

Networks for Embedded Control Systems

ARTIST2 China Summer School, 2008/Jul/14-18

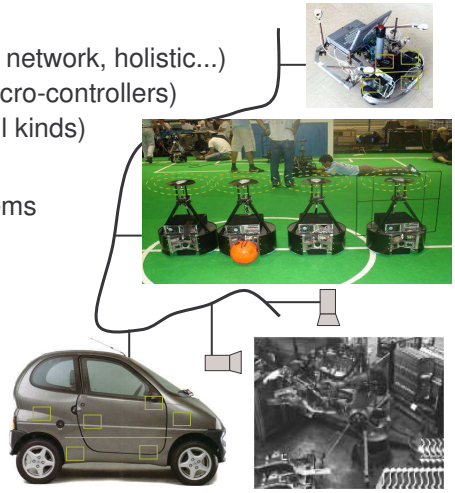
Luís Almeida
lda@ua.pt
Embedded Systems Lab @ IEETA / DETI
Universidade de Aveiro
Aveiro, Portugal ieeta

Outline

- ✓ **Trends in embedded systems**
  - ✓ Distribution, integration and operational flexibility
- ✓ **Timing issues in the network**
  - ✓ Timing parameters, time across the network
- ✓ **Temporal control of communication**
  - ✓ Event and Time triggered systems, information flow
- ✓ **Inside the protocol stack**
  - ✓ The physical, data link and application layers
- ✓ **Some related protocols**
  - ✓ WorldFIP, TTP/C, PROFIBUS, CAN, Ethernet, FlexRay, WiFi, ZigBee
- ✓ **Traffic model and scheduling issues**
- ✓ **Some on-going research topics**

My background

- Real-time systems:
  - Scheduling aspects (processor, network, holistic...)
  - RT kernels (mainly for small micro-controllers)
  - RT communication protocols (all kinds)
- Dependable systems:
  - Architecture of embedded systems
  - Safety and reliability aspects
  - Safety-critical systems
- Main *battle fields*:
  - Dynamic reconfiguration
  - On-line QoS adaptation
  - Flexible scheduling



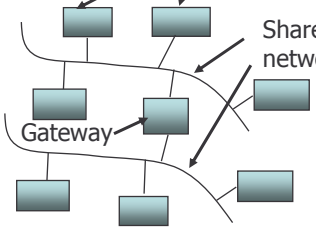
F-T-T-
Flexible Time-Triggered architecture

Part 1
Trends in embedded control systems

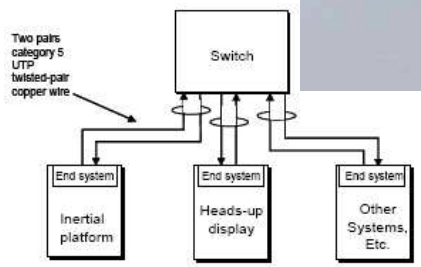
Towards distribution

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

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



Avionics

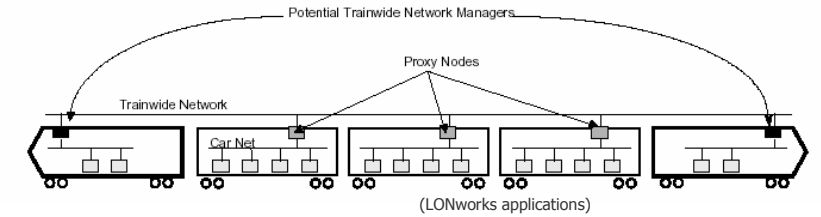


(AFDX applications – Condor Engineering)

Train control

Fast trains (Thalys, Korean project)

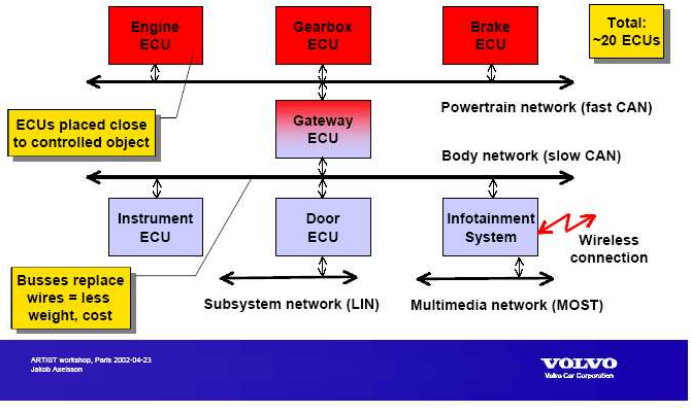
Automatic trains (Frankfurt Airport)



(LONworks applications)

Automotive

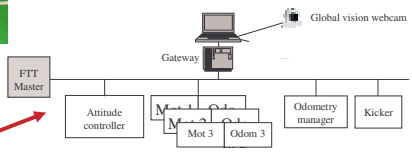
Typical Vehicle Electronic Architecture



Autonomous robotics

The CAMBADA robotic soccer team

HW control architectures for mobile robots



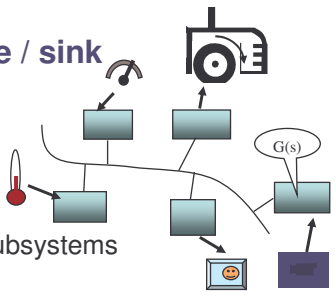
→ **9 ECUs** connected by **1 CAN-bus** conveying:

→ **9 periodic** messages

→ **12 aperiodic** messages

Why distributed architectures

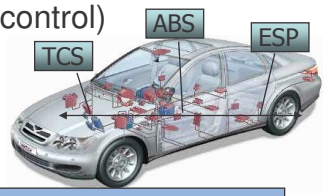
- ✓ **Processing closer to data source / sink**
  - ✓ Intelligent sensors and actuators
- ✓ **Dependability**
  - ✓ Error-containment within nodes
- ✓ **Composability**
  - ✓ System composition by integrating subsystems
- ✓ **Scalability**
  - ✓ Easy addition of new nodes with new or replicated functionality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



Dissemination of feedback control

The number of **automatic control loops** within complex embedded systems **increased dramatically** motivated by

- ✓Improved **performance** (e.g. TCS – Traction Control)
- ✓Increased **safety** (e.g. ABS, ESP – Elect. Stability Prog.)
- ✓Increased **energy efficiency** (e.g. Engine control)
- ✓New **functionality** (e.g. Autonomous parking)
- ✓More **comfort** (e.g. Cruise control)
- ✓**...**



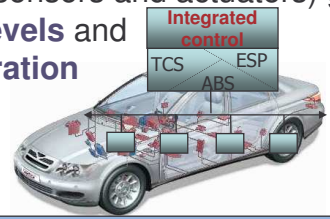
Most of such loops are distributed leading to **Distributed Computer Control Systems (DCCS)**

Towards functional integration

Integrating several quasi-independent **subsystems** allows further performance improvements

- ✓ E.g. integrating ABS, ESP, TCS and Active Suspension might significantly improve car overall stability control

Using **smart components** (sensors and actuators) greatly **simplifies higher control levels** and **facilitates functional integration**

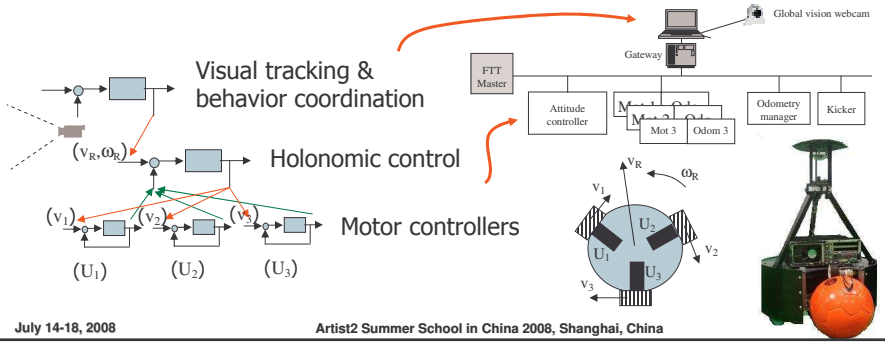


This is sometimes referred to as ***centralized* control**, but still generally using a **distributed HW architecture** !

Integration and control

***Centralized* control** is a form of **hierarchical control**

- ✓ **Integrating** the three wheels control in holonomic robots provides **full holonomic motion** (any direction with any orientation)
- ✓ Visual tracking uses **holonomic system** as a **smart actuator**

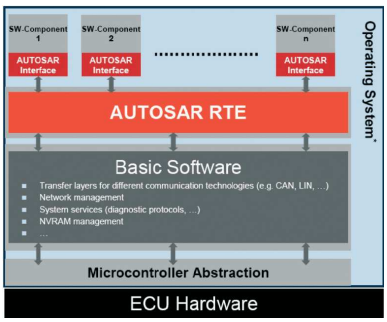


Why higher integration

- ✓ **Improve performance** by coupling subsystems (communication across functional domains)
- ✓ **Improve efficiency** in using system resources
- ✓ **Reduce** number of active **components** and **costs**
- ✓ **Manage complexity**

We need support for a **clear separation** between **SW** and **HW architectures**

- ✓ Providing SW that is unaware of the HW components
- ✓ e.g. AUTOSAR



Towards operational flexibility

Many DCCS operate in **dynamic scenarios**:

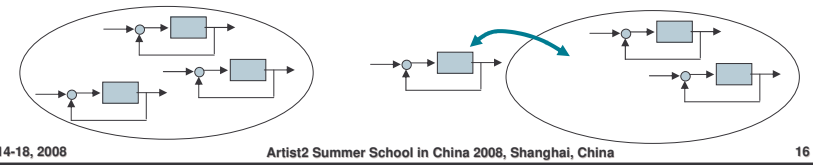
- ✓ Systems with **variable number of users** or **variable load** (traffic control, radar…)
- ✓ Systems that operate in **changing physical environments** (robots, cars…)
- ✓ Systems that can **self reconfigure dynamically** to cope with hazardous events or evolving functionality (cars, planes, trains, production cells…)

QoS adaptation, graceful degradation, survivability

Towards operational flexibility

In other words...

- ✓ Many **feedback control applications** will coexist and **share active resources** (ECUs and networks)
- ✓ Application **components will be added/replaced** during the system lifetime (system reconfiguration)
- ✓ **Backup** application components will be **switched on** at run-time to replace failed primary ones (fault tolerance)
- ✓ Application **components** will also be **adapted at run time** to cope with **current resource needs** (rate adaptation)



Putting it all together

Merging **distribution, control, integration and flexibility** raises many issues that need adequate treatment

- ✓Higher **potential for mutual interference**
  - ✓ **Higher jitter** in periodic executions
  - ✓ Control **performance degradation**
  - ✓ Control **instability**...
- ✓Conflict between **safety and flexibility**
  - ✓ Flexibility reduces **a priori knowledge**
  - ✓ Requires adequate **temporal isolation** and **temporal control**

Network-centric trend

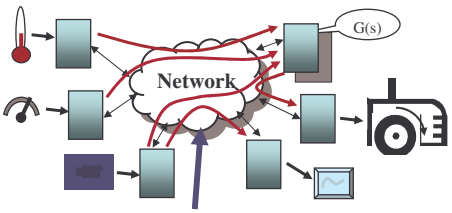
Higher integration and distribution also lead to a **stronger impact of the network** on the global system properties:

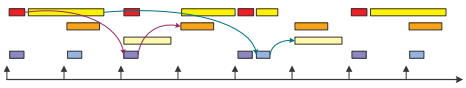
- ✓ Composability, timeliness, flexibility, dependability...

Network and nodes (tasks) are **intertwined**

- ✓ Tasks trigger messages and messages trigger tasks

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



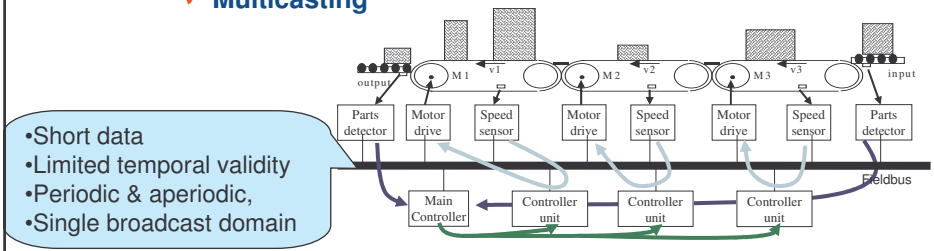


(figure by M. Calha)

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
- Limited temporal validity
- Periodic & aperiodic,
- Single broadcast domain

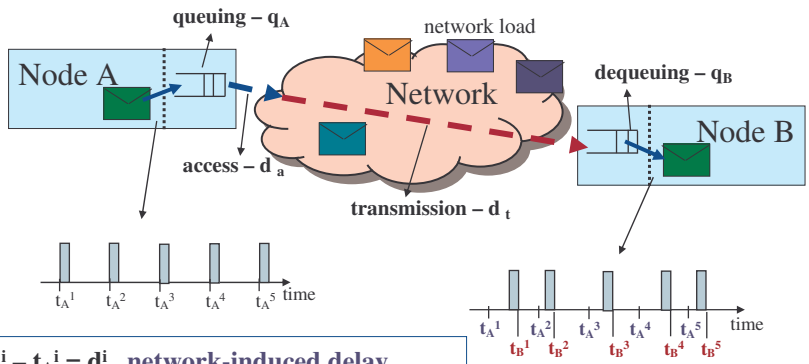
How difficult is it meeting these requirements?

