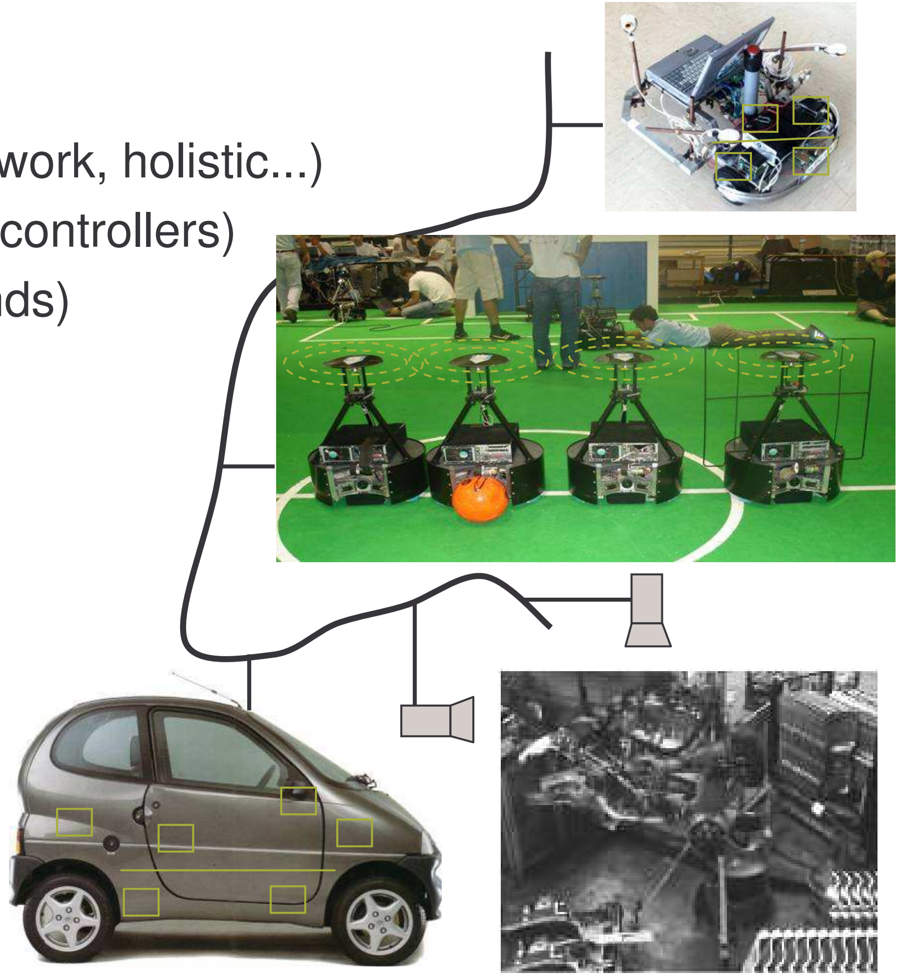
ARTIST Summer School in China
IOS-ISCAS - Beijing, China
August 8-12, 2011

Real-Time Communication in Embedded Systems

Luis Almeida
lda@fe.up.pt
IT – Porto, **IEETA** – Aveiro
DEEC – **University of Porto**, Portugal

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

ieeta U.PORTO artist

In this course

- Distribution-related trends in Embedded Systems
- Timing issues in communications
- Inside the protocol stack

First day

- Current technologies

(skip, slides only)

- Analyzing network delays
- Some practical examples

Second day

The *information* era

- It's all about **information**!
- We need to:
  - **Hold information** — **Static** (time invariant)
  - **Process information** → Transformation in the **value domain** (logic)
  - **Transmit information** → Transformation in the **space domain**

  Dynamic (need time!)

Here we deal with

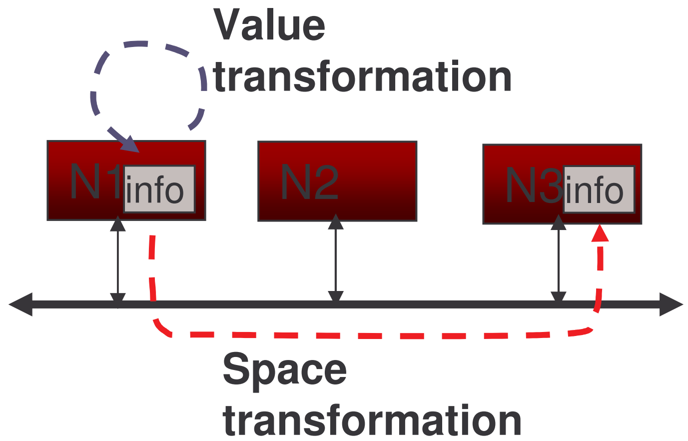
Information transmission

The operator for this transformation is

→ The network

And for time-bounded transformations

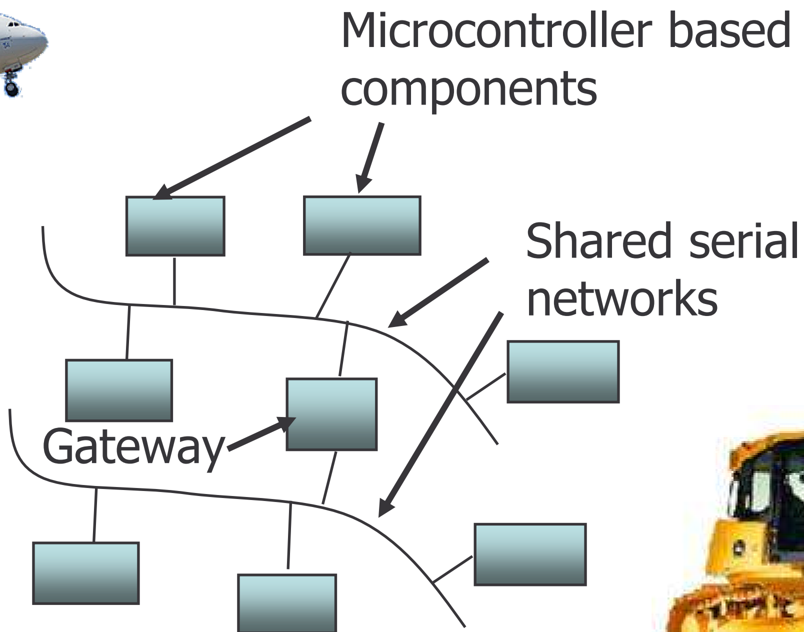
→ Real-Time Networks !

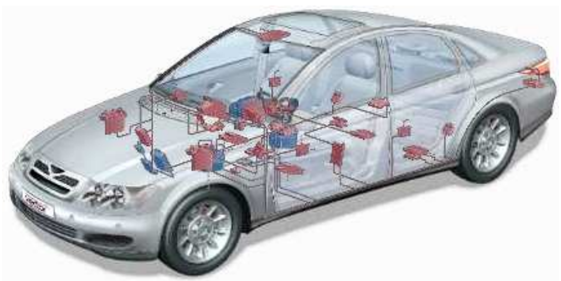


Networks everywhere

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

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



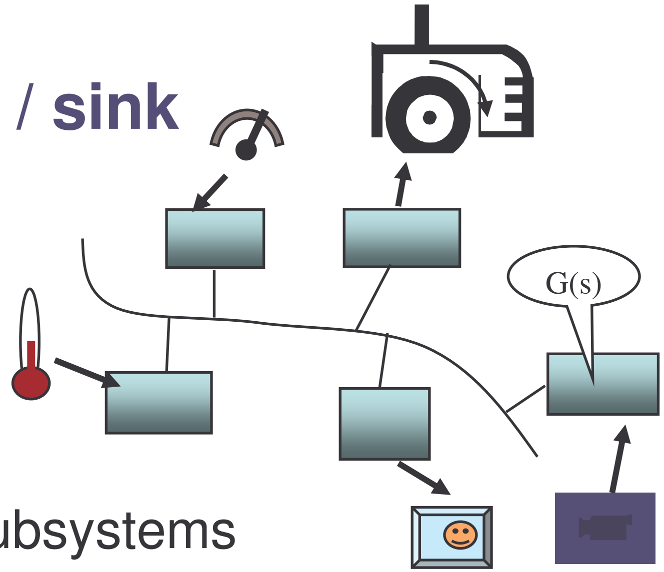


Networks everywhere

Networks for all sizes and scales

- ✓ **NoCs** – connecting processors inside MPSoCs
- ✓ **SPI, I2C**... – connecting discrete components inside boards
- ✓ **USB, FireWire**... – connecting peripherals around a PC
- ✓ **Bluetooth, RFID**... – connection of peripherals or sensors in small areas (BANs, PANs ...)
- ✓ **SCSI, SCI**... - High speed connection of servers in server farms (SANs)
- ✓ **CAN, fieldbuses**... – connection of sensors, actuators and controlling equiment in a monitoring or control plant (DCCS)
- ✓ **Zigbee, low power radios**... – connection of autonomous dispersed sensors (WSNs)
- ✓ **Ethernet, WiFi**... – connection of PCs, and independent equipment in a local setup (LANs)
- ✓ **10G Ethernet, ATM** ... – connection of large systems in large areas (MANs, WANs)
- ✓ **Telecommunication networks** – global communications(MANs,WANs)

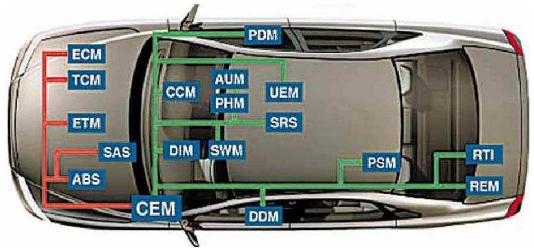
Why distributed architectures

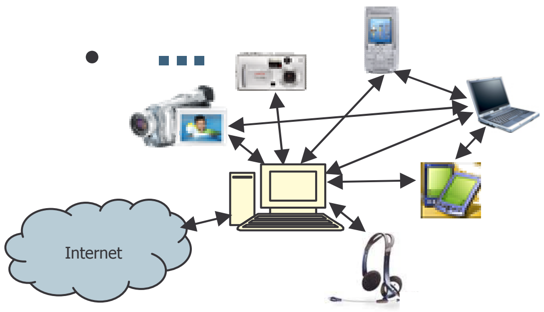


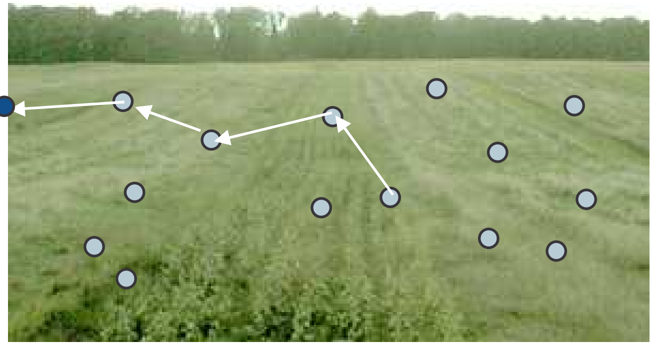
- **Processing closer to data source / sink**
  - Intelligent sensors and actuators
- **Dependability**
  - Error-containment within nodes
- **Composability**
  - System composition by integrating subsystems
- **Scalability**
  - Easy addition of new nodes with new or replicated functionality
- **Maintainability**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

Many related networking frameworks

- Distributed embedded systems (DES)
- Networked embedded systems (NES)
- Ubiquitous systems
- Wireless sensor networks (WSN)
- Mobile ad-hoc networks (MANET)
- Opportunistic networks
- …

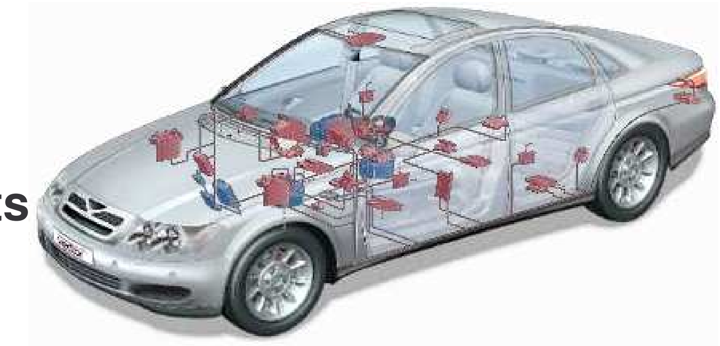






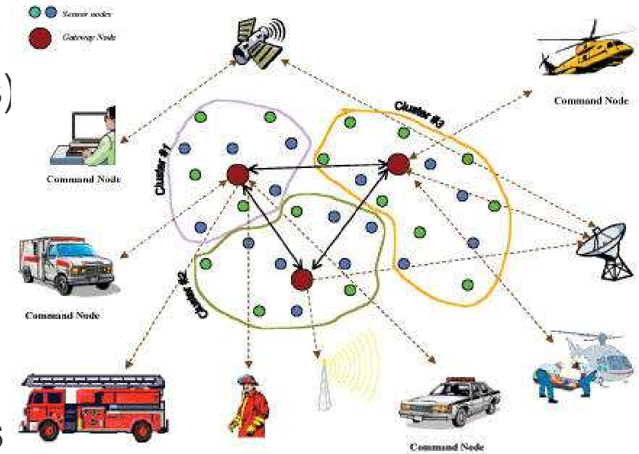
Distributed vs networked

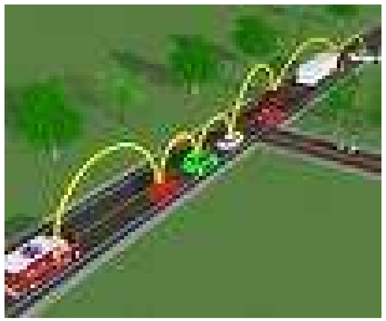
- **Distributed Embedded Systems**
  - **System-centered** (designed as a whole)
    - Confined in space (despite possibly large)
    - Normally fixed set of components
    - **Preference for wired networks**
      - **Fixed topology**
      - **Most interactions in single-hop segments**
  - Most frequent **non-functional requirements**
    - Real-time
      - End-to-end constraints on response to stimuli
      - Jitter constraints on periodic activities
    - Dependability
      - Ultra high reliability and safety, high availability
    - Composability
    - Maintainability

