  - **External control**
    Transactions are triggered upon explicit control signal from the application.
    Messages are queued at the interface.
    Highly sensitive to application design/faults.



  - **Autonomous control**
    The network triggers transactions autonomously.
    No control signal crosses the CNI.
    Applications exchange data with the network by means of buffers.
    Deterministic behavior.



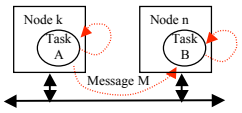
Transmission control

- All **4 combinations** of network type and transmission control are possible, depending on how much global a priori knowledge there is and whether it is located within or outside the network:
  - **ET network with external control**
    no global a priori knowledge
  - **TT network with autonomous control**
    network includes global a priori knowledge
  - **ET network with autonomous control**
    some global a priori knowledge within the network (e.g. predefined servers per node)
  - **TT network with external control**
    global a priori knowledge kept by the application but outside the network

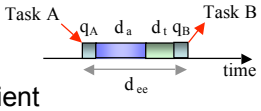
Typical approaches to combine ET and TT

Information flow

- Determining **end-to-end delay**

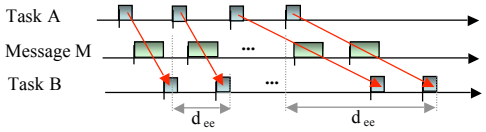


  - **ET-network with external control**
    Transactions are composed of several elementary actions carried out in sequence.



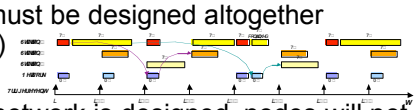
  - **TT-network with autonomous control**
    The elementary actions in each intervenient (transmitter, network, receiver) are decoupled, spinning at an appropriate rate.

Requires relative phase control to avoid high delays and jitter

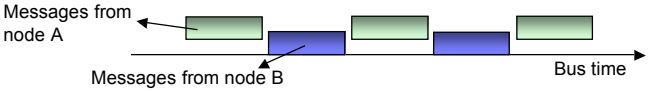


Information flow

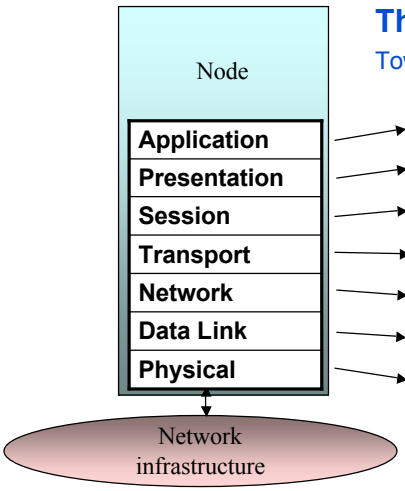
- **In a TT-network**
  - The tight control of the relative phase required between all system activities (message transmissions and task executions) imposes **rigid architectural** constraints
    - **Time-triggered architecture**
  - The whole system must be designed altogether (network and nodes)



  - However, once the network is designed, nodes will not interfere with each other (transmissions occur in disjoint intervals)
    - **Composability with respect to temporal behavior**



Protocol stack

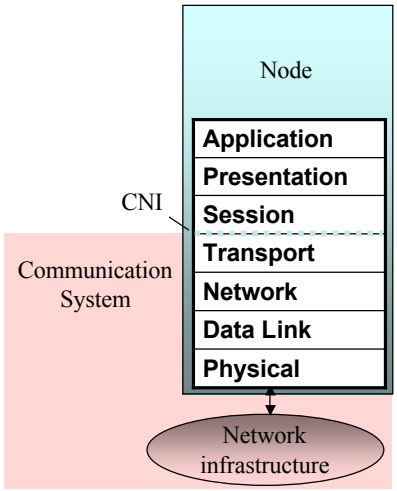


The OSI reference model

Towards interoperability and open systems

- Application: Common services for class of apps.
- Presentation: Data semantics
- Session: Remote actions
- Transport: End-to-end comm. control
- Network: Routing, logical addressing
- Data Link: Physical addressing, MAC, LLC
- Physical: Topology, medium, bit-encoding