How to build networks that meet them?

How to verify that the requirements are met?

Part 2
Timing issues in the network

Network delays



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

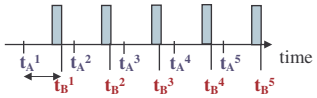
$d^i = q_A^i + d_a^i + d_t^i + q_B^i$, **delay components**

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

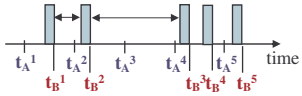
Reception instants may suffer irregular delays due to interferences from the network load, queuing policies and processor load

Delay and delay jitter

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

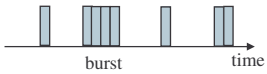


- **Delay jitter** – variation affecting the network delay.
  - Some applications (e.g. **streaming**) are not much affected by the network delay but are highly affected by delay jitter

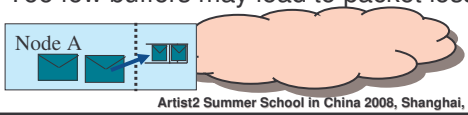


Burstiness and buffering

- **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**.

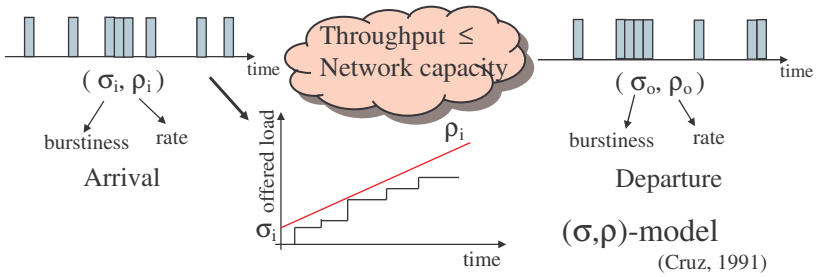


- **Buffer requirements** – Buffer capacity needed to hold packet bursts.
  - Too few buffers may lead to packet losses



Throughput and arrival / dep. rate

- ✓ **Throughput** – average amount of data, or packets, that the network dispatches per unit of time (bit/s and packets/s).
- ✓ **Arrival / departure rate** – long-term rate at which data arrives at/from the network (bit/s and packets/s).

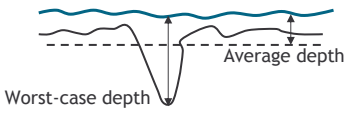


Throughput *versus* timeliness

- ***Throughput*** equates to *speed*, e.g., average number of packets delivered per unit of time
- ***Timeliness*** is related to the instant of executing certain computations (not directly related to *speed*!)
- **Timeliness** is highly influenced by the **scheduling** of activities in the system, i.e., when several packets are ready for transmission, which is sent first?

Worst-case *versus* average-case

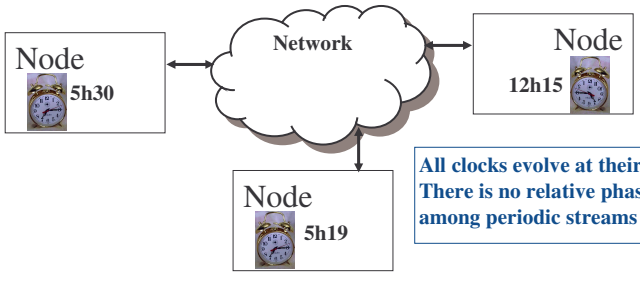
- Remember the case of the man that drowned crossing a river with average depth of 20cm!!



- Average **response times** to events can be short but a sequence of late handled events can turn a system unmanageable and lead to a critical failure (timing failure), e.g., feedback control applications.
  - In **real-time systems, late results** can
    - Be useless or even cause catastrophes (**Hard real-time**)
    - Just cause a degradation of QoS (**Soft real-time**)
  - **Importance of the worst-case delays!**

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 distributed systems
- ✓ 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**
  - **Coordinate transmissions**
    e.g. TDMA clock-based systems
  - etc.

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

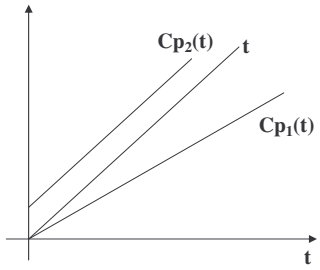
Synchronizing clocks

- 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)
- Standards: NTP, SNTP, IEEE 1588

Clock synchronization

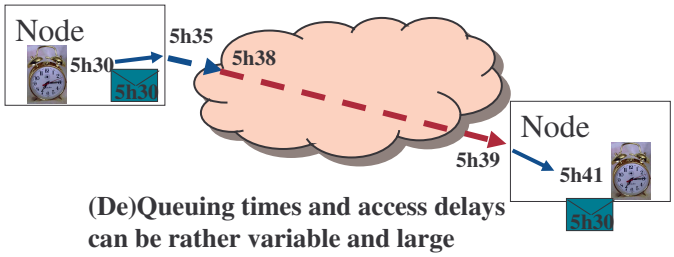
Few definitions ($Cp_i(t)$ is the clock of node i at instant t)

- **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

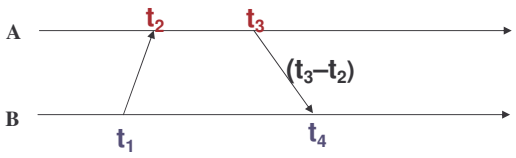
- Uncertainties in **network-induced delay** lead to limitations in the achievable **precision**



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

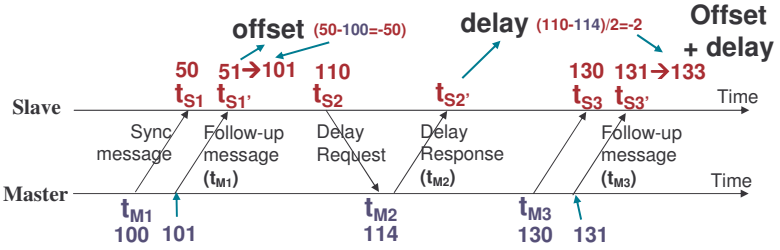
Clock synchronization

- 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 B (used to correct offset at B upon reception of time marks from A)



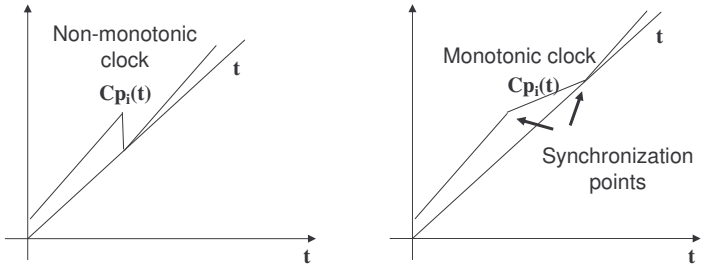
Clock synchronization

- Methods to achieve synchronization
  - **Follow up messages** (used by IEEE1588) (master-slave)
  - **Slave to Master offset** estimated from **(t_{S1}–t_{M1})** on Slave at $t_{S1'}$
    - slave clock corrected at $t_{S1'}$ ← $t_{S1'}$–(t_{S1}–t_{M1})–delay
  - **Network induced delay** estimated from **(t_{S2}–t_{M2})/2** at $t_{S2'}$
    - slave clock corrected at $t_{S3'}$ ← $t_{S3'}$–(t_{S3}–t_{M3})–**delay**



Clock synchronization

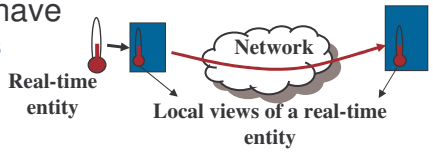
- Certain applications (e.g. banking) require **monotonic clocks**
  $t_1 < t_2 \Rightarrow Cp_i(t_1) < Cp_i(t_2)$ for all *i*, t_1, t_2
- Therefore, when applying a correction to a local clock care must be taken to keep the monotonicity property (not all protocols do!)



Part 3
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**

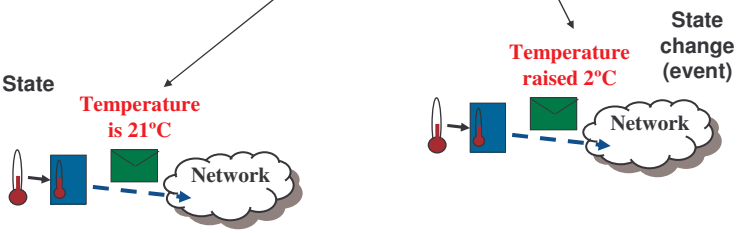


Real-time messages

- **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.

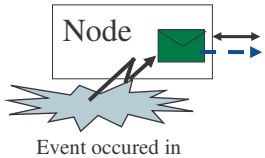
Event or Time triggering

- 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)

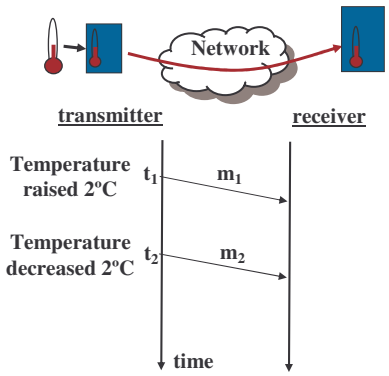


Event triggered network

- 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



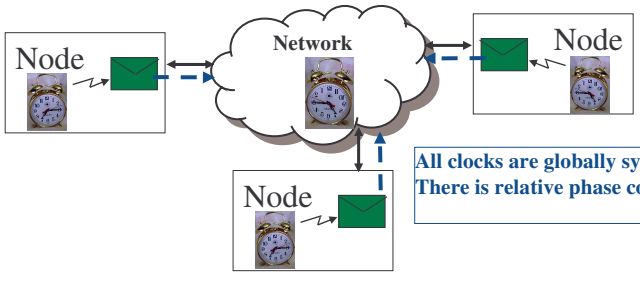
What if:
m_2 is lost?

Notice that:
$\Delta t=t_2-t_1$ is unbounded

Typical solutions involve limiting Δt (e.g. with heartbeat)

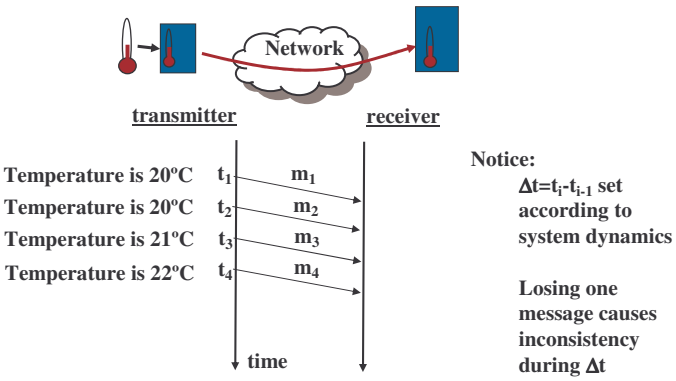
Time triggered network

- ✓ 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.



All clocks are globally synchronized
There is relative phase control

Time triggered network



Notice:

$\Delta t = t_i - t_{i-1}$ **set according to system dynamics**

Losing one message causes inconsistency during Δt

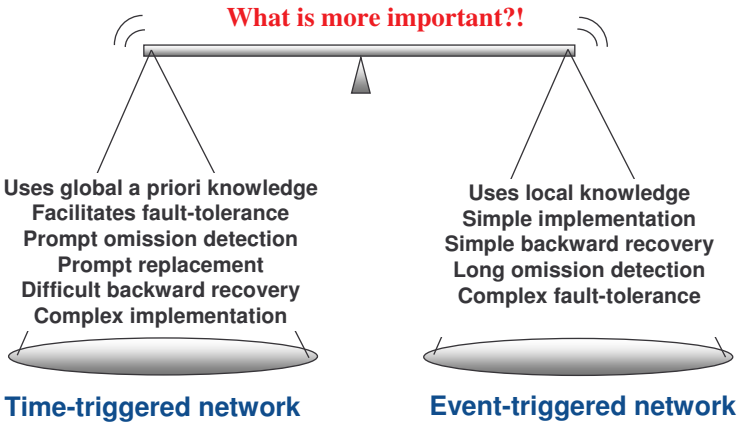
Event vs time triggering

- **Time-triggered network**
  - There is **more knowledge about the future**
    - Transmission instants are predefined ($t_{tx,i}=O_i+kT_i$, k=0,1,…)
  - It is **easier to design fault-tolerance** mechanisms
    - Low omission detection latency (expected t_{rx} known)
    - Simple management of spatial replication (idem)
  - 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 and different order 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

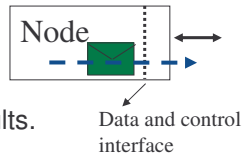
Event vs time triggering



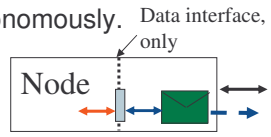
What about both?