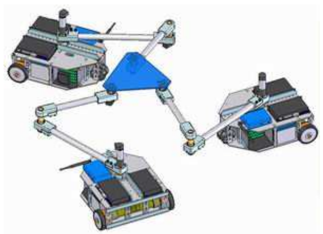


Distributed vs networked

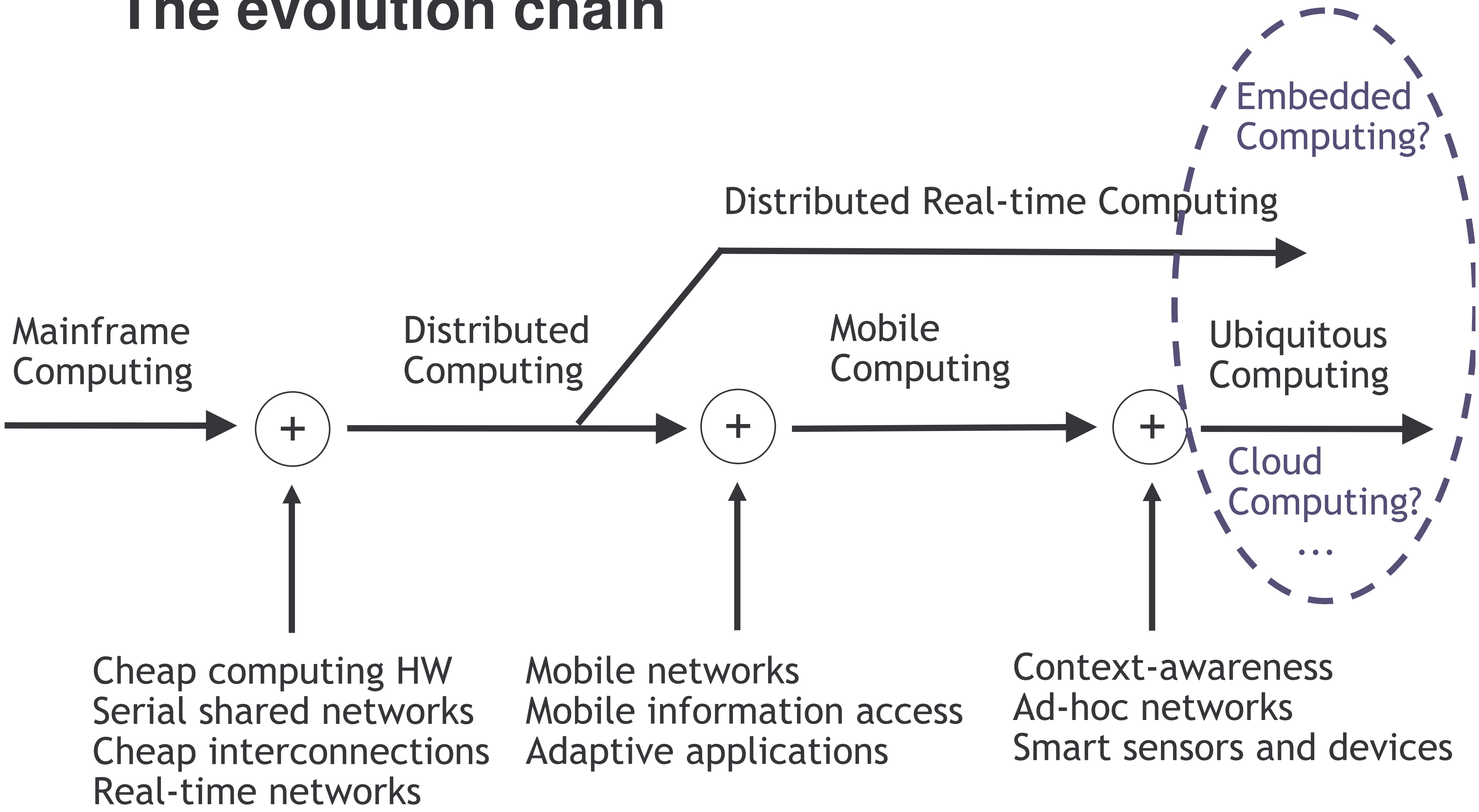
- **Ubiquitous / networked Embedded Systems**
  - **Communication-centered** (Interconnected stand-alone equipment for extra functionality)
    - Fuzzy notion of global system (and its frontiers)
      - too large / variable set of components
    - **Preference for wireless networks**
      - **Structured / Ad-hoc connections**
      - **Varying topology**
      - **Multi-hop communication**
  - Most common **non-functional requirements**
    - Scalability
    - Heterogeneity
    - Self-configuration
    - (Soft) real-time







The evolution chain



Adapted from
(Strang and Linnhoff-Popien, 2004)

Communication requirements

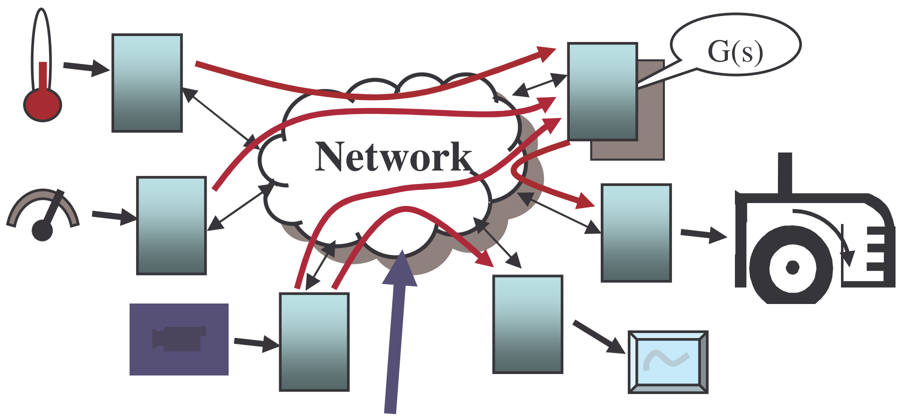
- **In both previous cases (DES + U/NES)**
  - Efficient transmission of **short data** (few bytes)
  - **Periodic** transmission (monitoring, feedback control) with **short periods** (ms-s) and **low jitter**
  - **Fast transmission** (ms-s) of **aperiodic requests** (alarms)
  - Transmission of **non-real-time data** (configuration, logs)
  - **Multicasting**

State messages

Event messages

Real-time messages

! There is also a growing interest on **real-time multimedia traffic** (e.g., machine vision)
➔ **long** RT data



Bandwidth:
Limited shared resource

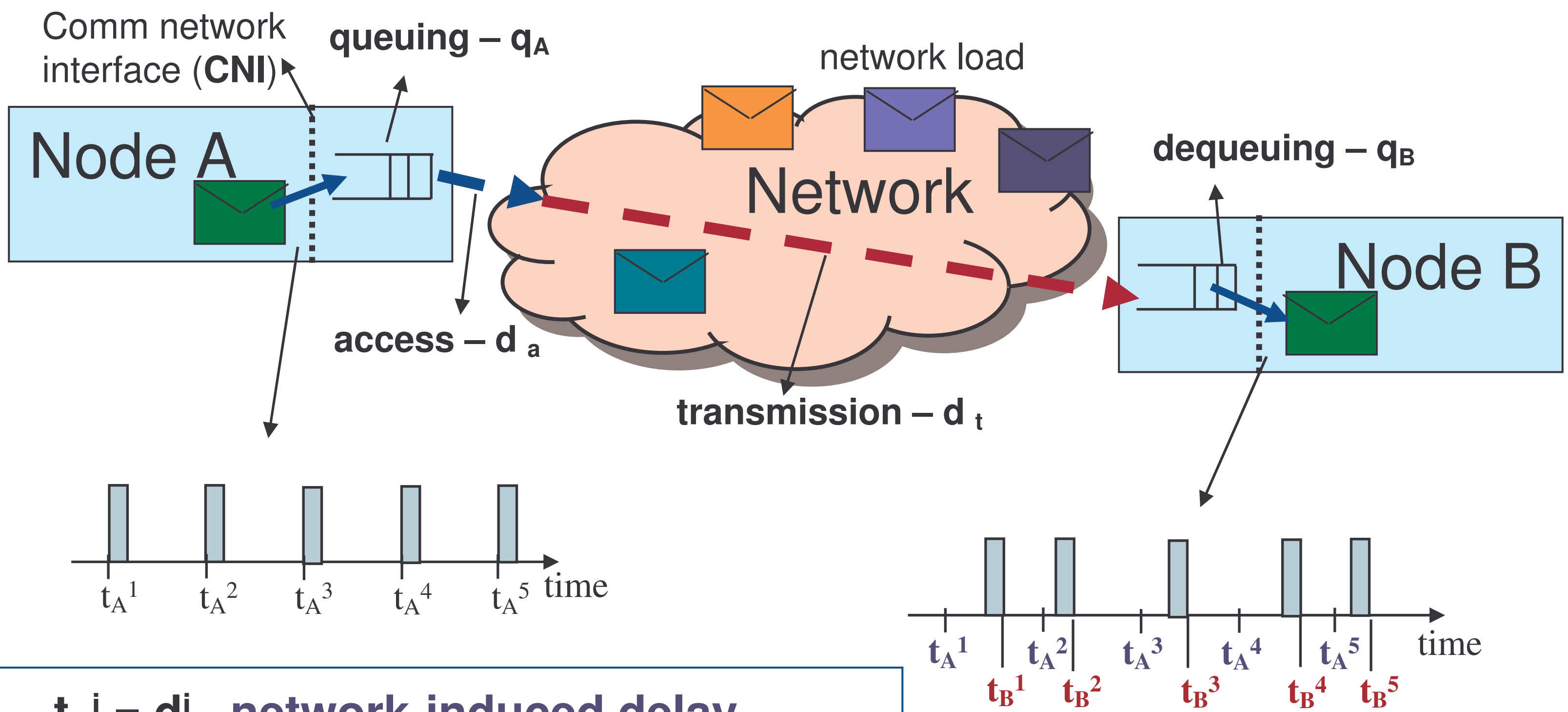
Rewinding...

Rewinding…

- ✓ **Networks allow sharing information**
  - ✓ Are the **backbone** of modern systems (**DES, NES, US, WSN …**)
- ✓ **Current trends**
  - ✓ Dissemination of **feedback control**
  - ✓ Towards **functional integration**
  - ✓ Towards **operational flexibility**
- ✓ **But**
  - ✓ Networks have a finite bandwidth → **transmission takes time**
  - ✓ Networks are typically shared → **interference among transmissions**
  - ✓ Networks suffer EMI → **transmission errors and packet losses**

Timing Issues in the Network

Network delay and delay jitter



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

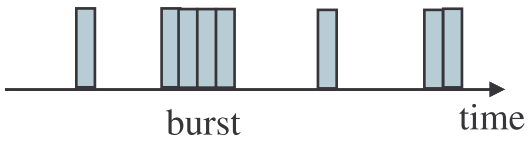
$d^i = q_A^i + d_a^i + d_t^i + q_B^i$, **delay components**

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

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

Burstiness and throughput

- **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.