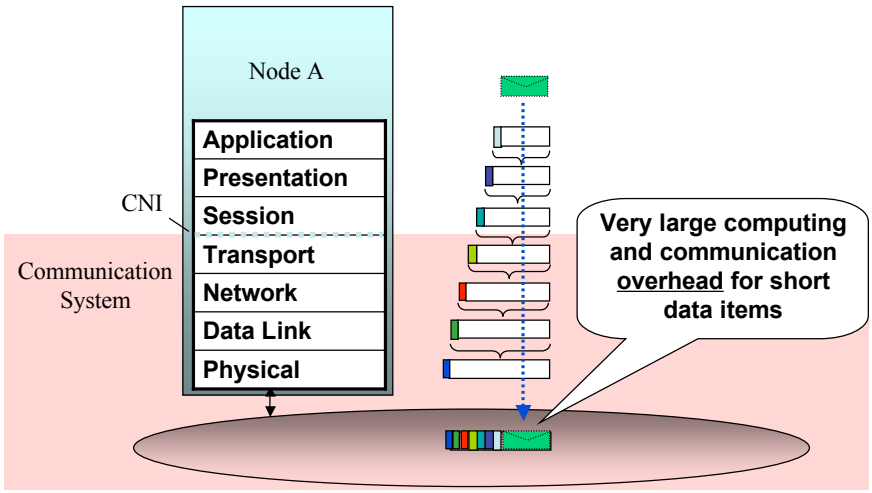
Protocol stack



Access to **information and communication services** for different types of applications

Services for the effective **transfer of information**

Protocol stack

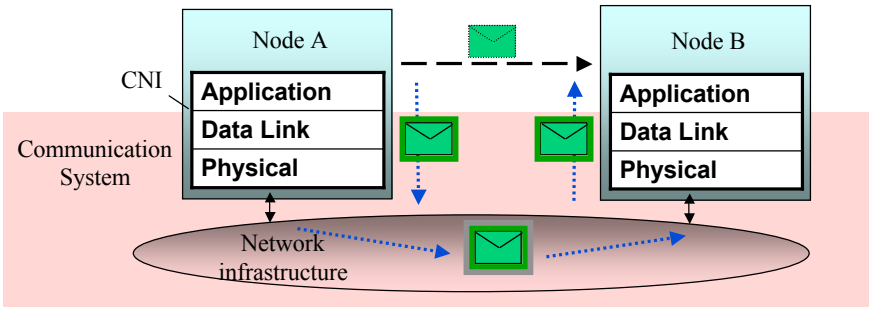


Real-time protocol stack

- The **end-to-end communication delay** must be **bounded**
  - **All services** at all layers must be **time-bounded**
  - Requires appropriate **time-bounded protocols**
- The **7 layers** impose a considerable computation and communication **overhead**…
  - The time to execute the protocol stack becomes dominant wrt the communication time
- Many real-time networks
  - are dedicated to a well defined application (no need for presentation)
  - use single broadcast domain (no need for routing)
  - use short messages (no need to fragment/reassemble)

OSI collapsed model

- ✓ Application services access the Data Link directly
- ✓ Other layers maybe present but not fully stacked
- ✓ In process control and factory automation these networks are called **Fieldbuses**



Rewinding...

Rewinding…

- ✓ The **network** is the **kernel** of a **distributed system**
- ✓ Transmits **messages** by means of network **transactions**
- ✓ Clocks in the nodes diverge unless synchronized
- ✓ Clock synchronization requires exchanging a few messages
- ✓ **Real-time** messages can be **event-** or **state-messages**
- ✓ Networks can be **event-triggered** or **time-triggered** leading to opposed characteristics wrt simplicity of the protocol implementation or the simplicity of enforcing fault-tolerance
- ✓ Transmission **control** can be **external** or **autonomous**
- ✓ **TT networks** with **autonomous control** require **judicious use of relative phase** to avoid high delays and jitter → Time-triggered architecture
- ✓ **ET networks** are inherently **flexible** at run-time
- ✓ **TT networks** are typically **static** but can use multiple modes or on-line scheduling of the periodic traffic

Rewinding…

- ✓ The **OSI 7 layers** reference model imposes **too much overhead** for real-time networks (mainly in embedded control applications)
- ✓ **Real-time** properties must be enforced in **all layers**
- ✓ Real-time networks frequently use a **collapsed 3 layers** structure:
  - ✓ **physical**, **data link** and **application** layers