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.



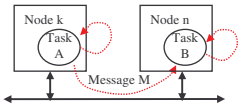
Transmission control

- 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

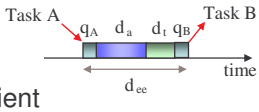
Typical approaches to combine ET and TT

Information flow

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

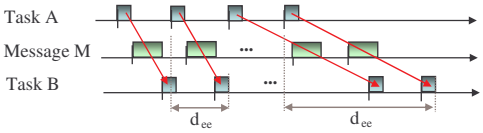


  - **ET-network with external control**
    Transactions are composed of several elementary actions carried out in sequence.



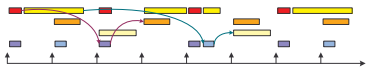
  - **TT-network with autonomous control**
    The elementary actions in each intervenient (transmitter, network, receiver) are decoupled, spinning at an appropriate rate.

Requires relative phase control to avoid high delays and jitter

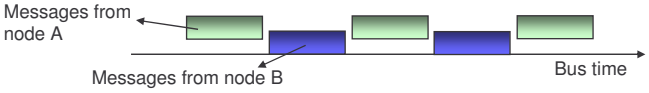


Information flow

- **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)

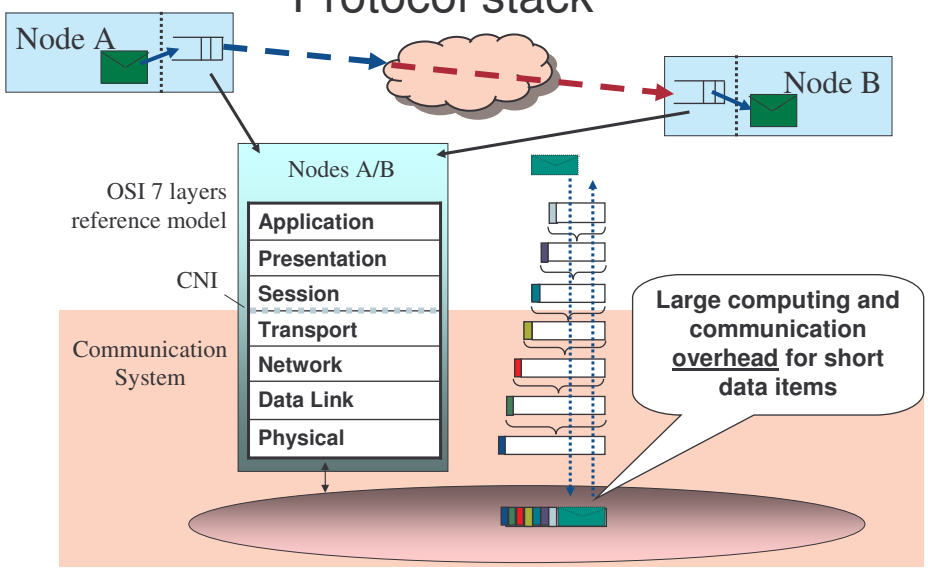


  - However, once the network is designed, nodes will not interfere with each other (transmissions occur in disjoint intervals)
    - **Composability with respect to temporal behavior**



Part 4
Inside the protocol stack

Protocol stack

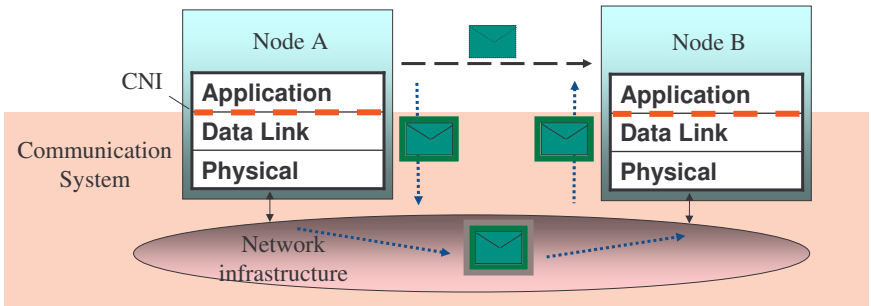


Requirements for a real-time protocol stack

- 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 may become 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)

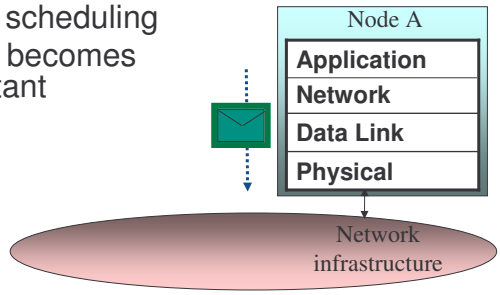
OSI collapsed model

- 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**



The case of WSNs

- A network layer with **routing** services is fundamental (multi-hop communication)
- Some more complex application layers: **Specific middlewares** with certain services (e.g., localization, tracking, data aggregation)
- **Energy aware / location aware** routing and traffic scheduling
- **Synchronization** becomes particularly important

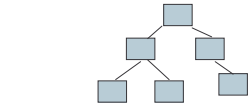


Physical layer

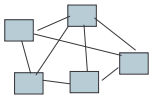
- Issues related with the physical layer:
  - **Interconnection topology**
  - **Physical medium**
  - Coding of digital information
  - **Transmission rate**
  - Maximum interconnection length
  - Max number of nodes
  - Feeding power through the network
  - **Energy Consumption**
  - Immunity to EMI
  - Intrinsic safety

Physical layer

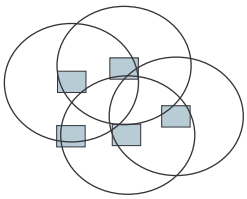
✓ **Interconnection topology**



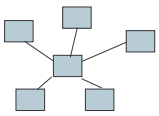
Tree



Mesh
(wired)

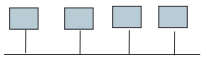


Mesh (wireless-RF)

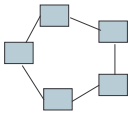


Star

Bus



Ring



Physical layer

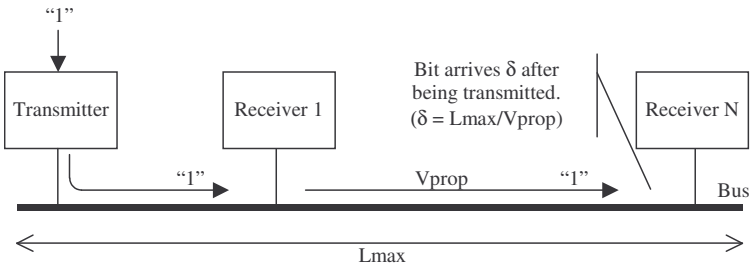
Topology	In favor	Against
Mesh	Point-to-point connections (wired, only). Several alternative paths.	Requires routing. Complex cabling (wired), difficult to maintain. Difficult to enforce total order
Tree (star)	Point-to-point connections. Simultaneous communication in parallel branches.	Requires routing. Potential long paths for deep nodes in different branches. Upper branches are bottlenecks.
Ring	Point-to-point connections. Simplified cabling.	Long path for back-to-back nodes. Depending on protocol, the whole ring is used as shared medium (more complex access control)
Bus	Simplified cabling. Direct communication (no routing)	Shared communication medium (more complex access control)

Physical layer

- **Physical medium**
  - **Copper wiring**
    - Cheaper cables and interfaces (+), suffers EMI (-)
  - **Optical fibers**
    - Immune to EMI, favors safety, wide bandwidth, low attenuation (+), expensive cables and interfaces (-)
  - **Wireless – Radio frequency**
    - Mobility, flexibility (+), very susceptible to EMI (-), multi-path fading (-), attenuation (-), open medium (+/-)
  - **Wireless – Infrared light**
    - Mobility, flexibility, immune to EMI (+), line-of-sight (-), open medium(+/-)

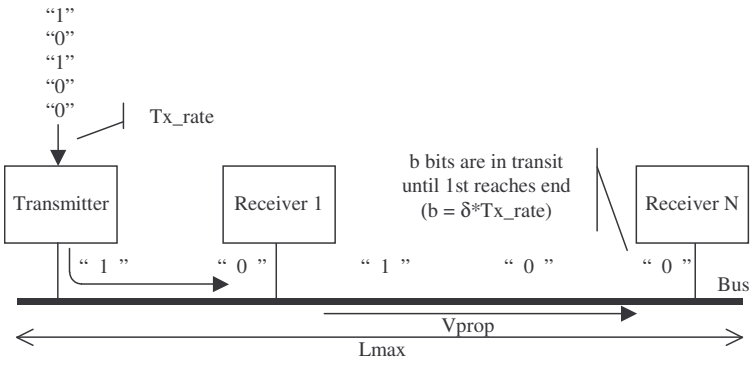
Physical layer

- **Propagation delay** in a bus
  - Time for a bit to traverse the full length of the bus (δ)



Physical layer

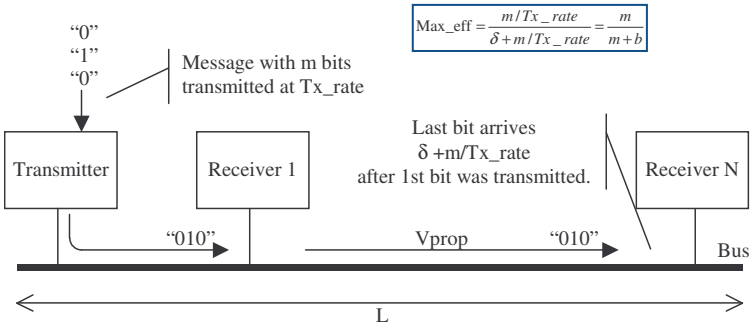
- **Bit length** in a bus
  - Number of bits in transit in the bus (b), given δ and Tx_rate



Physical layer

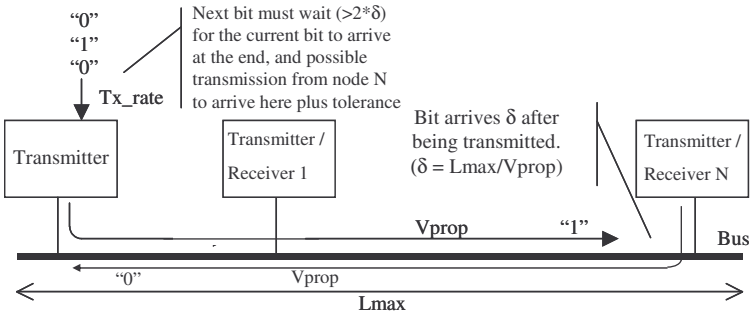
- **Limit to protocol efficiency in a bus**
  - Any message must be flushed from the bus before the next can be transmitted

$$\text{Max\_eff} = \frac{m/Tx\_rate}{\delta + m/Tx\_rate} = \frac{m}{m+b}$$



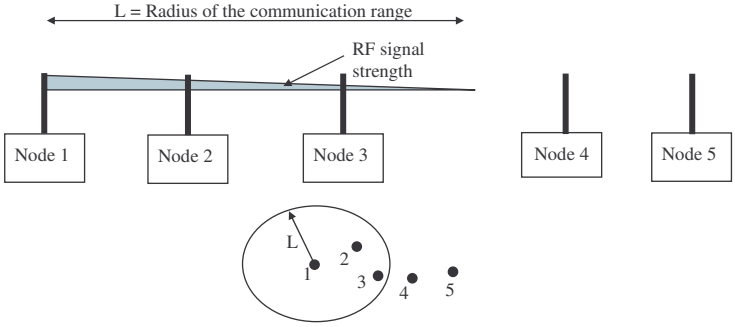
Physical layer

- **Special cases**
  - Some protocols require spatial coherence at the bit level (b=1 => Tx_rate < 1/(2δ)) (case of CAN)



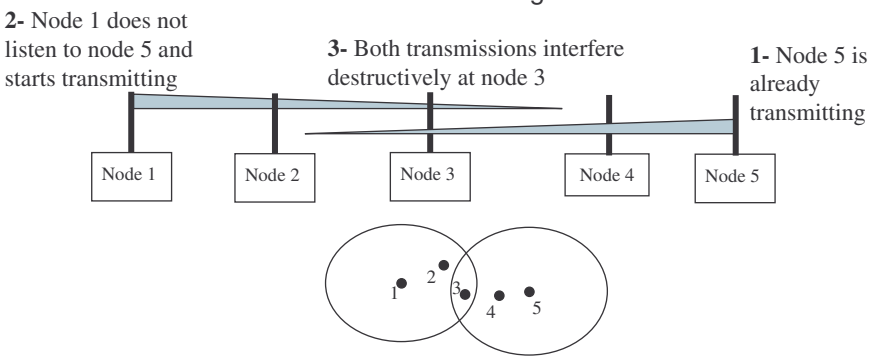
Physical layer

- **Special cases - wireless**
  - In wireless networks the **attenuation** is strong (even worse with obstacles) and transmissions from one node may not reach all nodes in the network.