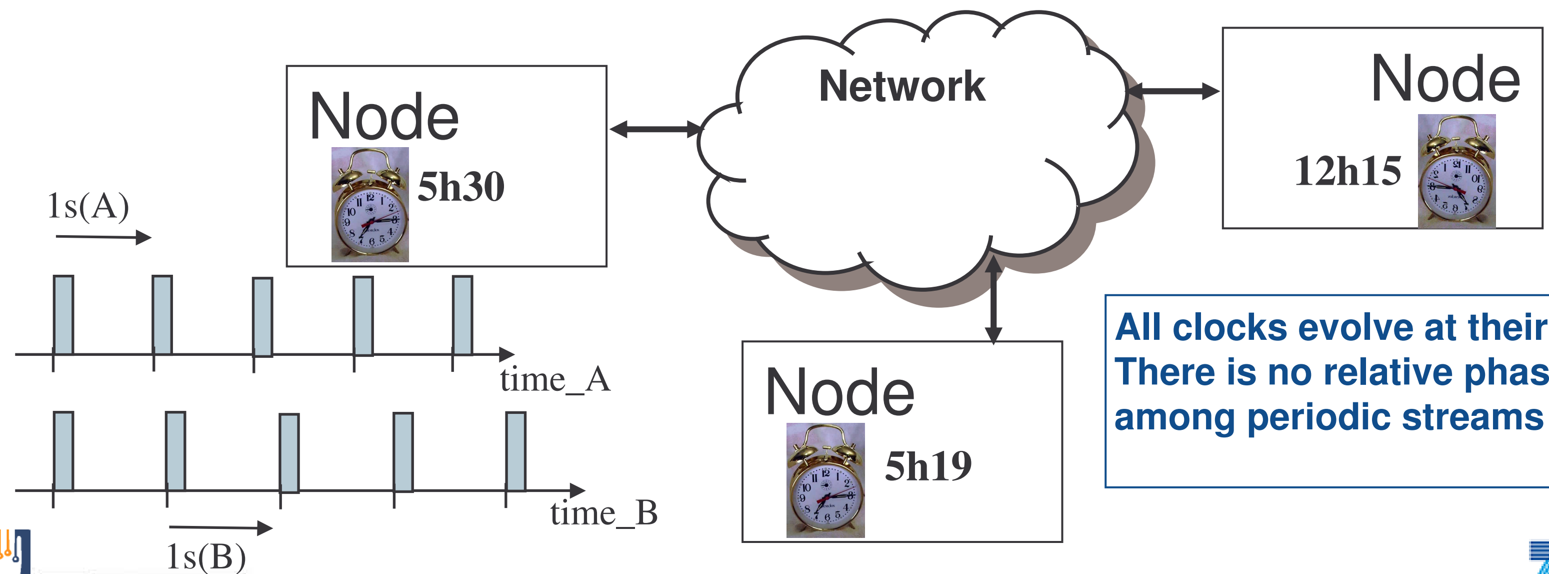


- **Throughput – average amount of data, or packets, that the network dispatches per unit of time (bit/s and packets/s).**
- **Arrival / departure rate – long-term rate at which data arrives at/from the network (bit/s and packets/s).**
- **Capacity – maximum (gross) bit-rate ... the network**

Network bandwidth

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

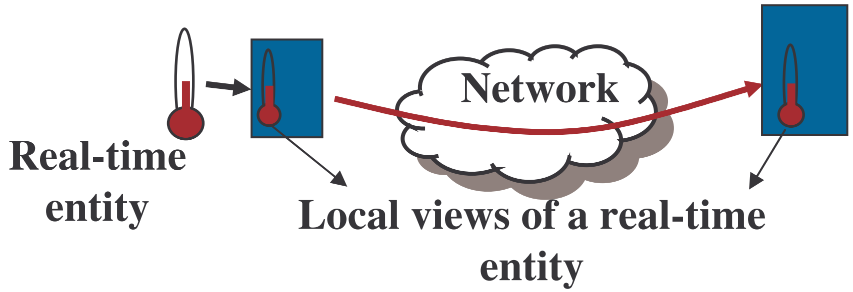
But how to synchronize the clocks across the network?

Synchronizing clocks

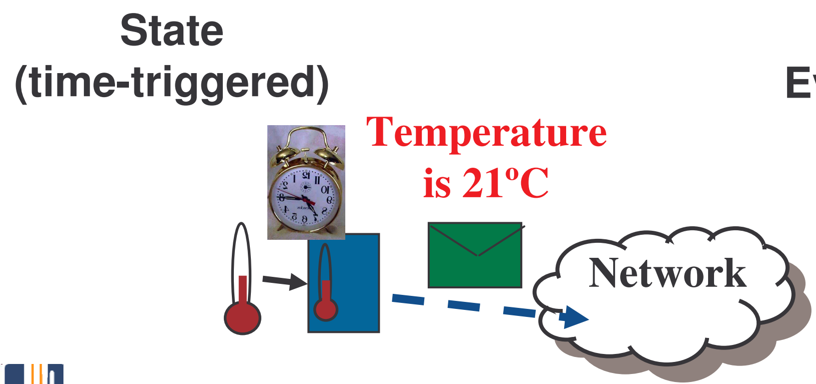
- **Clocks can be synchronized:**
  - **Externally** – an external source sends a time update regularly (e.g. GPS)
  - **Internally** – nodes exchange messages to come up with a global clock
    - **Master-Slave** – The time master spreads its own clock to all other nodes
    - **Distributed** – All nodes perform a similar role and agree on a common clock, for example, using an average (e.g. FTA, Fault-Tolerant Average)
- Standards: **NTP, SNTP, IEEE 1588**
- **Uncertainties in network-induced delay** lead to limitations in the achievable **precision**
  - Typical precision with SW methods in small networks is worse than 10μs
    - In LANs it is common to achieve 1-5ms precision
  - With special HW support, it is possible to reach 1μs or better

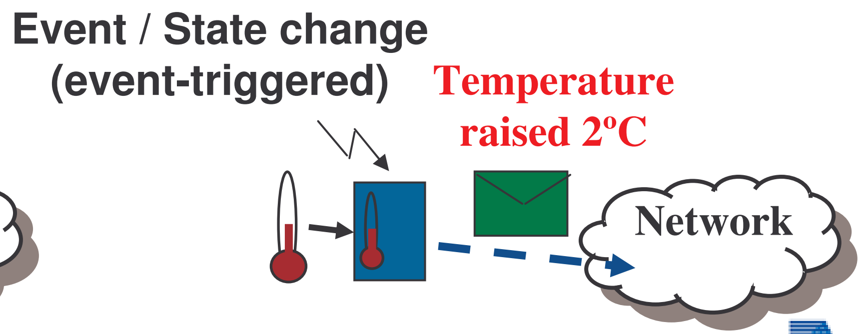
Constraints on the network delay

- **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



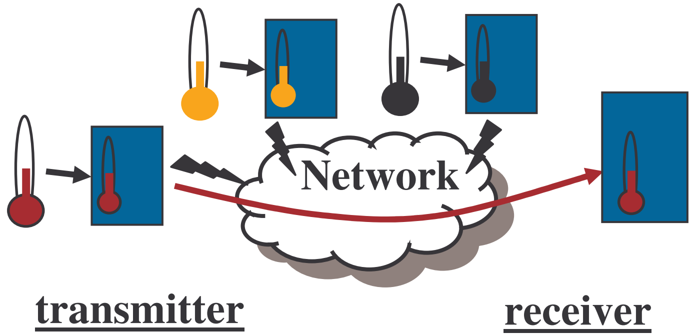
- This applies to both **event and state messages**

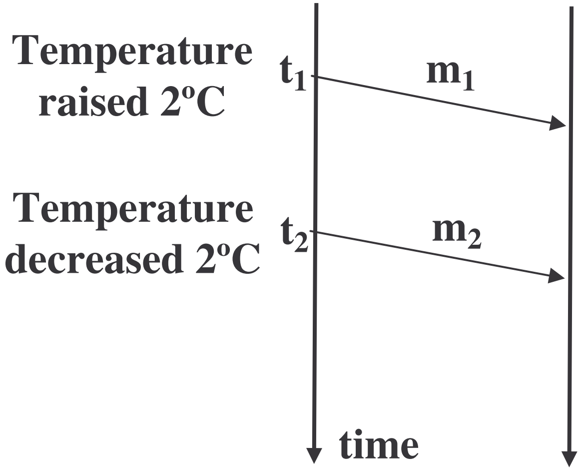




Event and time-triggered messages

Event-triggered

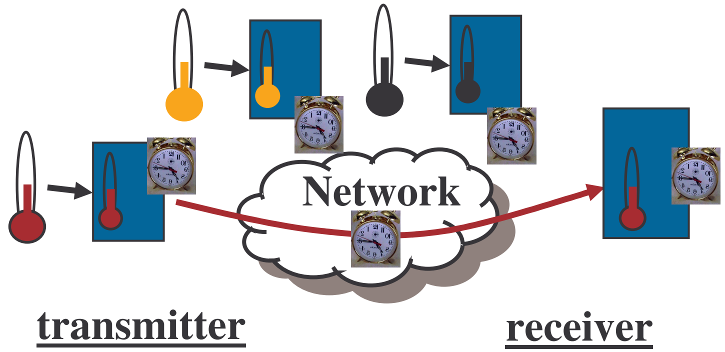


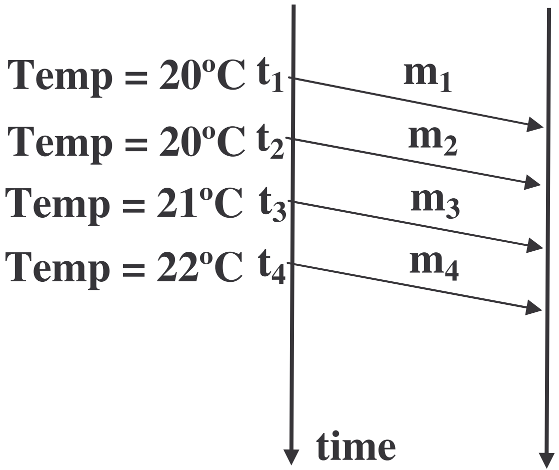


What if:
- m_2 is lost?
- or delayed?
- or the transmitter crashed?
- or nothing happened?

Typical solutions force transmissions every Δt (heartbeats)

Time-triggered

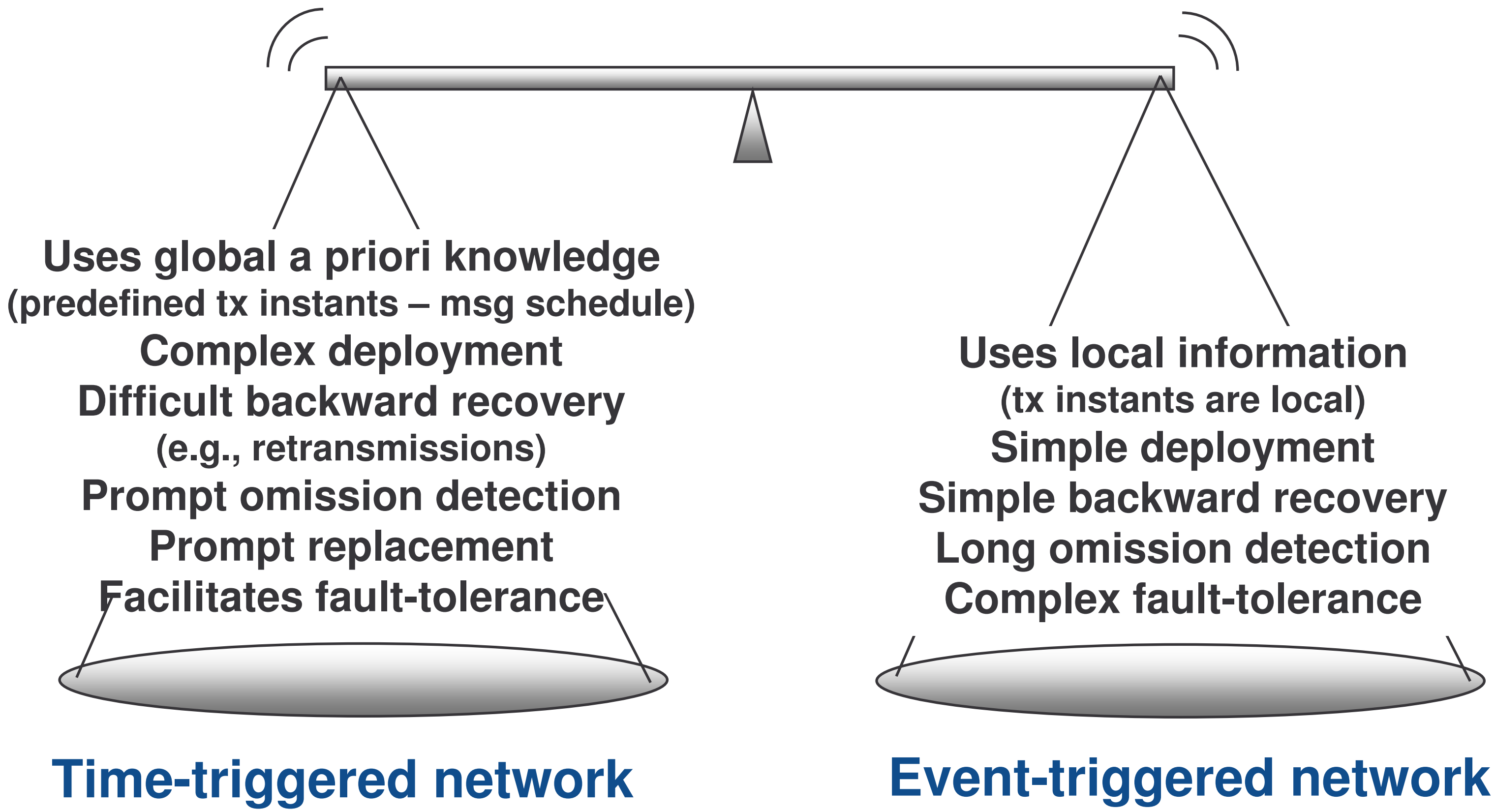




Notice:
$\Delta t = t_i - t_{i-1}$ set according to system dynamics

Losing one message causes inconsistency during Δt

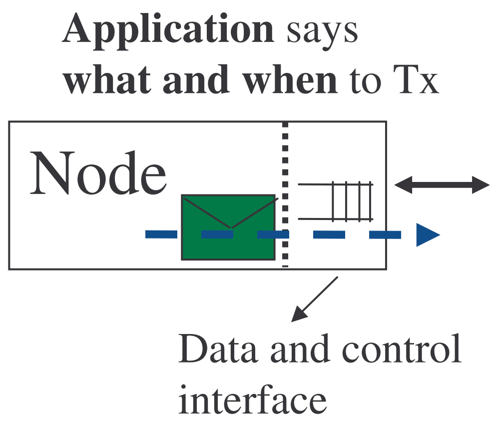
Event vs time triggering



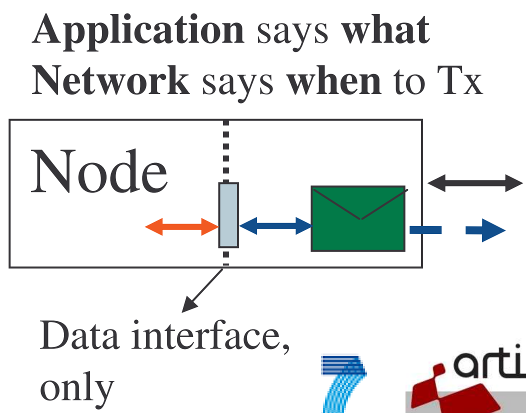
How to support event & state messages efficiently?

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.**

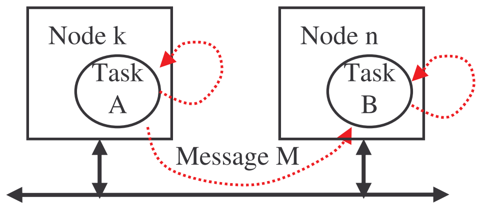


  - **Autonomous control**
    - The network **triggers** transactions **autonomously**.
    - No control signal crosses the CNI.
    - Applications **exchange data** with the network by means of **buffers**.
    - **Deterministic behavior.**

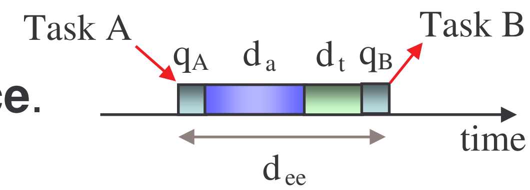


Information flow

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

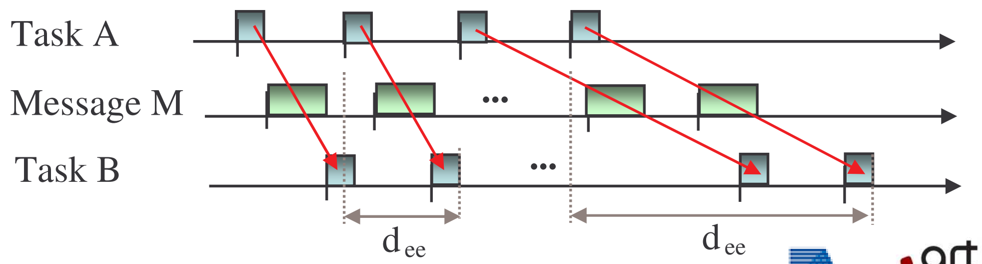


  - **ET-network with external control**
    - Transactions are composed of several **elementary actions** carried out **in sequence**.



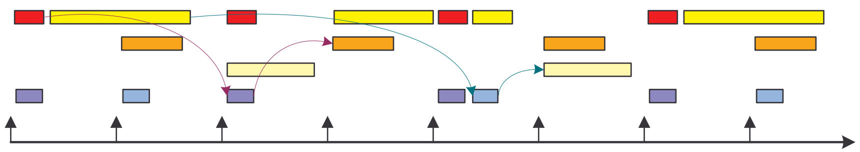
  - **TT-network with autonomous control**
    - The **elementary actions** in each intervenient (transmitter, network, receiver) are **decoupled**, spinning at an appropriate rate.

Requires relative phase control to avoid high delays and jitter

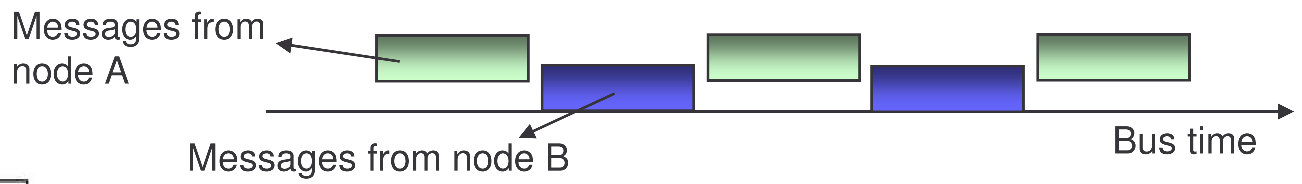


Information flow

- 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



An illustrative comparison

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

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

Inside the protocol stack

Physical entities & technologies

- **Computing - nodes**
  - Desktop and Laptop PCs
  - Embedded PCs
  - Single-board computers
  - Micro-controllers, specific processors, FPGAs, ...

DES

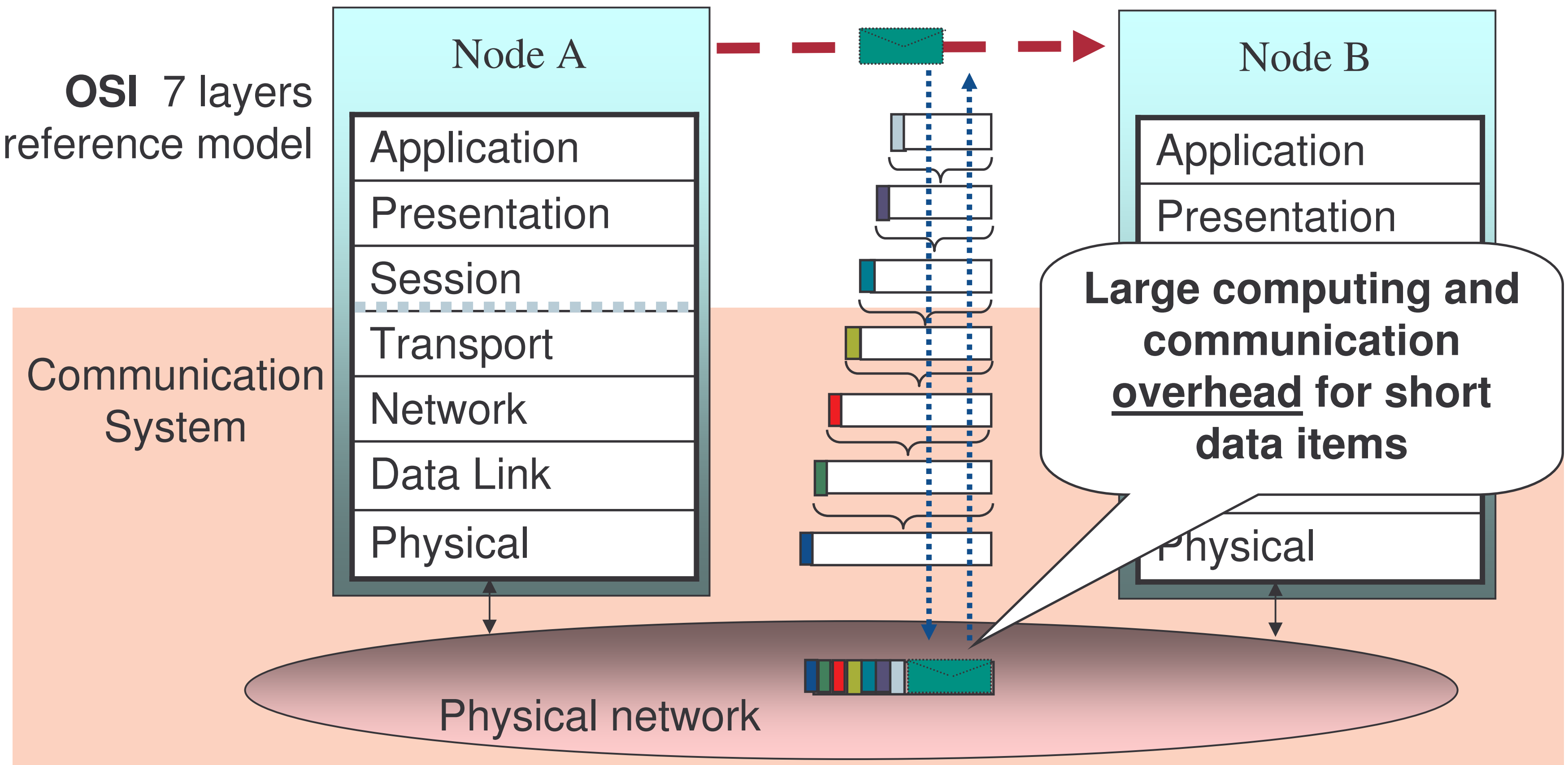
! Several of these technologies were developed for general purpose systems → potential impact on efficiency...

- **Communication - networks**
  - Public telephony networks (ADSL, GPRS, UMTS...)
  - Local Area Networks (Ethernet, WiFi, ...)
  - Control networks (CAN, FlexRay, PROFIBUS, ...)
  - Powerline networks (X10, HomePlug, ...)
  - Personal Area Networks (Bluetooth, IrDA, ...)
  - Sensor Networks (ZigBee, RFID, ...)

DES

U/NES

The *not only physical* communication process
(the OSI protocol stack)

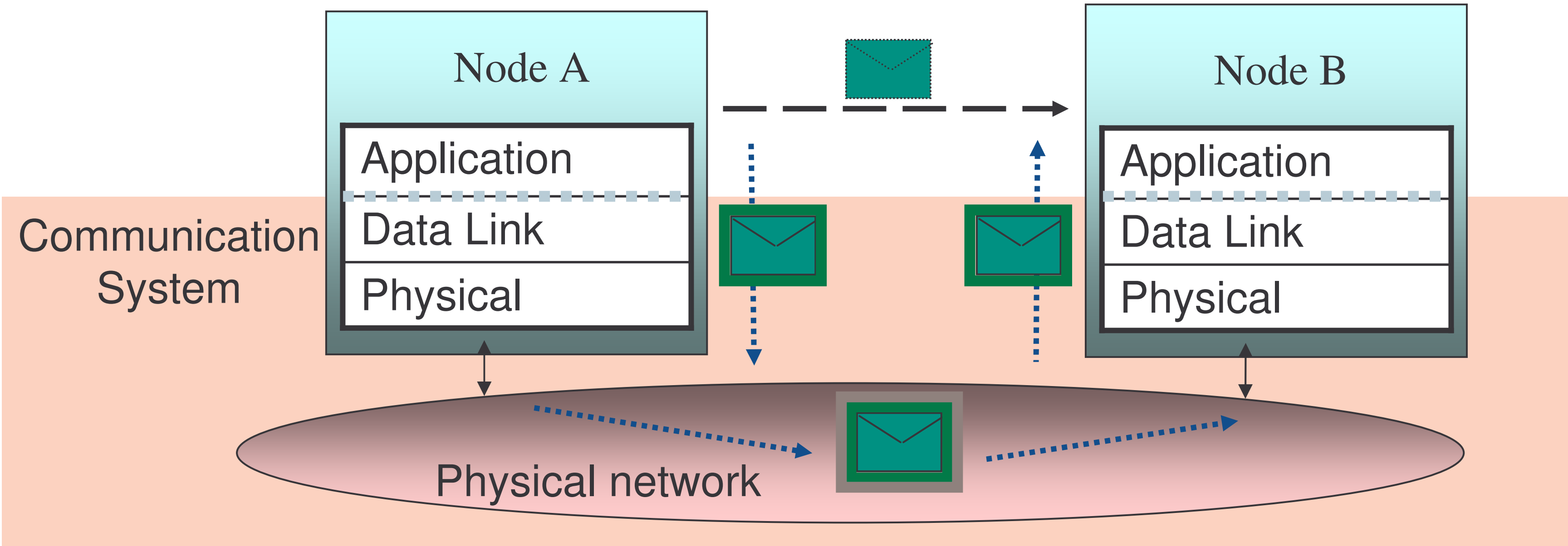


Requirements for an embedded / real-time protocol stack

- **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 overhead…**
  - **Time to execute** the protocol stack (can be dominant wrt tx time)
  - **Time to transmit** protocol control information (wrt to original data)
  - **Memory requirements** (for all intermediate protocol invocations)
- **Many embedded / 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)

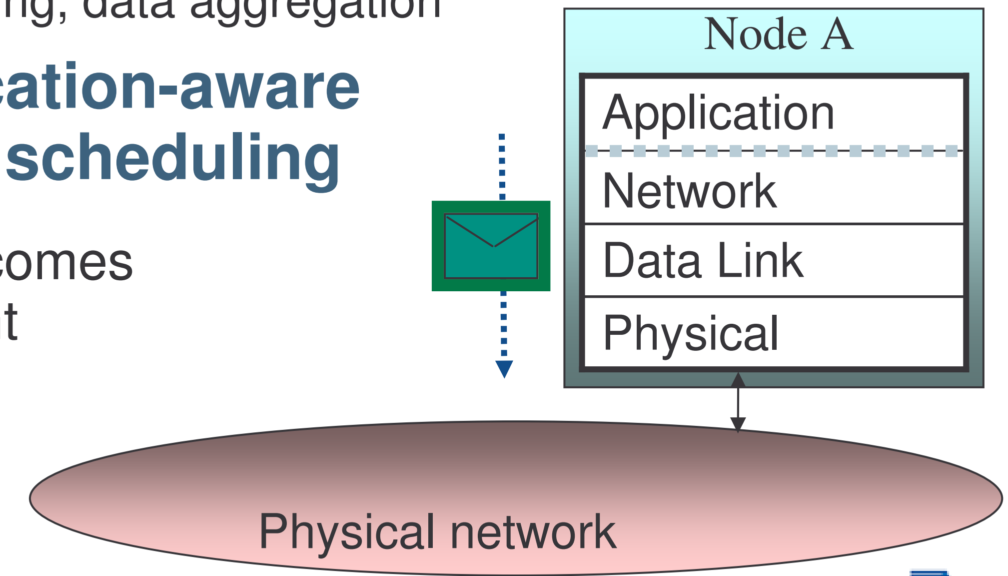
OSI collapsed model

- **Application services access the Data Link directly**
  - Services from other layers maybe present
- **In process control and factory automation these networks are called <u>Fieldbuses</u>**