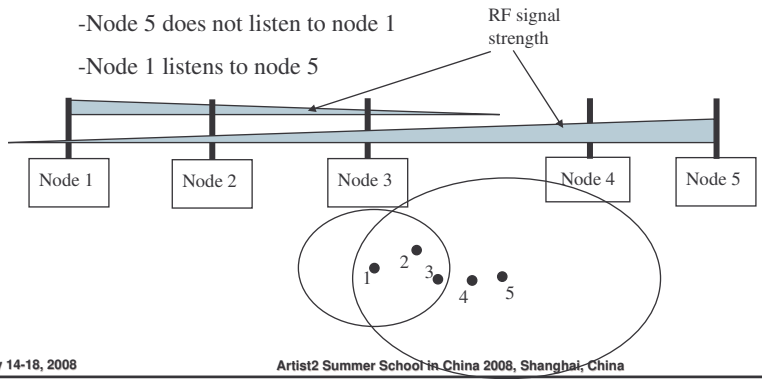
Physical layer

- **Special cases - wireless**
  - **Attenuation** is also responsible for a phenomenon called the **hidden node terminal** that jeopardizes transmission control based on carrier sensing



Physical layer

- **Special cases - wireless**
  - **Transmission power, antenna efficiency** and **local noise** deeply influences the communication range ➔ Differences cause **unidirectional** communication links



Physical layer

- **Detection of collisions**
  - In shared broadcast buses
    - If bit length **b=1**, **send 1 bit and listen**
    - If bit length **b>1**, it is not possible to relate the bit being transmitted with what is sensed on the bus. In this case a **jamming signal** is used
  - In **wireless** transmission it is not common to transmit and receive at the same time (expensive, requires multiple transceivers). **Collisions** can be sensed indirectly by means of **acknowledging**.

Physical layer

- **Energy consumption**
  - Influenced by: Transmission power, data rate, frame rate, network traffic, error control coding, retransmissions policy, type of medium access control and physical bit encoding.
  - Very **complex relationship** among the above parameters. Need to trade-off.
  - Highly relevant in **autonomous systems**, particularly in those expected to operate continuously for long term (**wireless sensor networks**).
  - **A nice challenge!**

Physical layer

- **Energy consumption trade-offs in wireless**
  - **Higher data rate** uses **more energy** but transmissions take **less time** (however, probability of errors and retransmissions also increases)
  - In a mesh-like network (WSNs), **higher tx power** might **reduce the number of hops**, thus less energy spent in storing and forwarding
  - **Asynchronous transmissions** (ET-like) **reduce power** on tx but receivers may need to be **awake all the time**!
  - **Collisions** require **retransmissions** but avoiding them requires a synchronization mechanism
  - ...

Physical layer

- **Energy consumption in wired networks**
  - Some protocols use **transceivers** that are particularly **energy expensive** (e.g. CAN).
  - **Reducing transmissions** might be desirable in autonomous mobile robots
    - In some PIC18F458 based boards, overall current consumption raised from 70mA to 110mA corresponding to no CAN transmissions and constant transmission, respectively.
  - ...

Data link layer

- Issues related with the data link layer:
  - **Addressing**
  - **Logical link control – LLC**
    - **Transmission error control**
  - **Medium access control – MAC**
    (for shared media)

Data link layer

- **Addressing**
  identification of the parts involved in a network transaction
  - **Direct addressing**
    The sender and receiver(s) are explicitly identified in every transaction, using **physical** addresses (as in Ethernet)
  - **Indirect (source) addressing**
    The message **contents** are explicitly identified (e.g. temperature of sensor X). Receivers that need the message, retrieve it from the network (as in CAN and WorldFIP)
  - **Indirect (time based) addressing**
    The message is identified by the time instant at which it is transmitted (as in TTP)

Data link layer

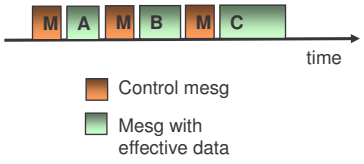
- **Logical Link Control**
  Deals with the transfer of network packets
  - Might have an impact on the network delay depending on whether **sender/receiver synchronization** is used
    - e.g., acknowledge mechanisms, retry mechanisms, request data on reply
    - Note that such synchronization using retries (PAR, ARQ) might have a strong negative impact on the data timeliness...
    - No point on continue trying to transmit out-dated values

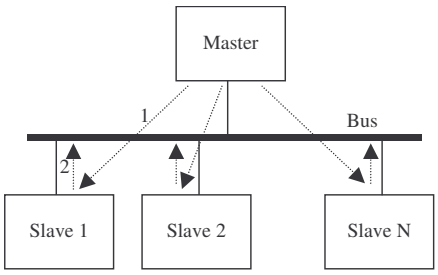
Data Link Layer

- **Medium Access Control**
  Determines the **waiting time** to **access** the network
  - It has a large impact on the network delay
- Two main families
  - **Controlled access**
    - based on transmission control
      - The network says when to transmit
  - **Uncontrolled access**
    - based on arbitration
      - The nodes try to transmit immediately when needed

Medium Access Control
Controlled access

- **Master-slave**
  - Access granted by the Master
  - Nodes synchronized with the master
  - Requires one control message per data message



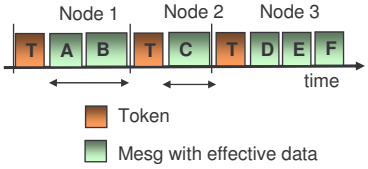


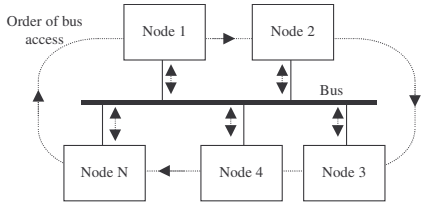
Medium Access Control
Controlled access

- **Master-slave**
  - The **traffic scheduling** problem becomes **local to the master** → good flexibility wrt scheduling algorithms
    - Requires special mechanisms to handle aperiodic communication
  - For high reliability the master must be **replicated**
  - Master messages are natural synchronization points → supports **precise tx triggering**
  - Ex: WorldFIP, Ethernet Powerlink, Bluetooth within each piconet

Medium Access Control
Controlled access

- **Token-passing**
  - Access granted by the possession of the token
  - Order of access enforced by token circulation
  - Asynchronous bus access (generally impossible to determine a priori the exact access instants)
  - Real-time operation requires bounded token holding time





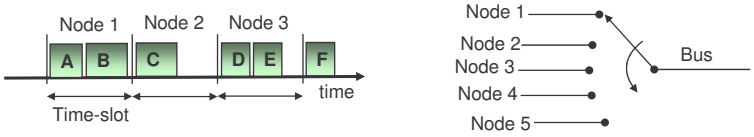
Medium Access Control
Controlled access

- **Token-passing**
  - **TTRT** – **Target** Token Rotation Time
    fundamental configuration parameter that determines the time the token should take in one round, under heavy traffic
  - **RTRT**- **Real** Token Rotation Time
    time effectively taken by the token in the last round
  - **ST** – Synchronous time
  - **THT** – Token **holding** time
  - High tx jitter
  - Requires mechanisms to handle **token losses**
  - Similar to **Round Robin Scheduling**
  - Ex: IEEE 802.4, FDDI, PROFIBUS

$$\Delta = TTRT - RTRT$$
$$THT = \begin{cases} \Delta, & \Delta > ST \\ ST, & \Delta \le ST \end{cases}$$

Medium Access Control
Controlled access

- **TDMA**
  Time-Division Multiple Access
  - Access granted in dedicated time-slot
  - Time-slots are pre-defined in a cyclic framework
  - Tight synchronization with bus time
    (bus access instants are predetermined)
  - Requires global (clock) synchronization

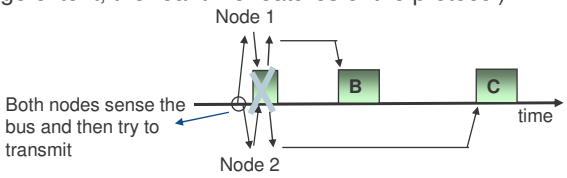


Medium Access Control
Controlled access

- **TDMA**
  Time-Division Multiple Access
  - Addressing can be based on time → **High data efficiency**
  - Quality of synchronization bounds efficiency
    (length of **guarding windows** between slots)
  - Typically uses **static table based scheduling**
  - Ex: TTP/C, FlexRay-static, TT-CAN, PROFINET

Medium Access Control
Uncontrolled access

- **CSMA**
  Carrier-Sense Multiple Access
  - Set of protocols based on sensing bus inactivity before transmitting (asynchronous bus access)
  - There may be collisions
  - Upon collision, nodes back off and retry later, according to some specific rule (this rule determines, to a large extent, the real-time features of the protocol)



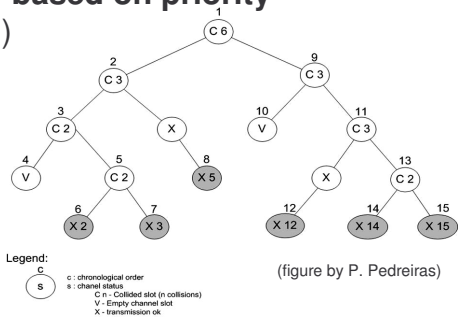
Medium Access Control
Uncontrolled access

- **CSMA/CD**
  Carrier-Sense Multiple Access with Collision Detection
  - Used in shared Ethernet (hub-based)
  - Collisions are **destructive** and are detected within **collision windows** (*slots*)
  - Upon collision, the retry interval is **random** and the randomization window is doubled for each retry until 1024 slots (BEB - Binary Exponential Back-off)
  - Non-deterministic (particularly with chained collisions)

Medium Access Control
Uncontrolled access

CSMA/DCR
Carrier-Sense Multiple Access with Deterministic Collision Resolution

- Collisions are destructive
- **back off and retry based on priority** (binary tree search)
- Deterministic
- Ex. G. Le Lann's modified Ethernet

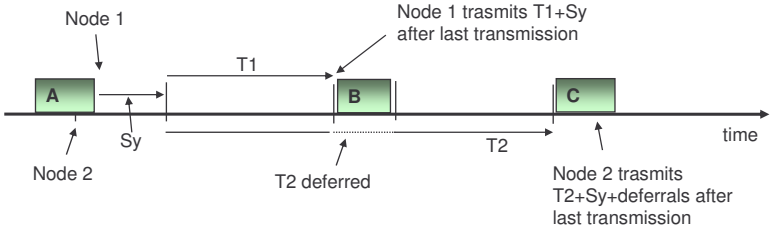


(figure by P. Pedreiras)

Medium Access Control
Uncontrolled access

CSMA/CA
Carrier-Sense Multiple Access with Collision Avoidance (async)

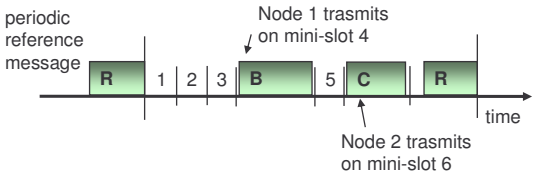
- Access based on sensing bus inactivity during a synchronization interval (Sy) plus a uniformly distributed random access interval with probability p (*p-persistent*)
- Not collision-free (Ex: IEEE 802.11)