Ubiquitous systems (e.g., WSN)

- **A network layer with routing services is fundamental**
  - multi-hop communication, logical addresses
- **Some more complex application layers**
  - Specific middlewares with certain services
    - Localization, tracking, data aggregation
- **Energy-aware / location-aware routing and traffic scheduling**
  - Synchronization becomes particularly important



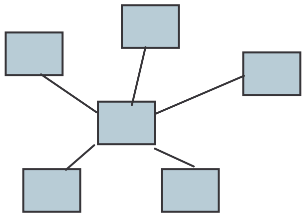
Issues in the Physical Layer

Physical layer

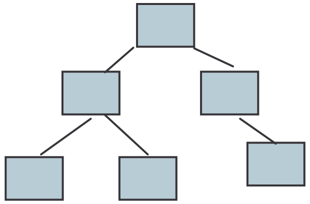
- **Issues related with the physical layer:**
  - **Interconnection topology**
  - **Physical medium**
  - Coding of digital information
  - **Transmission rate**
  - Maximum interconnection length
  - Max number of nodes
  - Feeding power through the network
  - **Energy Consumption**
  - Immunity to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

Physical layer

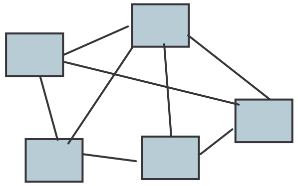
- **Interconnection topology**



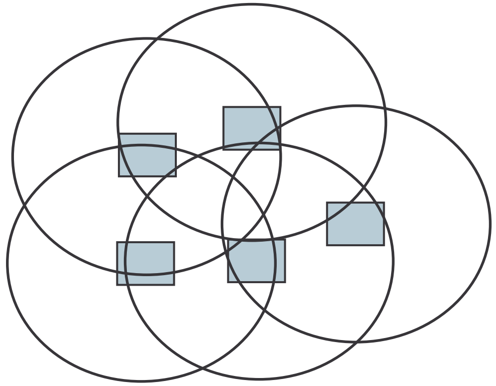
Star



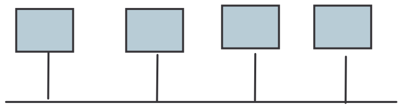
Tree



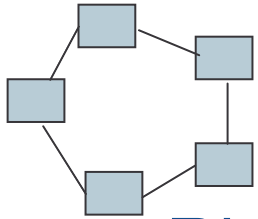
Mesh (wired)



Mesh (wireless-RF)



Bus

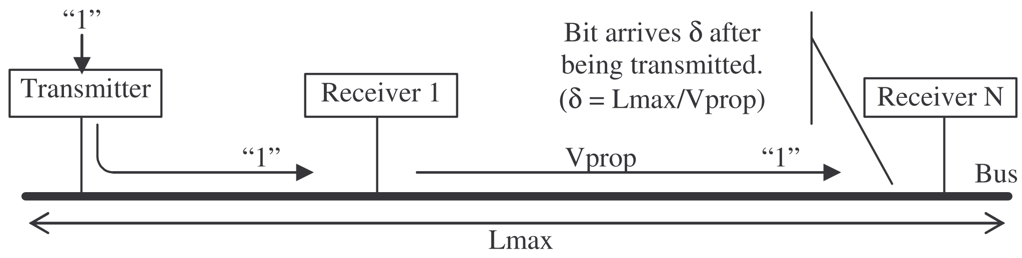


Ring

- **Physical medium**
  - **Copper wiring**
  - Optical fibers
  - **Wireless – Radio Frequency**
  - Wireless – Infra-red light

Physical layer

- **Propagation delay (δ = L/V)**
  - Caused by the **limited speed** of the electromagnetic wave
    - Typically, about **5ns/m in copper cables**, **3.3ns/m in the air**



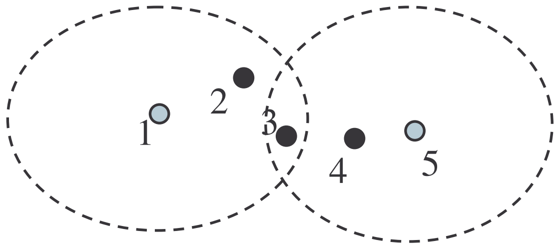
- **Bit length (b = δ*Tx_rate)**
  - Number of bits traveling on the medium at the same time
    - High bandwidth networks always have b>1
    - b=1 → all nodes "see" the same bit at a time (CAN)

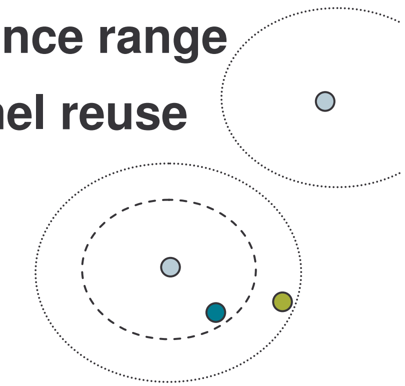
Physical layer

- Wireless propagation
  - Strong attenuation
    - **Free-space attenuation model**, increases with log of distance

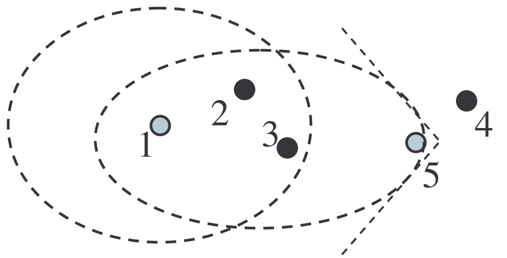
$$P(d) = P_0 - \alpha \lg d + X_\sigma$$

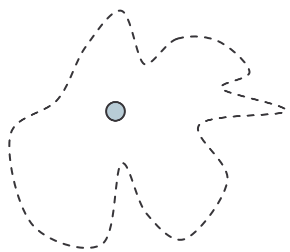
    - Establishes a **communication range** and an **interference range**
    - Cause of **hidden-nodes** but also allows **spatial channel reuse**





  - Multi-path fading, directional antennas, asymmetric links...





Physical layer

- **Energy consumption**
  - Very **complex relationship** among transmission power, bit rate, frame rate, network traffic, error control coding, retransmissions policy, type of medium access control and physical bit encoding. **Need to trade-off**.
- **Example of energy 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**

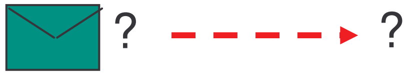
Issues in the Data Link Layer

Data link layer

- **Issues related with the data link layer:**
  - **Addressing**
  - **Logical link control – LLC**
    - Flow control
    - 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 (MAC address)

  – **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

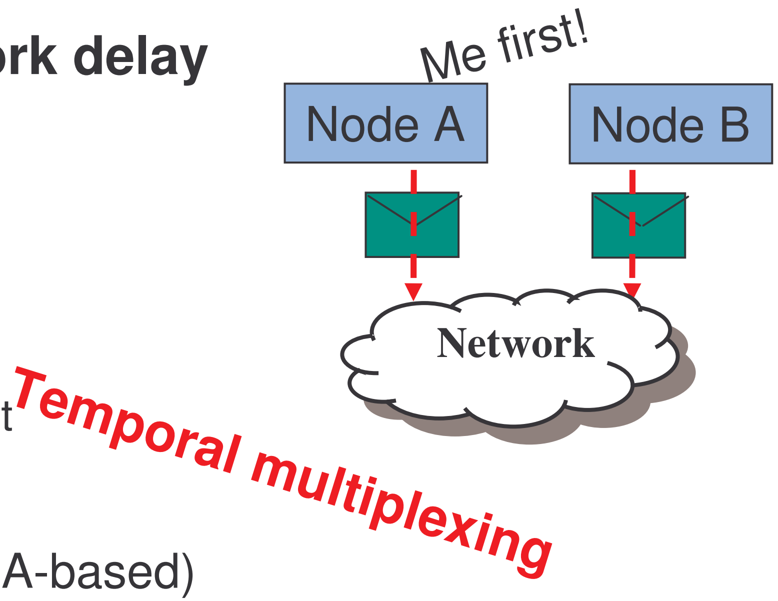
  – **Indirect (time-based) addressing**
    The message is **identified by the time** instant at which it is transmitted

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., ack. mechanisms, retry mechanisms, request data on reply
    - **Retries mechanisms** (e.g., PAR, ARQ) might have a strong **negative impact** on the data **timeliness**...
      - No point on continue trying to transmit out-dated values

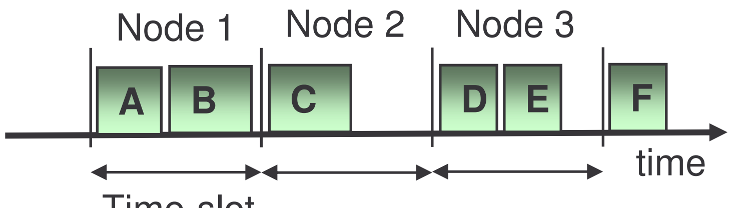
Data link layer

- **Medium Access Control**
  Sorts out the multiple access to a shared medium
  - 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 (typically CSMA-based)
      - The nodes try to transmit immediately when needed



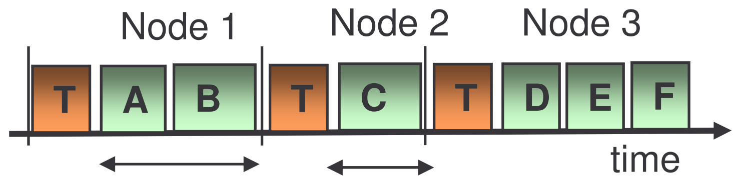
Data link layer

- ## Controlled access



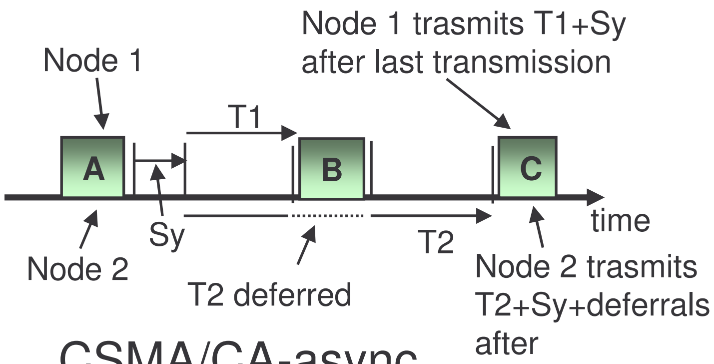
TDMA

Master-Slave

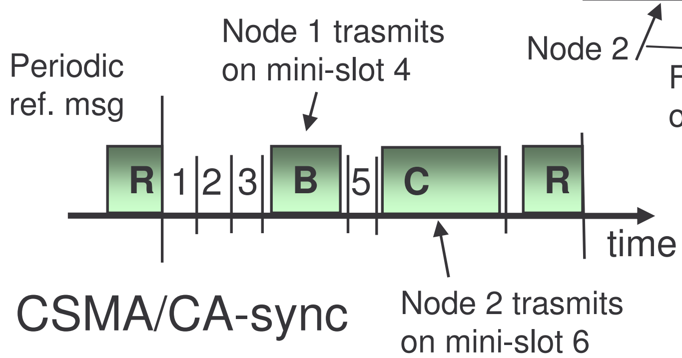


Token-passing

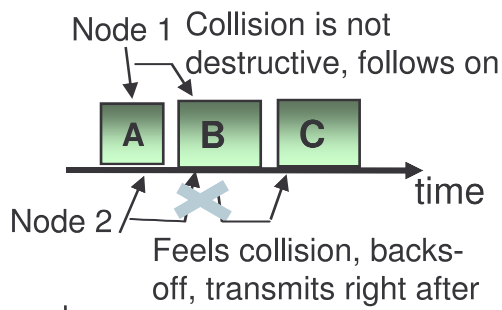
- ## Uncontrolled access



CSMA/CA-async
(p-persistent)



CSMA/CA-sync



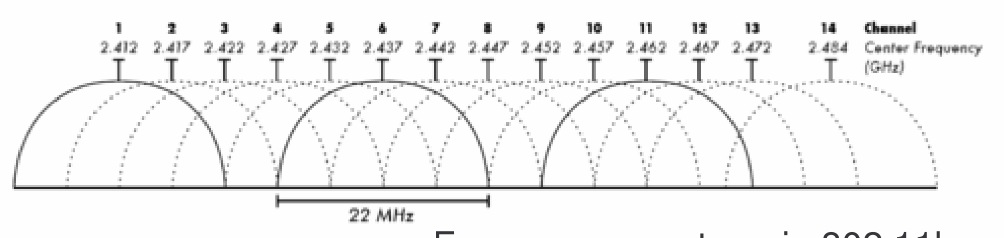
CSMA/BA
(1-persistent)

Data link layer

- **Master-Slave**
  - Inherent synchronization, any scheduling, single point of failure
- **Token-passing**
  - Inherent bandwidth reclaiming, distributed, token is s.p. of failure
- **TDMA**
  - Isolation among nodes, composability, requires synchronization
- **CSMA/CA-async**
  - Distributed, simple, not collision-free
- **CSMA/CA-sync (mini-slotting)**
  - Distributed, priority-based, requires synchronization
- **CSMA/BA**
  - Distributed, very efficient priority-based, requires dominance property

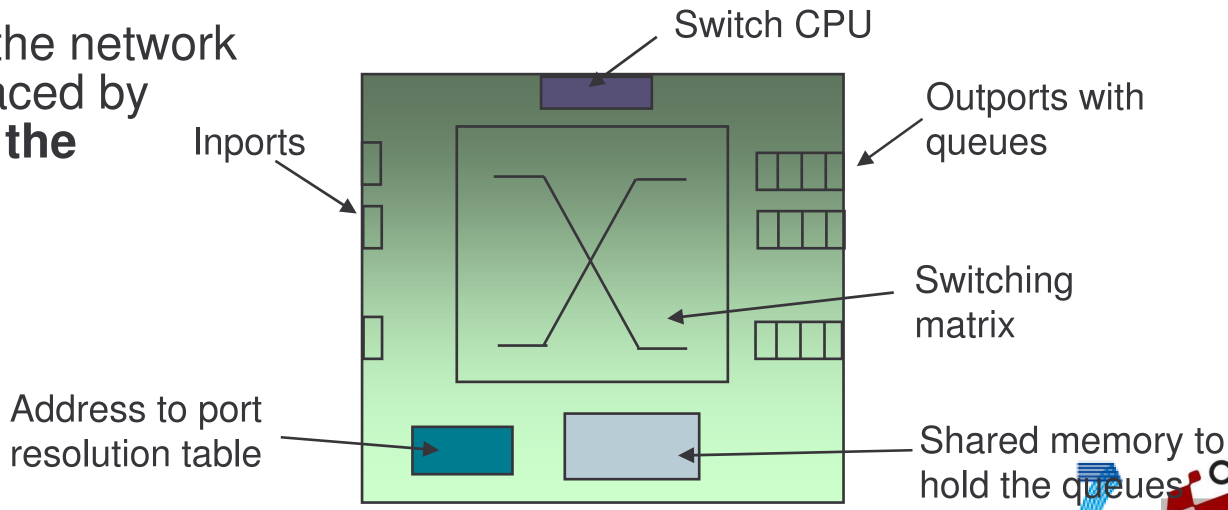
Data link layer

- **Frequency multiplexing**
  - Wireless channels
    - Assignment: dynamic (GSM, cognitive radio) versus static (WiFi, ZigBee)



Frequency spectrum in 802.11b

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



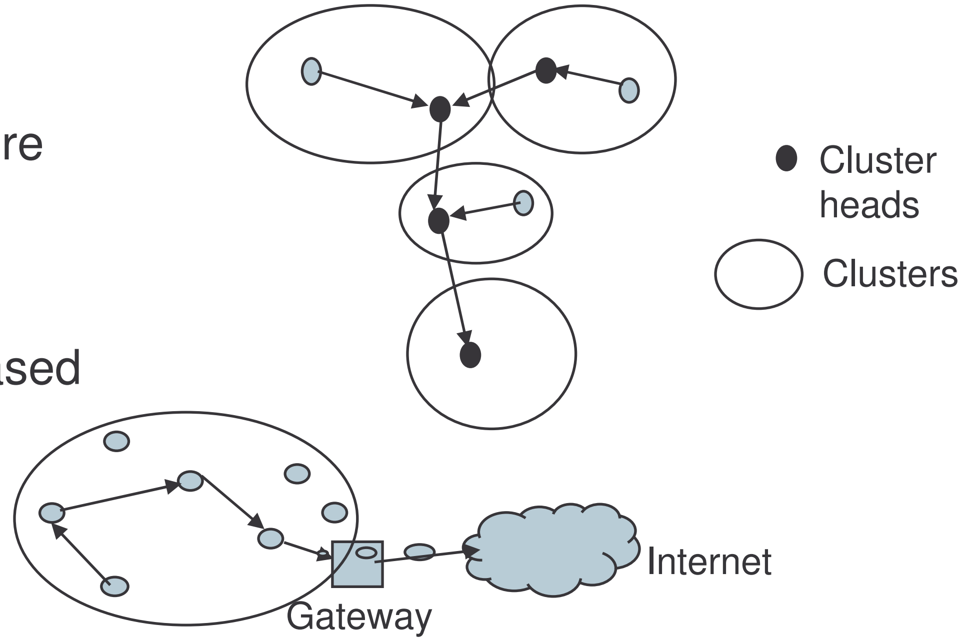
Issues in the Network Layer

Network layer

- **Issues related with the network layer:**
  - Logical addressing
  - Routing

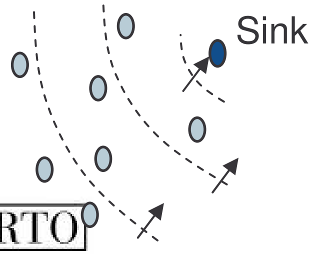
Network layer

- **Logical Addressing**
  - Independence of higher protocols wrt the physical hardware
  - Requires an ARP (address resolution protocol)
- **Routing**
  - **Inter-network routing**
    - Pre-planned infrastructure
  - **Ad-hoc routing**
    - Flat routing
    - Hierarchical / Cluster-based
  - **Setting routes**
    - Proactive
    - Reactive



Network layer

- **Channel reuse in space**
  - Nodes that are sufficiently apart can transmit simultaneously
    - communication range versus interfering range
- **Synchronized frameworks**
  - Communication occurs in slots that are defined across the network (TDMA fashion)
    - Require synchronization (clock, beacons)
  - Enables reducing the data routing delay by defining adequate transmission (slot) sequences →**slot sequence = data progression**
  - Enables controlling on-off switching for energy saving



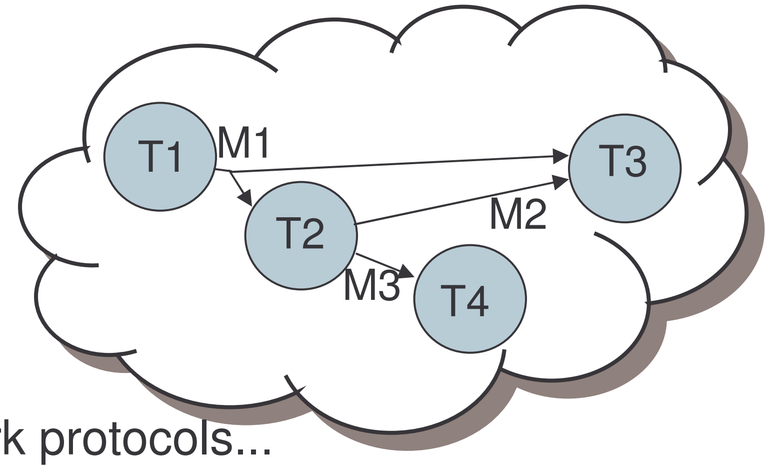
Issues in the Application Layer

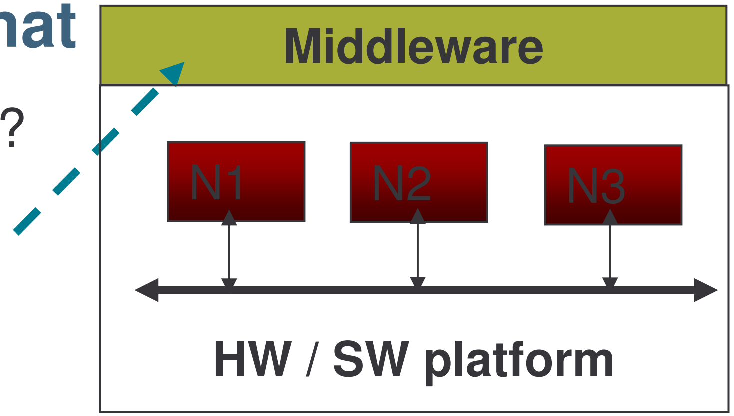
Application layer

- **Issues related with the application layer**
  - Set of services that are common for a class of applications
  - **Middleware issues**
    - Transparency wrt Distribution, OS, Languages...
    - Support for different programming paradigms (SO, OO, CB...)
    - **Cooperation model**
      - Client-Server
      - Producer-Consumer
      - Producer-Distributor-Consumer
      - Publisher-Subscriber

Application layer: Abstracting away platform details

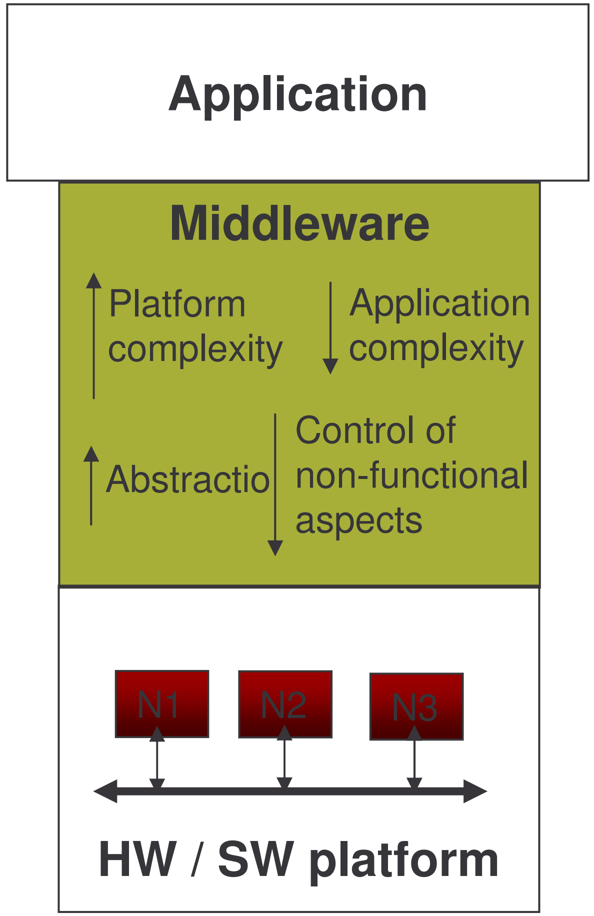
- **Applications are executed on HW / SW platforms**
  - Computing and communication
- **Implying many idiosyncrasies**
  - Dependence on HW / SW features
    - Processors, OSs, languages, network protocols...
- **How to develop applications that**
  - Are agnostic to such idiosyncrasies?
  - Still delivering their services and exhibiting the desired properties...
  - And executing on a distributed platform...





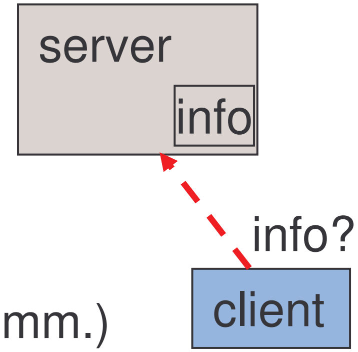
Application layer: Middleware

- **The middleware is a SW layer that**
  - Hides unnecessary details to the application (e.g., distribution)
  - Simplifies development, adds new services
- **But it implies trade-offs**
  - The HW and low-level SW have a profound impact on non-functional properties
    - timing, performance, dependability...
  - The simpler it is to develop applications the more complex the platform is
  - If the **right properties** are not properly enforced in the **lower layers** it will be hard (inefficient) to enforce them on the upper ones



Application layer: cooperation models

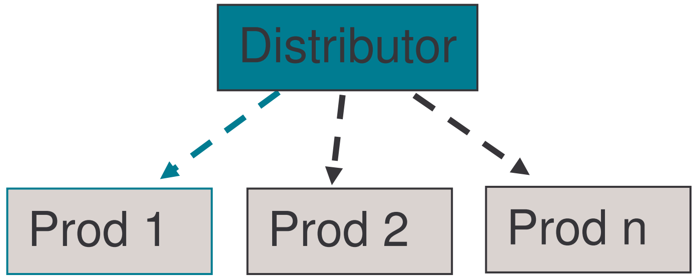
- ## Client-Server



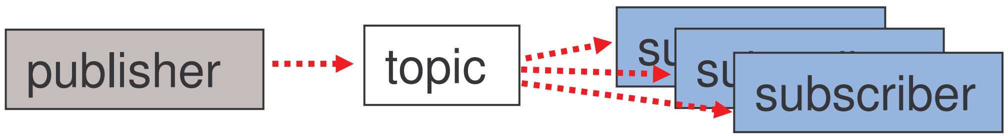
  - **Servers hold** information. **Clients request** it
    - Transactions **triggered by the receivers** (clients)
    - Typically based on **unicast** transmission (one to one comm.)
  - Transactions can be **synchronous** (client blocks until server answers) or **asynchronous** (client follows execution after issuing the request)

- ## Producer-Consumer
  - **Producers disseminate** information. **Consumers use** it
    - Transactions **triggered by the senders** (producers).
    - Based on **broadcast** transmission (each message is received by all)

Application layer: cooperation models



- **Producer-Distributor-Consumer**
  - Similar to Producer-Consumer
  - Adds a **central scheduler** to control Producers transmissions
    - Combination with Master-Slave access control
- **Publisher-Subscriber**
  - Uses concept of **group communication**
  - Nodes must adhere to groups either as **publisher** (produces information) or as **subscriber** (consumes information)
    - In some cases, multiple publishers for the same item are allowed
  - Transactions are **triggered by the publisher** of a group and disseminated among the **respective subscribers**, only (multicast)
    - Subscribers can in some cases define filters for the desired information