Medium Access Control
Uncontrolled access

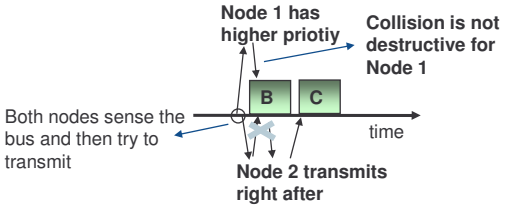
- **CSMA/CA**
  Carrier-Sense Multiple Access with Collision Avoidance (with sync)
  - Access based on sensing bus inactivity during a number of predefined time-slots (mini-slots) after reception of a synchronous reference message
  - Corresponding mini-slot determines priority
  - Collision-free and deterministic

  (Ex: FlexRay-dynamic (Byteflight), ARINC629-async)



Medium Access Control
Uncontrolled access

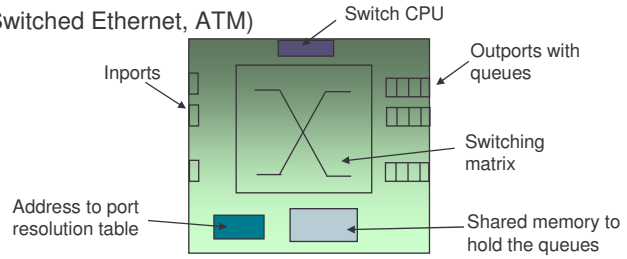
- **CSMA/BA (CA?)**
  Carrier-Sense Multiple Access with Bit-wise Arbitration
  - **Bit-wise** arbitration with **non-destructive** collisions → *Dominance protocol*
  - Upon collision, highest priority node is unaffected. Nodes with lower priorities retry right after.
  - Deterministic (e.g. CAN)



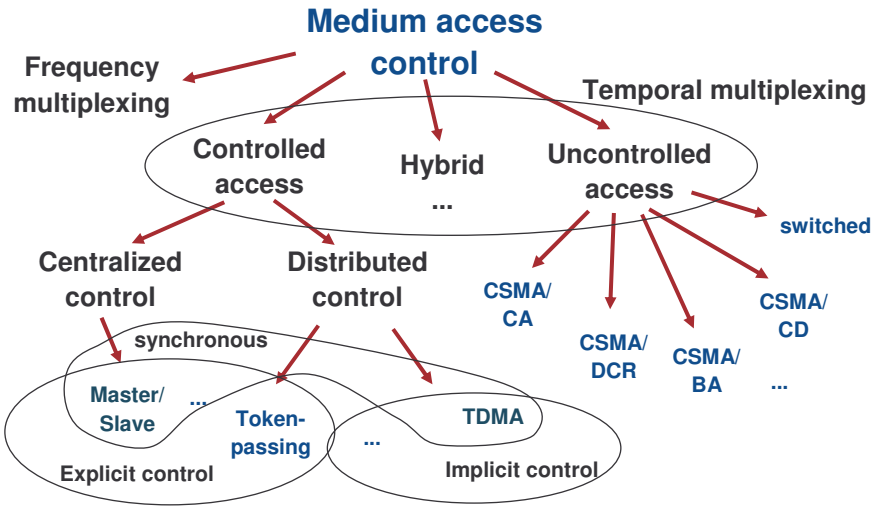
Medium Access Control
Uncontrolled access

- **Switched**
  micro-segmented network with central switching hub
  - Nodes send asynchronously messages to the switch using point-to-point links – **no collisions**
  - Contention at the network access is replaced by contention at the output ports

(e.g. Switched Ethernet, ATM)



Data link layer



Application layer

- Issues related with the application layer:
  (middleware issues)
  - **Cooperation model**
    - Client-Server
    - Producer-Consumer
    - Producer-Distributor-Consumer
    - Publisher-Subscriber

Application layer

- Cooperation model - **Client-Server**
  - Transactions are **triggered** by the **receiver** of the requested information (**client**).
  - Nodes that **generate** information are **servers** and only react to client requests.
  - The model is based on **unicast** transmission (one sender and one receiver)
  - Transactions can be **synchronous** (client blocks until server answers) or **asynchronous** (client follows execution after issuing the request)

Application layer

- Cooperation model – **Producer-Consumer**
  - Transactions are **triggered** by the nodes that **generate** information (**producers**).
  - The nodes that **need** the information, identify it when transmitted and retrieve it from the network (**consumers**)
  - The model is based on **broadcast** transmission (each message is received by all nodes)

Application layer

- Cooperation model – **Producer-Distributor-Consumer**
  - Basically similar to Producer-Consumer
  - Transactions are **triggered** by a particular node, the **distributor**, upon request from the producers or according to a pre-established schedule.
  - It is an implementation of PC over master-slave

Application layer

- Cooperation model – **Publisher-Subscriber**
  - Elaborate version of Producer-Consumer using the concept of **group communication**
  - Nodes must adhere to groups either as publisher (produces information) or as subscriber (consumes information)
  - Transactions are **triggered** by the **publisher** of a group and **disseminated** among the respective **subscribers**, only (**multicast**).

Part 5
Some related network protocols

WorldFIP
Factory Instrumentation Protocol

www.worldfip.org

WorldFIP

- ✓ Created, in the 80s, in France for use in **process control** and **factory automation**.
- ✓ AFNOR standard C46601..7 (89-92)
- ✓ CENELEC standard EN50170,vol.3 (96)
- ✓ IEC standard 61158, type 7 (2000)
- ✓ Typical in train control systems

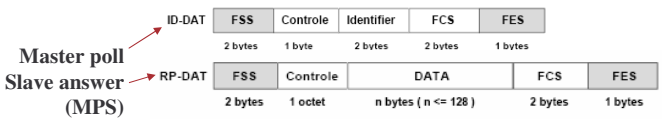
WorldFIP

- ✓ **Broadcast** serial **bus**
- ✓ Synchronous transmission
- ✓ Manchester encoding
- ✓ Transmission rates **32 Kbit/s**, **1 Mbit/s** and **2.5 Mbit/s** on copper or **5 Mbit/s** on optical fiber
- ✓ Maximum Length 2000m @ 1Mbit/s
- ✓ Max. number of nodes 256

WorldFIP

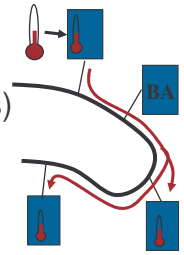
2 messaging systems:

- ✓ **MPS** – **real-time** services, periodic, aperiodic;
  MMS (ISO 9506) **subset** – non-real-time messaging
- ✓ Data **payload** between **0 and 128 bytes** (256 for non real-time messages)
- ✓ **Source addressing** (message identifiers with 16 bits)
- ✓ **Master - Slave** bus access control
  - ✓ BA - Bus Arbitrator



WorldFIP

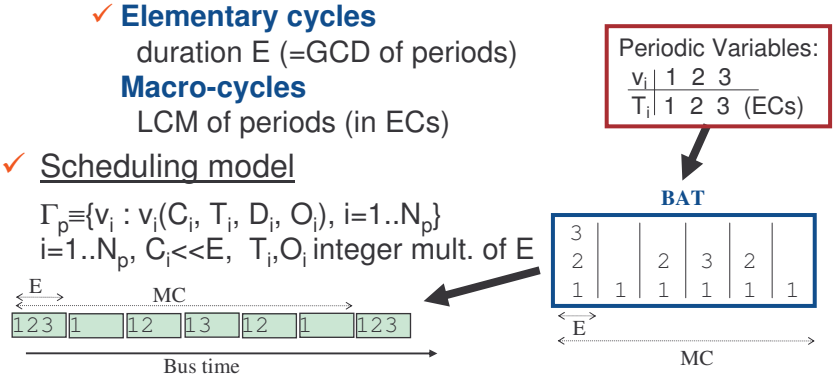
- **MPS** – *Messagerie Periodique e Sporadique*
- **Producer-Distributor-Consumer** model
- Concept of **Network Variable**
  - Entity that is distributed (several local copies coexist in different nodes)
  - Can be **periodic** or **aperiodic**
  - Local copies of **periodic variables** are **automatically refreshed** by the network
  - Local copies of **aperiodic variables** are refreshed by the network upon **explicit request**



WorldFIP

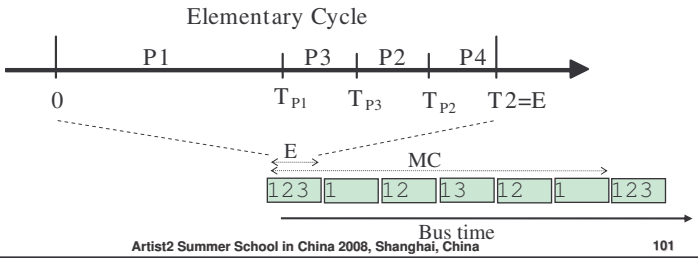
- **Table based** scheduling of **periodic** traffic
- Table (BAT) organized in **cycles**
  - **Elementary cycles**
    duration E (=GCD of periods)
    Macro-cycles
    LCM of periods (in ECs)
- Scheduling model

$\Gamma_p \equiv \{v_i : v_i(C_i, T_i, D_i, O_i), i=1..N_p\}$
$i=1..N_p$, $C_i << E$, T_i, O_i integer mult. of E



WorldFIP

- Elementary cycles organized in **phases**:
  - **Periodic** (P1)
  - **Aperiodic** (P3)
  - MMS **messages** (P2)
  - Sync - **padding** (P4)



TTP
Time-Triggered Protocol
(TTP/C – for SAE class C)

www.tttech.com

TTP/C

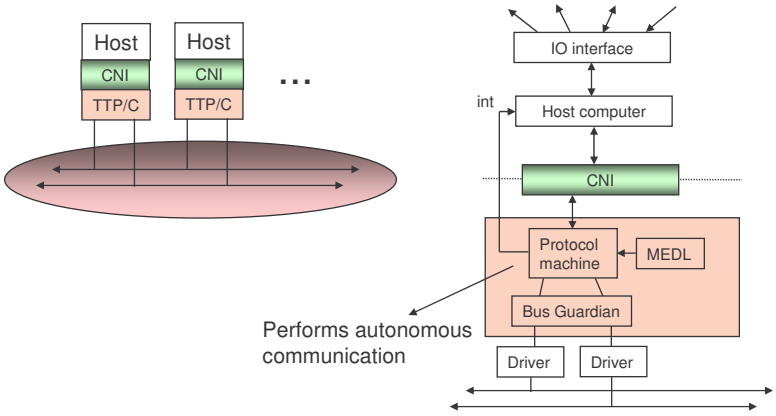
- Created around 1990 within the **MARS project** in the Technical University of Vienna
- Aims at **safety-critical** applications
- Considers an architecture with nodes integrated in **fault-tolerant units** (FTUs), interconnected by a **replicated bus**
- Includes support for prompt **error detection** and **consistency checks** as well as **membership** and **clock synchronization** services.

TTP/C

- **Multi-master**, **broadcast**, serial **bus**
- Transmission rate of **0.5, 1, 2 Mbit/s** with **MFM** (modified frequency modulation) bit encoding
- Higher tx rates with Ethernet PHY (e.g. **25 Mbit/s**)
- Max. number of nodes is **64** (possibly larger if some share the access to the network)
- CNI based on **Dual-port RAM**

TTP/C

- **Network and nodes architecture**

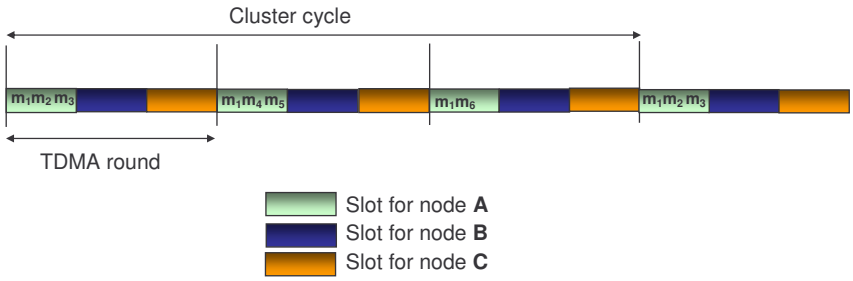


TTP/C

- **TDMA** access scheme with **one slot** allocated **per node** and **per round**
- The periodic sequence of slots is a **TDMA round** (typical values 1-10ms)
- In **each slot** the respective node may send **one frame** (up to 240 bytes)
- Each frame may contain **several messages**
- All message transmission instants are stored in a **distributed static table**, the **MEDL**

TTP/C

✓ TDMA round and Cluster Cycle



PROFIBUS
PROcess FieldBUS

www.profibus.com

PROFIBUS

- Created in the late 80's by **Siemens,** in Germany
- Aims at **process control** and **factory automation**
- DIN standard 19245-1 to 19245-3 (90)
- CENELEC standard EN50170,vol.2 (96)
- IEC standard 61158, type 3 (2000)
- Dominant protocol in factory automation in Europe