Communication technologies
for (Distributed / Networked) embedded systems

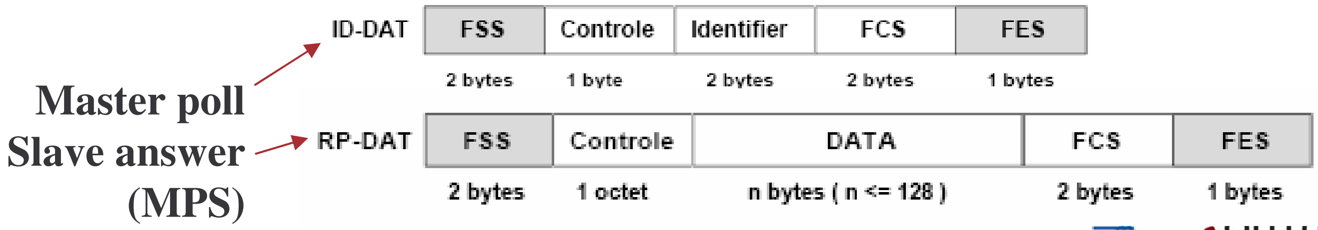
Communication protocols

WorldFIP

Factory Instrumentation Protocol

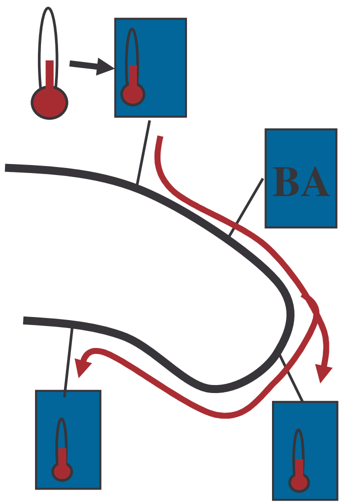
www.worldfip.org

- **Created, in the 80s, in France for use in** process control **and** factory automation**.**
- **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 NRT)**
- Source-addressing **(message identifiers with 16 bits)**
- Master-Slave **bus access control**
  - BA - Bus Arbitrator



WorldFIP

- MPS – ***Messagerie Periodique e Sporadique***
- Producer-Distributor-Consumer **middleware**
- **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**



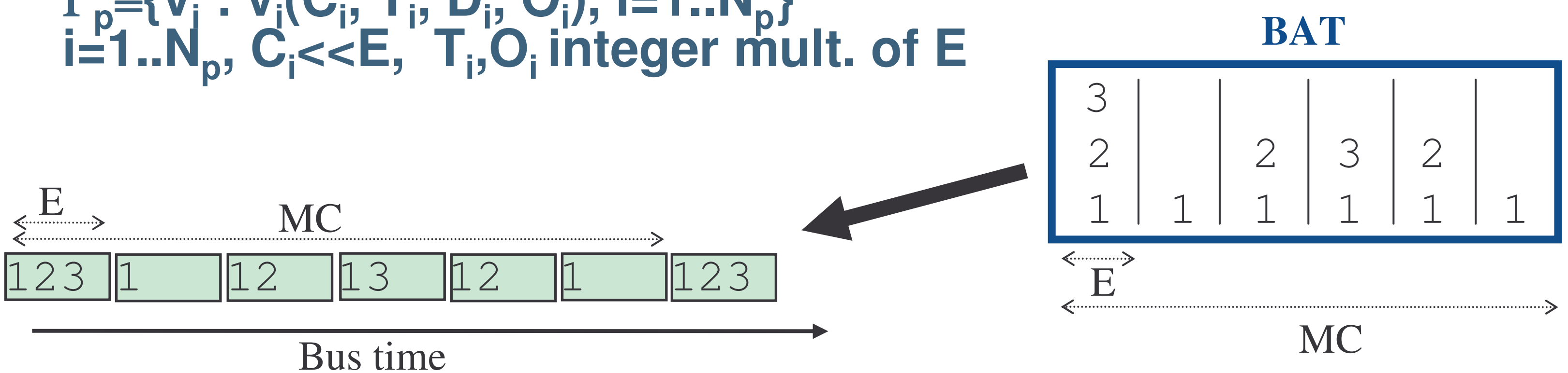
WorldFIP

- 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

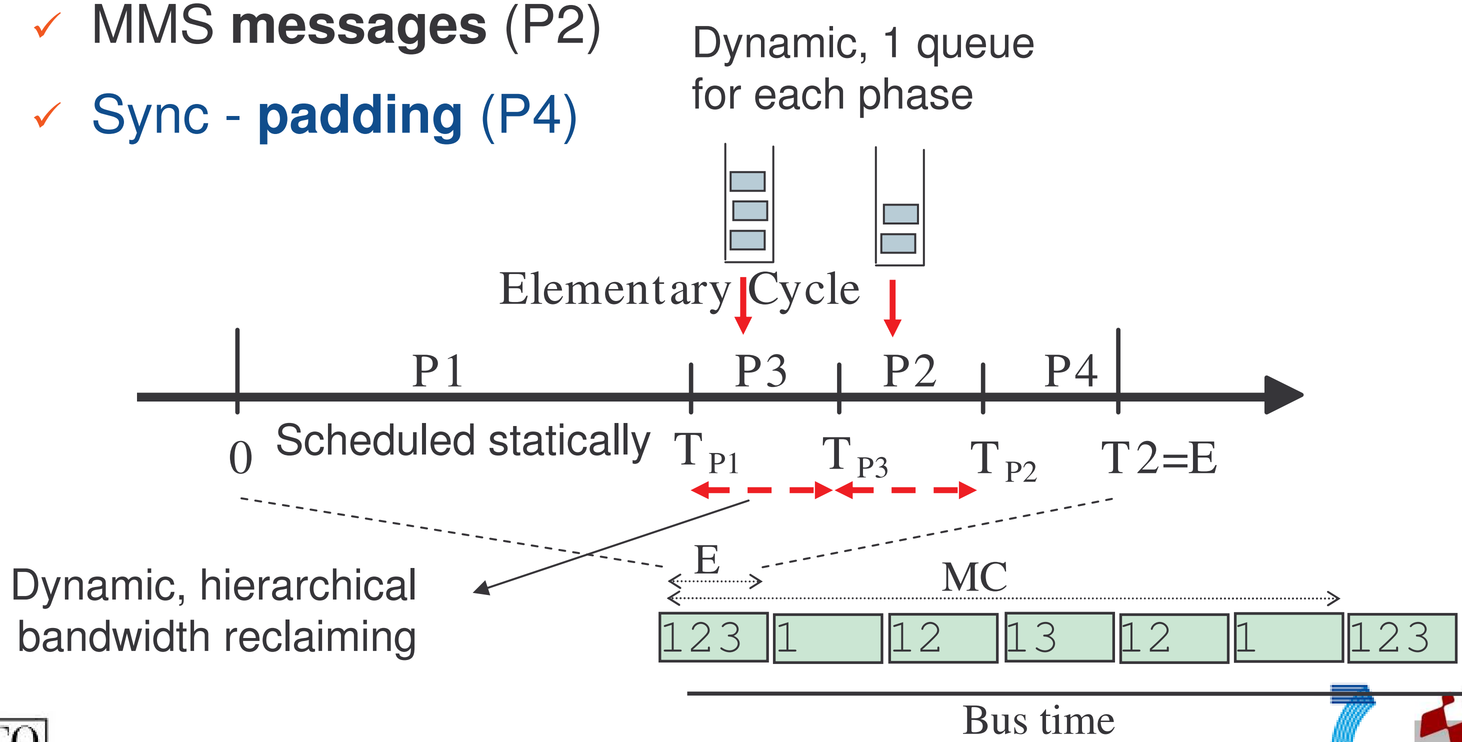
Periodic Variables:

v_i	1	2	3	
T_i	1	2	3	(ECs)



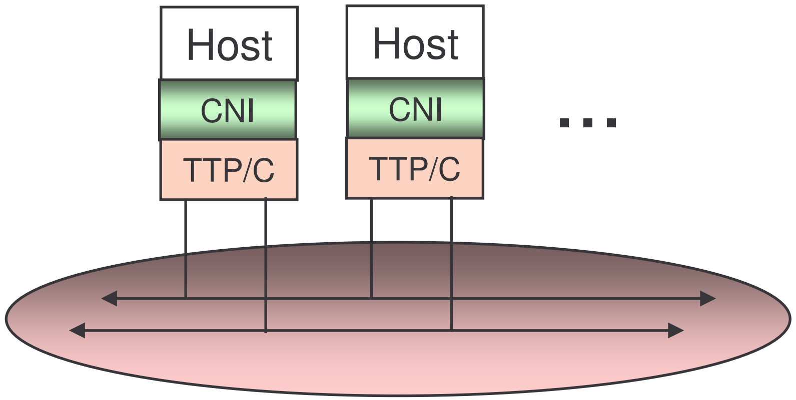
WorldFIP

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



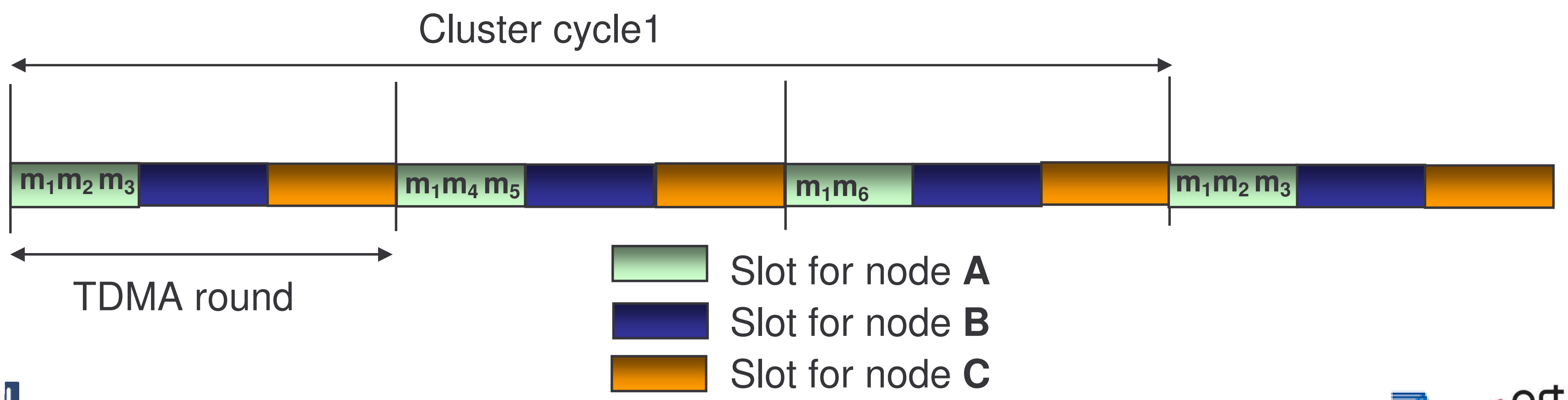
TTP/C
Time-Triggered Protocol
www.tttech.com

- 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

- **TDMA access** 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 240B)
  - **Each** frame **may contain** several messages
- The periodic sequence of messages is a **Cluster cycle**
- All **message transmission** instants are stored in a **distributed static table**, the MEDL

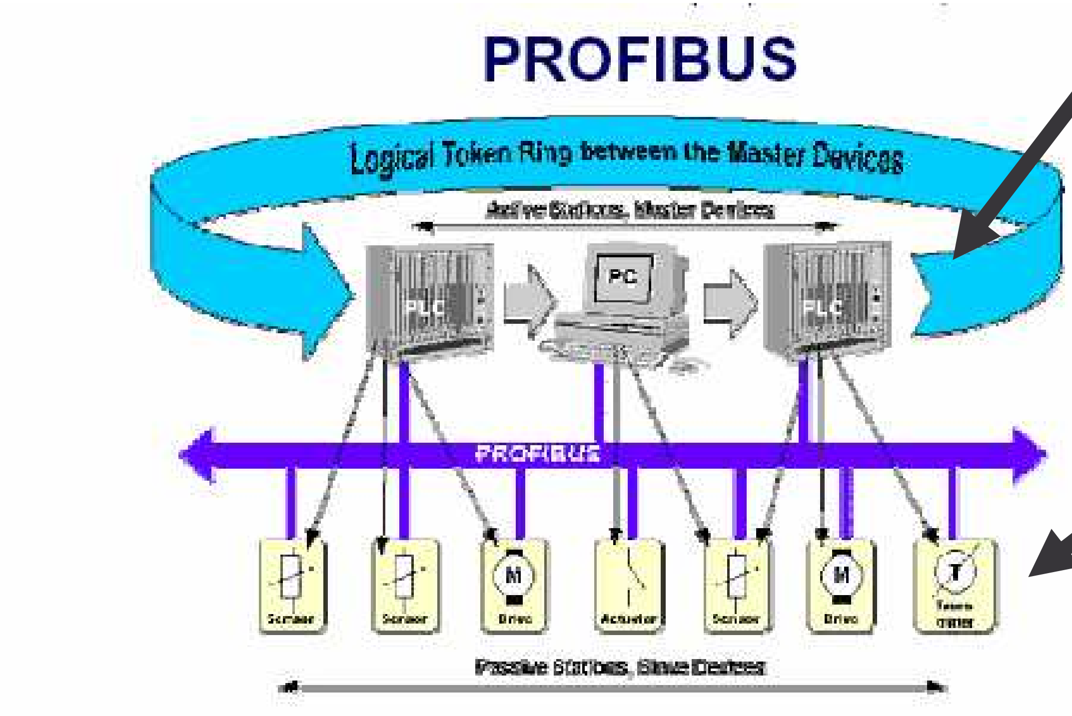


PROFIBUS
PROcess FieldBUS
www.profibus.com

- **Created in the late 80's by** Siemens, **in Germany**
- **Two main application profiles (client-server middlewares):**
  - **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