PROFIBUS

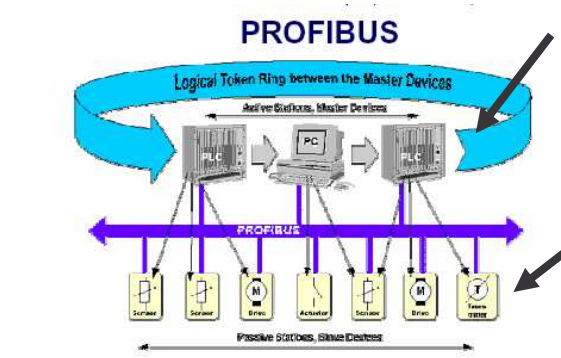
- **Broadcast** serial **bus**
- Asynchronous transmission based on UARTs
- Transmission rates up to **12 Mbit/s** over **RS-485**(-IS) on twisted pair, coaxial cable, optical fiber, power cable
- Maximum Length: **200m @ 1.5Mbit/s**, **1.2km @ 93.75kbit/s**. Extensible with repeaters
- Max. number of nodes **127** (32 masters)

PROFIBUS

- Two main application profiles:
  - **PROFIBUS / FMS** - Fieldbus Message Specification
  - **PROFIBUS / DP** - Decentralised Peripherals
    - The most common profile (~90%)
- Data **payload** between **0** and **246 bytes**
- **Direct-addressing** (1 byte, possibly extended)
- **Hybrid** bus access control
  - **Token-passing** among masters
  - **Master-Slave** in each individual data transaction

PROFIBUS

- **General architecture**

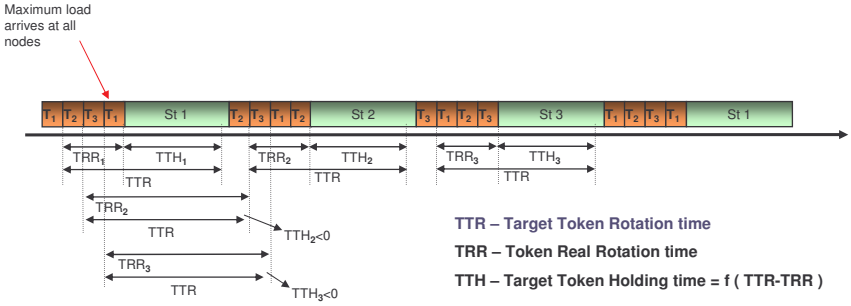


Masters: controllers, regulators, … (PC, PLC,…)

Slaves: sensors, actuators (motors, valves, thermometers…)

PROFIBUS

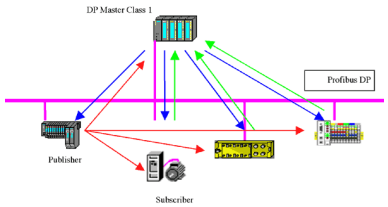
- **Traffic pattern under maximum load:**
  - Equal distribution of bandwidth among N nodes
  - Intervals of bus inaccessibility =~ TTR*(N-1)



TTR – Target Token Rotation time
TRR – Token Real Rotation time
TTH – Target Token Holding time = f (TTR-TRR)

PROFIBUS

- Recent support for: (PROFIBUS / DP-v2)
  - **Isochronous** traffic
    - **Equidistant** token arrivals forming **cycles**
    - Cycle synchronization (specific global frame - broadcast)
    - Two windows in the cycle (isochronous and async traffic)
  - **Direct slave to slave** communication
    - **Producer – Consumer** style
  - Specifically developed for high accuracy drives



CAN
Controller Area Network

www.can-cia.de

CAN

- ✓ Created by **Bosch**, GmbH, for use in the **automotive industry**.
- ✓ Version 2.0 released in **1991**.
- ✓ ISO Standards 11519 (94) e 11898 (95)
- ✓ Expanded to process control, manufacturing automation and embedded application domains
- ✓ Defines **physical** and **data link** layers
- ✓ Many application layers

CAN

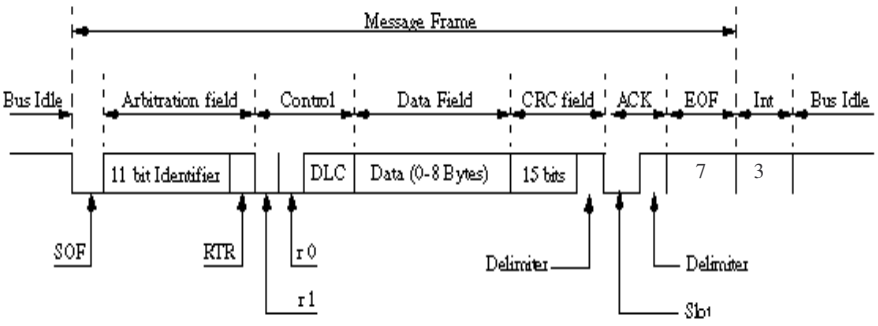
- **Multi-master**, **broadcast**, serial **bus**
- Transmission rate from **5 Kbit/s** to **1 Mbit/s**
- On transmission, nodes synchronize on bit level (requires bus spatial coherency on bit basis)
- Length depends on tx rate (aprox. 40m @ 1Mbit/s, 1000 @ 50Kbit/s)
- Max. number of nodes depends on bus transceivers (32, 64, 128…)

CAN

- Data **payload** between **0 and 8 bytes**
- **Source-addressing** (message identifiers) with 11 bits in version A and 29 bits in version B
- **Asynchronous** bus **access**
- **Non-destructive arbitration** based on message identifiers (establish priority)
  - Bit-wise deterministic collision resolution **CSMA/BA** (CA?,DCR?)

CAN

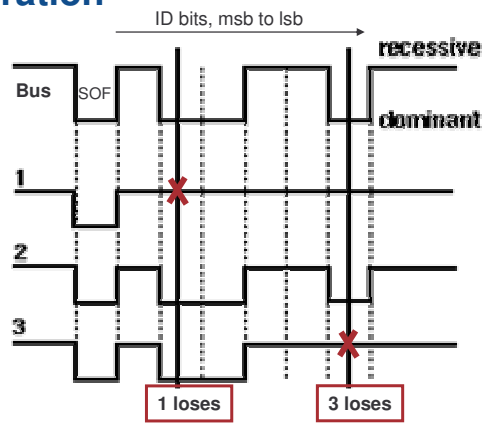
✓ CAN 2.0A message frame



CAN

✓ Prioritized arbitration

- ✓ All nodes transmit and listen every bit
- ✓ If different, then lost arbitration and backoff



Ethernet

standards.ieee.org/getieee802/802.3.html

Ethernet

- **Created in the mid 70s** (!) by Robert Metcalfe at the Xerox Palo Alto Research Center.
- Aimed initially at **sharing** expensive **computing peripherals** in office environment (particularly, it was developed to connect a printer to a computer)
- From then on, it became extremely popular, being used in many domains beyond its original scope. Particularly in **industrial systems**, it became the most serious candidate for the unifying communications protocol, from the plant floor to the management.
- Standardized in the mid 80s as **IEEE 802.3**

Ethernet

- **Multi-master**, **broadcast**, serial **bus** (initially) or **star** (currently)
- Synchronous transmission with Manchester bit coding
- Transmission rate of **10, 100Mbit/s, 1** and **10Gbit/s**
- Max. number of nodes is **1024** (normally limited by a lower number of ports of the networking equipment)
- Max. link length is 100m (100Mbit/s, UTP cat5 cabling)
- Max. of 2 hubs between any 2 nodes (100Mbit/s)
- 2 architectures: **shared** (hubs), **segmented** (switches)

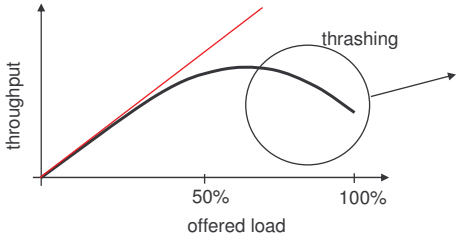
Ethernet

- **CSMA/CD** non-deterministic arbitration (used in shared Ethernet, only)
- **1-persistent** transmission (transmits with 100% probability as soon as the medium is considered free)
- Collisions can occur during the interval of one ***slot*** after start of transmission (512 bits)
- When a collision is detected a **jamming** signal is sent (32 bits long)
- Frames vary between 64 (min) and 1518 (max) octets

Ethernet

- **Thrashing effect**

  As the offered load increases,
  the actual load handled by the network
  (throughput) decreases



Maximum network-induced delays can be very long with high packet-drop rates

Ethernet

- **Motivations for using Ethernet**
  in application domains beyond its original one
  e.g. factory automation, large embedded systems
  (Decotignie, 2001)
  - Cheap wrt other high speed technologies
  - Widely available
  - Scalable tx rate
  - High bandwidth
  - Easy integration with office networks
    (important for logging, management and multi-level integration, e.g. CIM)
  - IP stacks widely available
  - Well known and mature technology

Ethernet

- **Downside of using Ethernet**
  in application domains beyond its original one
  e.g. factory automation, large embedded systems
  (Decotignie, 2001)
  - Connection costs higher than traditional fieldbuses (PHY, transformer, MAC)
  - High communication overhead (for short data items)
  - High computing power requirements (for efficient bandwidth usage)
  - Typical star topology not always adequate (may lead to extra long cabling)
  - Existing protocol stacks (mainly IP) not optimized for real-time operation

Ethernet

- **Making Ethernet real-time**
  - Keep **network load low** (~1%, e.g. NDDS) !
  - **Avoid bursts** (e.g. traffic smoothing)
  - **Modify the back off and retry** mechanism (e.g. CSMA/DCR, Virtual-time, windows, EQuB)
  - **Control transmission** instants (e.g. token-passing (RETHER, RT-EP), master-slave (FTT-Ethernet, ETHERNET-Powerlink), TDMA (PROFINET-IRT),…)
  - Micro-segment the network using **switches** (IEEE 802.1D) – Avoids collisions!

Ethernet

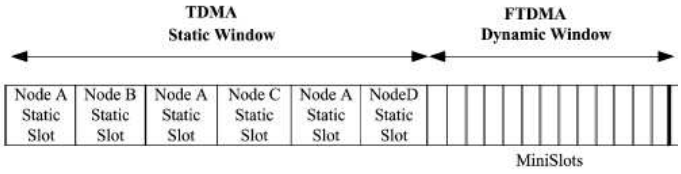
- **Using switches**
  - Became the **most common** solution
    - Current switches are *wire-speed* (non-blocking)
    - 802.1D – possibly with multiple priority queues (802.1p)
    - 802.1Q – Virtual LANs
  - **Not perfect !**
    - **Priority inversions** in queues (normally FIFO)
    - **Mutual interference** through shared memory and CPU
    - **Additional forwarding delay** (with jitter caused by address table look up, address learning, flooding)
    - **Delays vary** with switch technology and internal traffic handling algorithms

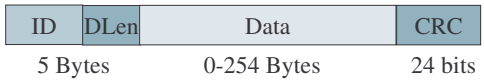
Other protocols

FlexRay

www.flexray.com

- ✓ **Created in the early 2000s** by an industrial consortium from the automotive domain.
- ✓ First public specifications available since 2004 (several changes before that)
- ✓ Is expected to replace CAN and contend with TTP/C !
- ✓ Tx rate up to 10Mbit/s



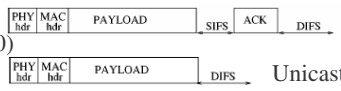


WiFi (802.11)

standards.ieee.org/getieee802/802.11.html

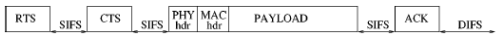
- ✓ **Created in the 1990s** for the SOHO domain. As Ethernet, 802.11 became **widely used** as **wireless** communication **infrastructure**.
- ✓ 3 modes/bands: 802.11b/g (ISM-2.4GHz), 802.11a (5GHz)
- ✓ Scalable Tx rates between 1Mbit/s and 54Mbit/s
- ✓ Many mechanisms to **reduce collisions** and **hidden-terminals**
- ✓ Growing interest for **QoS support** (initial PCF mode was abandoned, currently there is a new mode, **802.11e**)

(Bianchi, 2000)



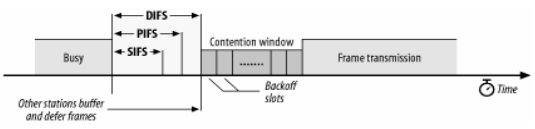
Unicast acknowledged frame

Unicast/Broadcast unacknowledged frame



RTS/CTS for reduction of hidden terminals

(Gast, 2005)



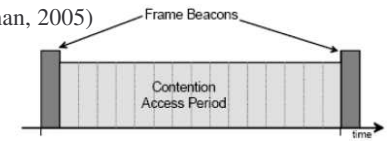
CSMA-CA

ZigBee

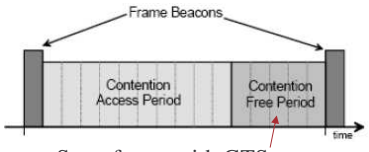
www.ZigBee.org

- **Created in the early 2000s** for the **wireless monitoring and control**. Targets **wireless sensor networks.**
- Works on top of 802.15.4, a PAN DLL. 3 bands: ISM-2.4GHz, 868/915 MHz. Targets very low consumption.
- Data rate of **250 Kbit/s** with range up to 300m
- **Beacon mode** and possible **PAN coordinator** to structure cells and QoS support. **Structured** and **peer-to-peer** modes. Routing support.
- Up to **65K nodes** (full and reduced function devices)

(Raman, 2005)



Superframe without GTS (Guaranteed Time Slot)



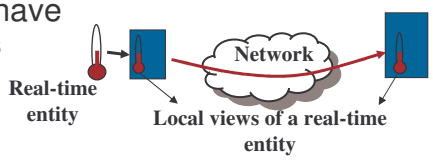
Superframe with GTS (Guaranteed Time Slot)

Part 6

Traffic model and scheduling issues

Real-time messages (revisited)

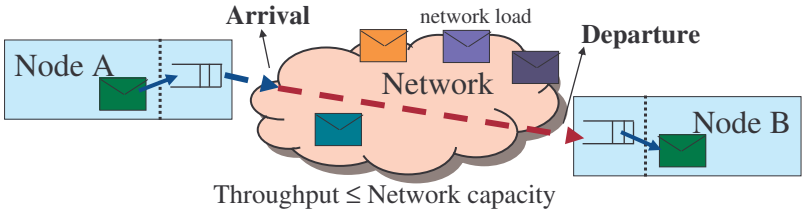
- A message related to a *real-time entity* (e.g. a sensor) is a **real-time message**.
- Real-time messages must be transmitted within precise **time bounds (deadlines)** to assure **coherence** between senders and receivers concerning their **local views** of the respective real-time entities
- Real-time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



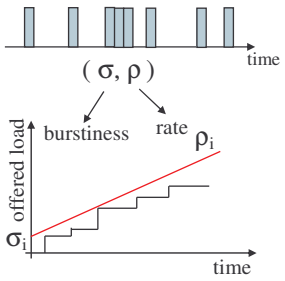
Purpose of traffic scheduling

- Traffic scheduling establishes the **order** in which the traffic is dispatched
- The **traffic scheduling** algorithm is essentially executed at the **data link level** (determined by the **MAC** and by local **queuing** policies), as well as at the **network layer** (routing queues) if existing
- It can be distributed, or centralized in a particular node.
- In real-time systems it is important to check at design time if **deadlines** of real-time messages **will be met**
  - This is called **schedulability analysis**

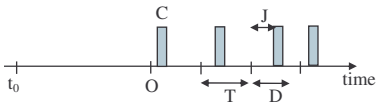
Traffic models



(σ,ρ)-model
(Cruz, 1991)



Recurrent model

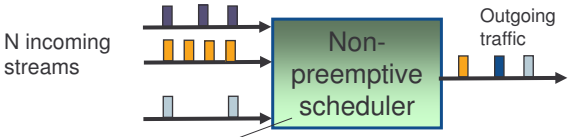


(C, D, T/mit, (P), (O), J)

WC tx time
Deadline
Period
Priority
Offset
Release jitter

Traffic scheduling model

$M \equiv \{m_i (C_i,T_i,J_i,D_i,P_i,O_i), i=1..N\}$



Scheduling policies
- EDF
- RM
- DM
- FCFS
- WFQ
- …

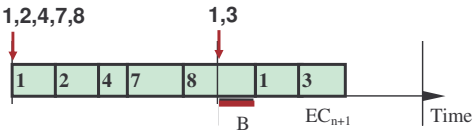
Knowing the **scheduling policy** and the **arrival pattern of the incoming** flows allows determining the **departing pattern** of the **outgoing flow**

Using bandwidth utilization

(recurrent model)

- ✓ Based on **bandwidth utilization** ($U_i=C_i/T_i$)
  - ✓ **EDF: Earliest Deadline First** (Dynamic priorities, D=T)
  - ✓ **RM: Rate-Monotonic Scheduling** (Fixed priorities, D=T)

Preemptive task scheduling with **non-preemption blocking**



B – non-preemption blocking
1..8 - messages

RM: $\sum_{1}^{N}\frac{C_i}{T_i}+\max_{1..N}\left(\frac{B_i}{T_i}\right)<N(2^{1/N}-1)$

EDF: $\sum_{1}^{N}\frac{C_i}{T_i}+\max_{1..N}\left(\frac{B_i}{T_i}\right)<1$

$B_i=\max_{l=j..N}(C_l)$

j is first var that can cause blocking

If these conditions are met then **one transmission is guaranteed every period**

Using bandwidth utilization

(recurrent model)

- ✓ Based on **bandwidth utilization** ($U_i=C_i/T_i$)
  - ✓ **EDF: Earliest Deadline First** (Dynamic priorities, D=T)
  - ✓ **RM: Rate-Monotonic Scheduling** (Fixed priorities, D=T)

More accurate tests:

RM: $\forall_{i=1..N}\sum_{j=1}^{i}\frac{C_j}{T_j}+\frac{B_i}{T_i}<i(2^{1/i}-1)$

EDF: $\forall_{i=1..N}\sum_{j=1}^{i}\frac{C_j}{T_j}+\frac{B_i}{T_i}<1$

$B_i=\max_{l=j..N-1}(C_l)$

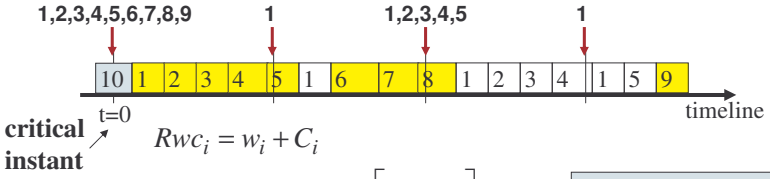
j is first var that can cause blocking

If these conditions are met then **one transmission is guaranteed every period**

Response-time analysis
(recurrent model)

- Based on **a response time upper bound** (Rwc_i)
  - **FP: Arbitrary fixed priorities** (D≤T, J=0)

Consider the following set of 10 messages with periods given by T_1=1, $T_{2..5}$=2, $T_{6..10}$ >3



critical instant

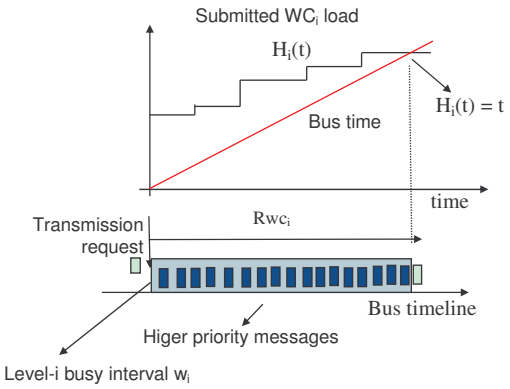
$$Rwc_i = w_i + C_i$$

$$w_i = B_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{w_i + \tau}{T_j} \right\rceil * C_j$$

$\forall_i$ $Rwc_i \le D_i$ => all deadlines are met

Response-time analysis
(recurrent model)

- **Response time upper bound** (graphical view)



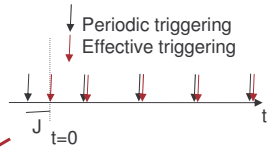
$$Rwc_i = w_i + C_i$$

$$H_i(t) = B_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{t + \tau}{T_j} \right\rceil * C_j$$

$$w_i = 1^{st}\, t : H_i(t) = t$$

Response-time analysis
(recurrent model)

- Based on **a response time upper bound** (Rwc_i)
  - **FP: Arbitrary fixed priorities** (Static priorities, D≤T, J≠0)
  - Impact of **release jitter** (J_i)



$$Rwc_i = J_i + w_i + C_i$$

$$w_i = B_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{J_j + w_i + \tau}{T_j} \right\rceil * C_j$$

Response-time analysis
(recurrent model)

- Based on **a response time upper bound** (Rwc_i)
  - **FP: Arbitrary fixed priorities** (Static priorities, D>T, J≠0)
    (arbitrary deadlines)

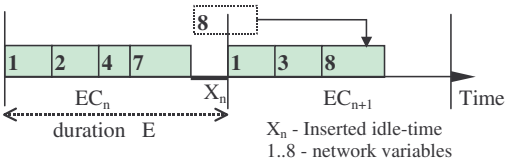
$$Rwc_i = \max_{q=0,1,2...} \left(J_i + w_i(q) - qT_i + C_i\right)$$

$$w_i(q) = B_i + qC_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + w_i(q) + \tau}{T_j} \right\rceil C_j$$

The *level-i busy interval* includes **q instances** of message i
q gives an approximation of **buffer requirements** for message i

Scheduling within slots/cycles
(recurrent model)

- When using slots or cycles whose duration must be strictly respected (e.g., TDMA, mini-slotting), it is necessary to use **inserted idle time** (X)
  - However, there is **no more blocking**!
  - In this case, **any analysis** for **preemptive** scheduling can be used with inflated transmission times (C')



$$C_i' = C_i * \frac{E}{E - X_{max}}$$

Inserted idle-time compensation factor

One of our humble contributions ☺

Scheduling within slots/cycles
(recurrent model)

- **Utilization based tests** ($U_i = C'_i/T_i$)
  - **EDF: Earliest Deadline First** (Dynamic priorities, D=T)
  - **RM: Rate-Monotonic Scheduling** (Fixed priorities, D=T)
  - Without Blocking
  - With Release Jitter (J)

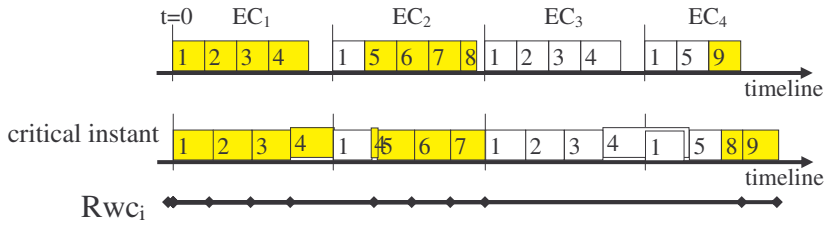
RM: $$\forall_{i=1..N} \sum_{j=1}^{i} \frac{C'_j}{T_j} + \frac{\max_{j=1..i}(J_j)}{T_i} < i(2^{1/i} - 1)$$

EDF: $$\forall_{i=1..N} \sum_{j=1}^{i} \frac{C'_j}{T_j} + \frac{\max_{j=1..i}(J_j)}{T_i} < 1$$

A very recent contribution ☺ - applicable to general preemptive scheduling

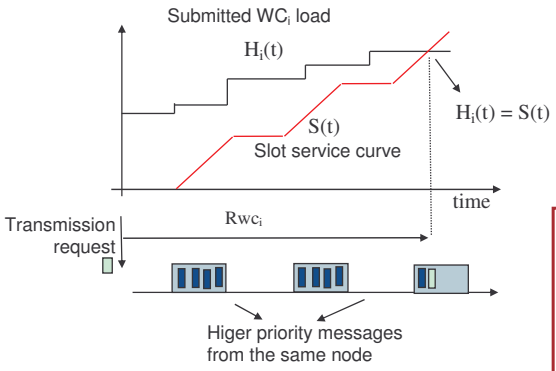
Scheduling within slots/cycles
(recurrent model)

- **Response time analysis** with fixed priorities can also be used, given the *inserted idle-time compensation* factor and **without blocking**
  - Consider the following set of 9 variables with periods given by T_1=1, $T_{2..5}$=2, $T_{6..9}$ >3



Scheduling within slots/cycles
(recurrent model)

- **Response time upper bound** (graphical view)



$$Rwc_i = w_i + C'_i$$

$$H_i(t) = \sum_{j\ in\ hp(i)} \left\lceil \frac{t+\tau}{T_j} \right\rceil * C'_j$$

$$w_i = 1^{st} t : H_i(t) = S(t)$$

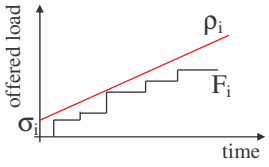
Another of our humble contributions ☺

Network Calculus

((σ,ρ)-model)

- **Flexible traffic model**
  - Cumulative arrival at queue *i* (F_i) upper bounded by (σ_i,ρ_i): $F(t) - F(s) \leq \sigma_i + \rho_i*(t-s) \quad \forall_{0 \leq s \leq t}$
  - Decreasing priorities with *i*
  - Channel capacity *c*
  - Upper bound on queue *i* delay D_i

(σ_i,ρ_i) →TSPEC



$$D_i = \frac{\sum_{j=1}^{i} \sigma_j + \max_{i+1 \leq j \leq N} (C_j)}{c - \sum_{j=1}^{i-1} \rho_j}$$

Stability condition

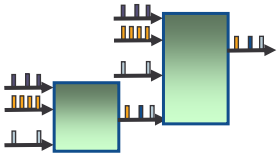
$$c \geq \sum_{j=1}^{N} \rho_j$$

Network Calculus

((σ,ρ)-model)

- **Network calculus** is generally **more pessimistic** than **response time analysis**
- However, it is **more general** (applicable to arbitrary traffic patterns) and allows determining a bound to the burstiness of the outgoing traffic (σ'_i) and consequently to the **buffer requirements** of that flow
- This is very important for **hierarchical composition** of network segments

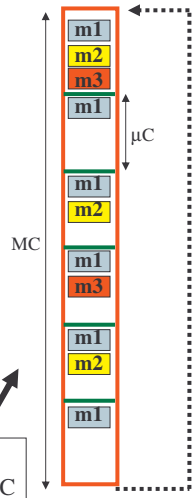
$$\sigma'_i = \sigma_i + \rho_i * \frac{\sum_{j=1}^{i-1} \sigma_j + \max_{i+1 \leq j \leq N} (C_j)}{c - \sum_{j=1}^{i-1} \rho_j}$$



Using scheduling tables

- Some systems use **static table based** scheduling (e.g., WorldFIP, TTP/C…)
- The schedule is **built off line** and it is typical to use optimization techniques to **optimize the schedule**
  - e.g. wrt to **jitter** or **end-to-end delays**

  controlling the offsets.
- The table contains a number of micro-cycles that form a Macro-cycle, which is repeated in an infinite loop



μC = MDC(T_i) (*GCD*)
MC = MMC(T_i) (*LCM*)

$O_i = 0, C_i = 1ms$,
$T_1 = 5ms$
$T_2 = 10ms$
$T_3 = 15ms$

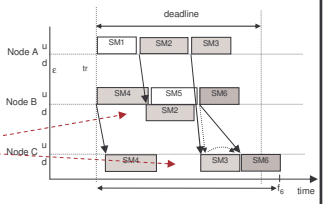
GCD=5ms→μC
LCM=30ms→MC
MC={6μC}

Constraints imposed by the MAC

- **Minimum** transmission **period,** e.g., TDMA round cycle, or microcycle in Master-Slave.
- **Jitter** in Token-Passing systems, due to the irregularity of token arrivals.
- **Blocking** term in asynchronous systems (no offset or phase control).
- **Inserted idle time** in synchronous systems with variable size data.
- **Dead interval** in polling systems (e.g. Master-Slave, Token-Passing) to handle aperiodic communication requests.

Further constraints

- Several forms of **Blocking.**
  - **FIFO queues** at the AL
    - Attention to the **protocol stacks**!
  - **FIFO queues** at the DLL
    - Attention to the **device drivers**!
  - **Causal effects** in 1xN switching
  - ...



- Some of these may also cause **release jitter**
- **FIFO** queues have **poor temporal** behavior and can lead to **large blocking** periods
- **Response time analysis for FIFO** queues is still needed

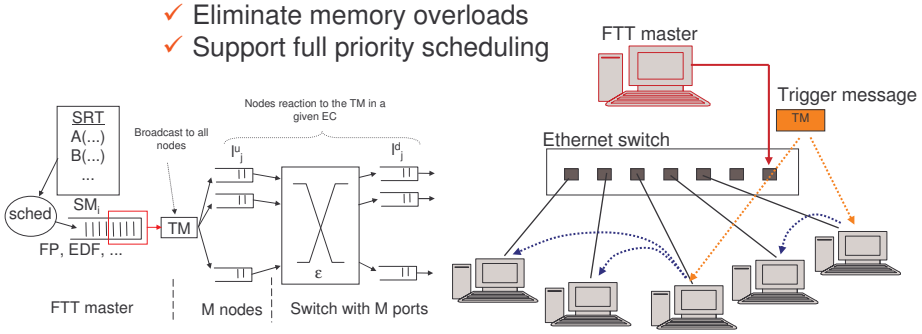
Further constraints

- More issues that need further attention
  - **Combination of periodic and aperiodic traffic**
    - With temporal isolation → handle aperiodic traffic with servers
  - **Robust communication**
    - On-line traffic scheduling with admission control and traffic policing
    - Adaptive mechanisms to provide best-effort communication under uncontrolled interference
  - **Response-time analysis for switches**
    - Bears some resemblance to multi-processor scheduling
  - **Jitter control at the device-drivers level**
  - **Composition of segments**

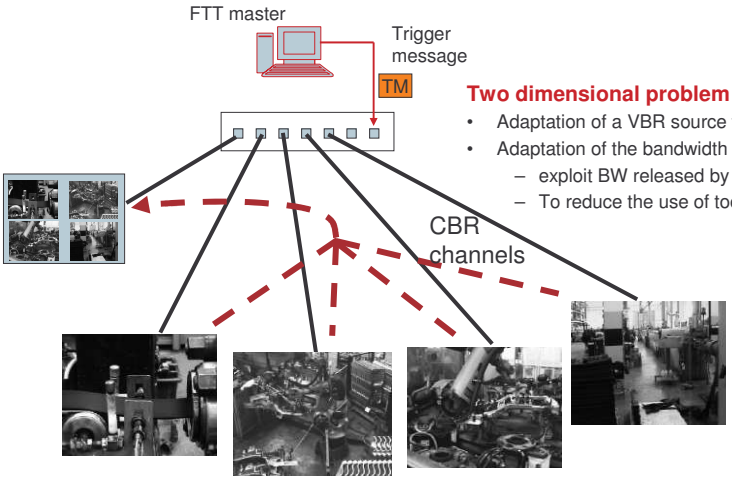
Part 7
Some related on-going research

FTT-SE

- **Keeping under control** the traffic **load** submitted to a switched network
  - Schedule traffic per cycles
  - Submit only the traffic that fit in a cycle
  - Eliminate memory overloads
  - Support full priority scheduling



QoS adaptation with FTT-SE



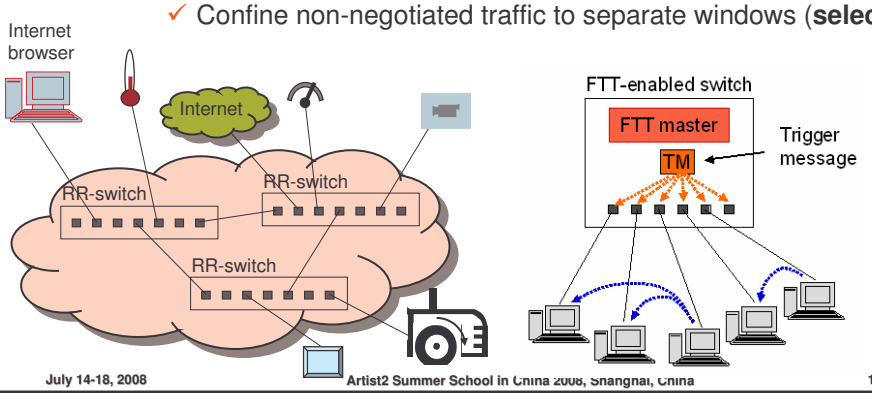
Two dimensional problem

- Adaptation of a VBR source to a CBR channel
- Adaptation of the bandwidth of the CBR channels
  - exploit BW released by streams that are off
  - To reduce the use of too strong compression

MJPEG encoders – VBR bit streams

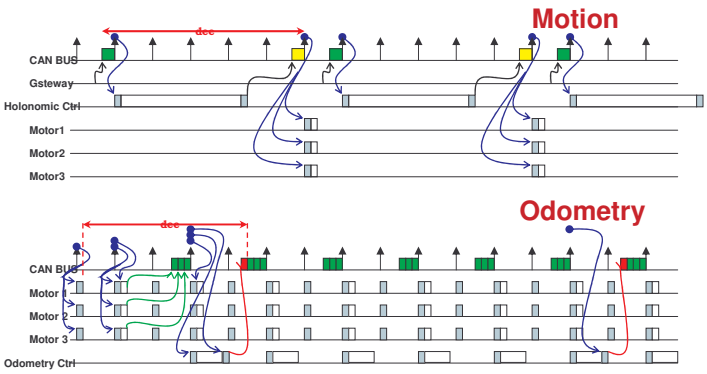
Resource-reservation switch

- ✓ Providing timeliness, flexibility and high robustness in switched Ethernet networks
  - ✓ Enforce negotiated channel characteristics (**policing**)
  - ✓ Reject abusive negotiated traffic (**filtering**)
  - ✓ Confine non-negotiated traffic to separate windows (**selection**)



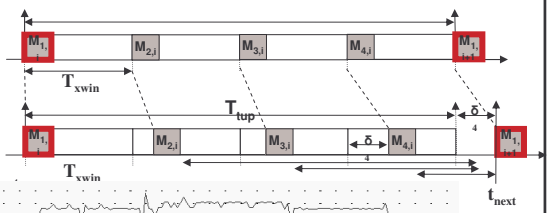
FTT-CAN

- ✓ Providing timeliness (synchronization) and flexibility in embedded CAN networks

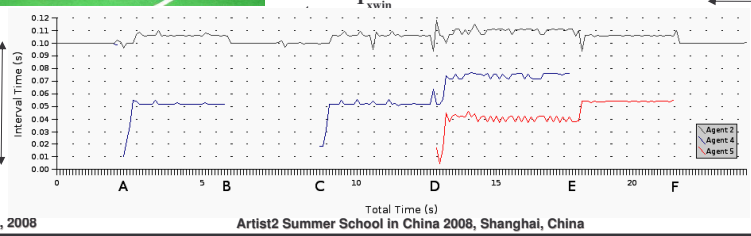


Robust IEEE802.11 communication

- ✓ Providing robust IEEE 802.11 communication for teams of autonomous mobile robots
  - ✓ Reconfigurable and adaptive TDMA to cope with uncontrolled traffic

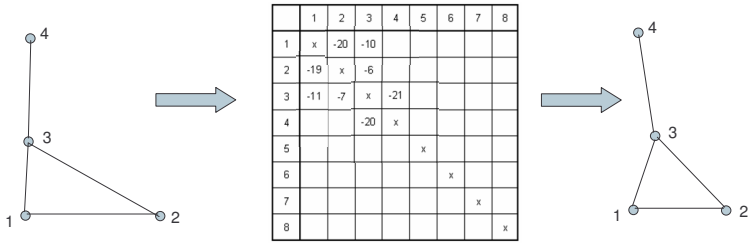


TDMA round with dynamic # of slots



Relative localization with IEEE802.15.4

- Using IEEE 802.15.4 communication to
  - **Discover and track the topology** of a team of autonomous mobile robots
  - **Find their relative positions** for team coordination



Physical locations | Topology matrix | Estimated relative positions

Conclusion

- The **network** is a **fundamental** component within a distributed system (supports system integration)
- **Real-time operation** requires **time bounded** communication
  - appropriate **protocols** must be used
- **Growing distribution, integration** and **flexibility** in distributed embedded systems require more support from the network to **maintain temporal guarantees, safety** and **robustness**
- Many analysis exist to support timeliness in the network and thus in the system. However, several situations still require further attention to provide the **required levels of robustness**

Bibliography

- N. Navet, F. Simonot-Lion (eds.). ***Automotive Embedded Systems Handbook***. CRC Press, 2008.
- Richard Zurawski (ed.). ***The Industrial Communication Systems Handbook***. CRC Press, 2005.
- B. Bouyssounouse, J. Sifakis (eds.) ***Embedded Systems Design, The ARTIST Roadmap for Research and Development.*** LNCS 3436, Springer 2005.
- Le Boudec, J.-Y., Thiran, P., **Network Calculus: a theory of deterministic queuing systems for the Internet**. Springer-Verlag, Vol. 2050, 2001 (available for free download).
- 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

- J.-D. Decotignie. **Ethernet-based Real-time and Industrial Communications**. In Proceedings of the IEEE, volume 93, pages 1102–1117, June 2005.
- 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.
- J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. **Realtime communication and coordination in embedded sensor networks**. In Proceedings of the IEEE, volume 91, pages 1002–1022, July 2003.
- P. Koopman. **Critical Embedded Automotive Networks.** *IEEE Micro*, IEEE Press, July/August 2002.
- J.-D. Decotignie. **Wireless fieldbusses – a survey of issues and solutions**. In Proc. 15th IFAC World Congress on Automatic Control (IFAC 2002), Barcelona, Spain, July 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.