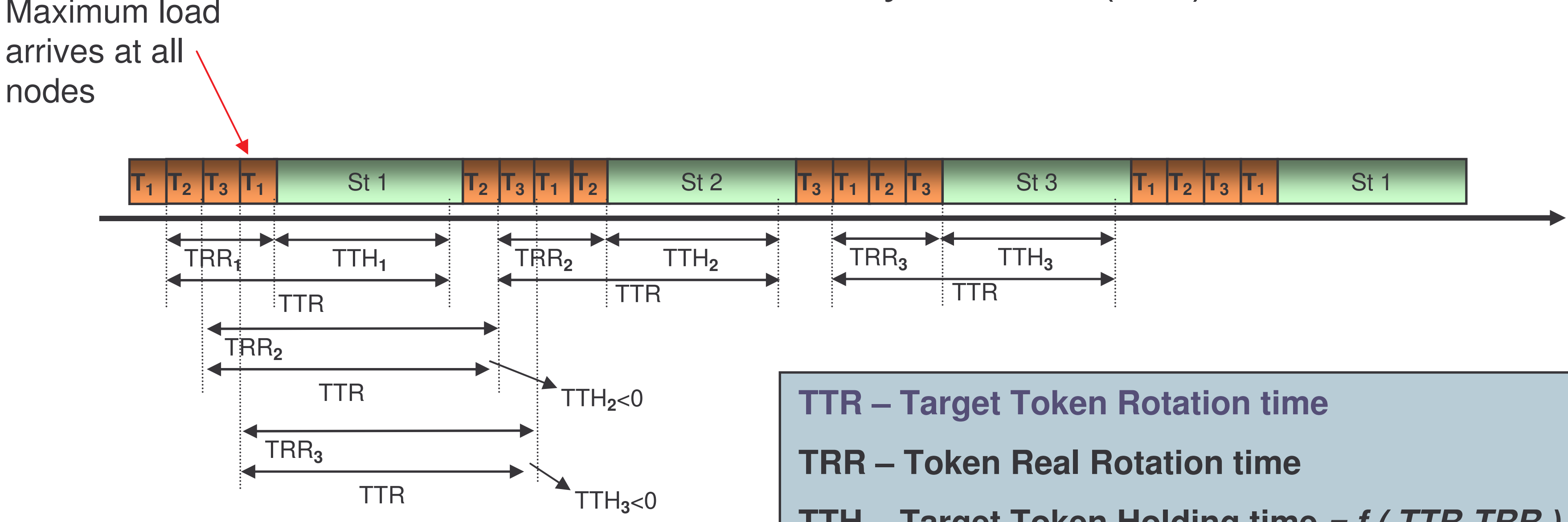


Masters:
controllers, regulators, … (PC, PLC,…)

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)



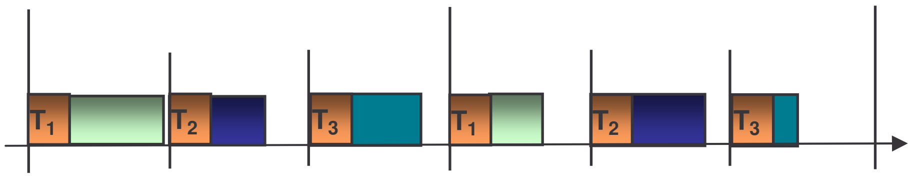
TTR – Target Token Rotation time

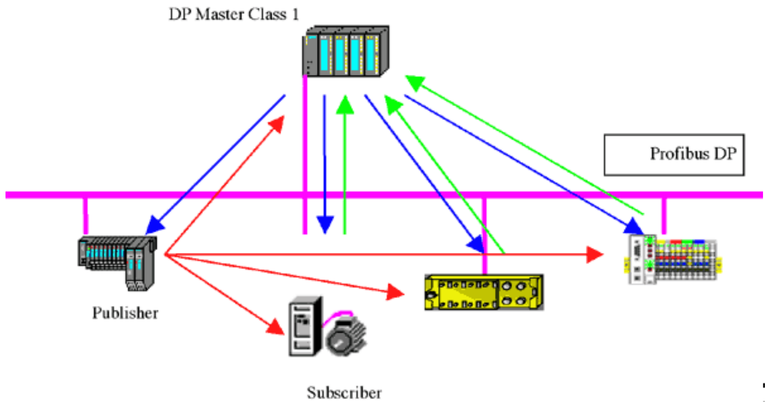
TRR – Token Real Rotation time

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

PROFIBUS

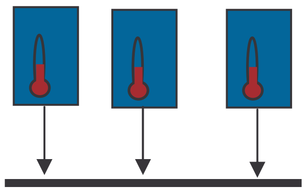
- **Recent support for: (PROFIBUS / DP-v2)**
  - **Isochronous** traffic
    - **Equidistant** token arrivals forming **cycles**
    - Cycle synchronization (specific global frame - broadcast)
    - Two windows in the cycle (isochronous and async traffic)
  - **Direct slave to slave** communication
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





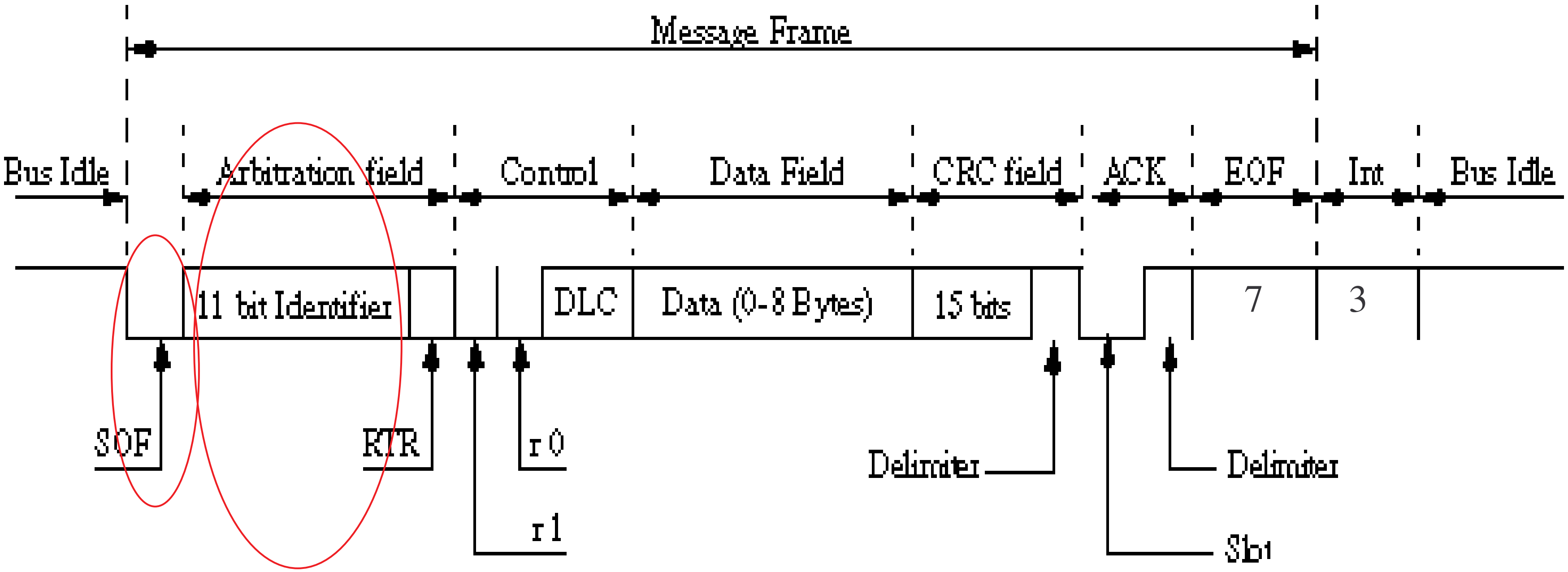
CAN
Controller Area Network
www.can-cia.de

- Created in the early 90s by Bosch, GmbH, for use in the automotive industry.
- 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** (CSMA type)
- **Non-destructive arbitration** based on message identifiers (establish priority)
  - Bit-wise deterministic collision resolution CSMA/BA (CA?,DCR?)



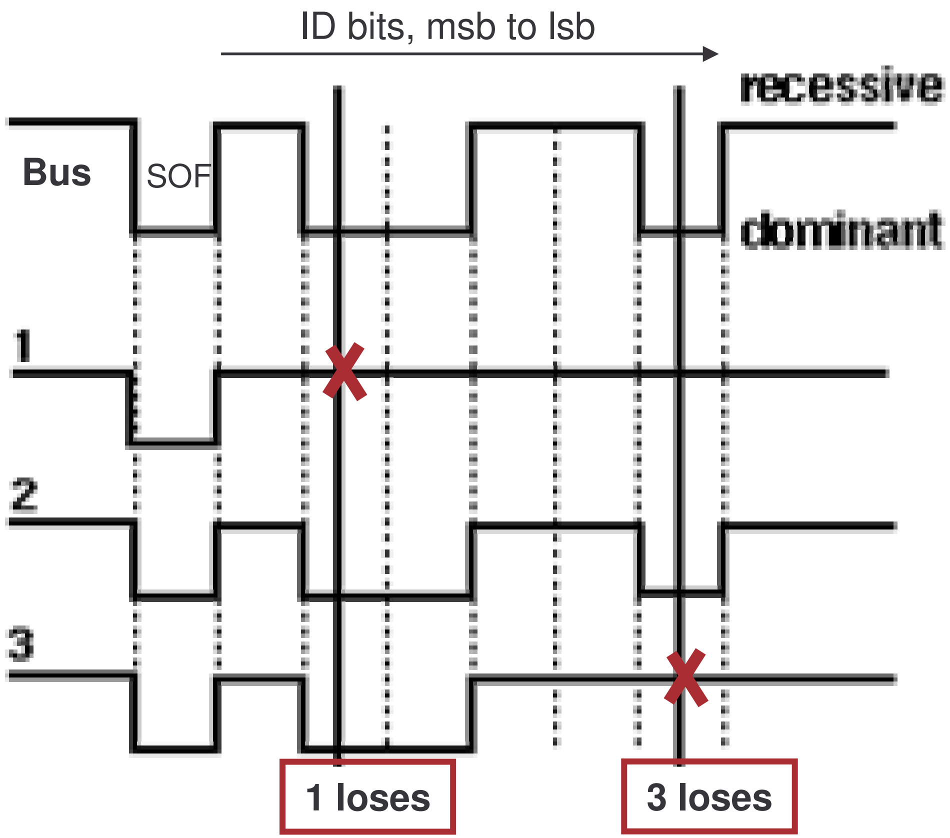
CAN

- ## CAN 2.0A message frame



CAN

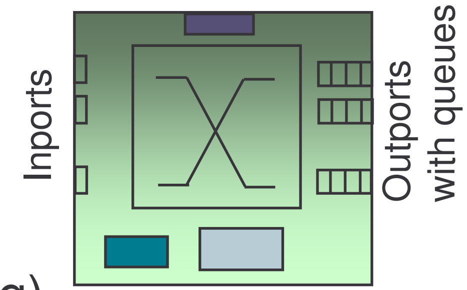
- **Prioritized arbitration**
  - All nodes transmit and listen every bit
  - If different, then lost arbitration and backoff



Ethernet
standards.ieee.org/getieee802/802.3.html

- Created in the mid 70s (!) at the Xerox Palo Alto Research Center.
- CSMA/CD non-deterministic arbitration **(outdated...)**
  - 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



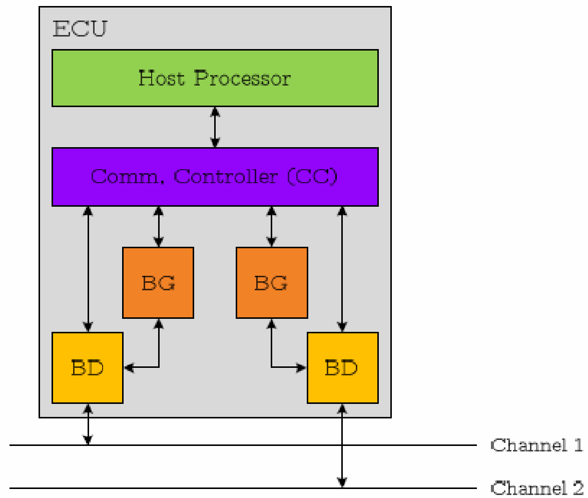
- **Using switches**
  - Became the **most common solution**
    - Current switches are **wire-speed** (non-blocking)
    - 802.1D – possibly with **multiple priority queues** (802.1p)
    - 802.1Q – **Virtual LANs**
  - Not perfect !
    - **Priority inversions** in queues (normally FIFO)
    - **Mutual 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

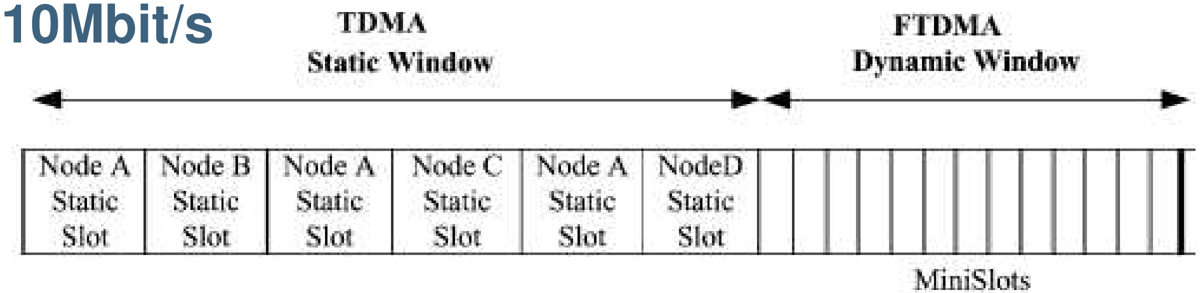
Ethernet

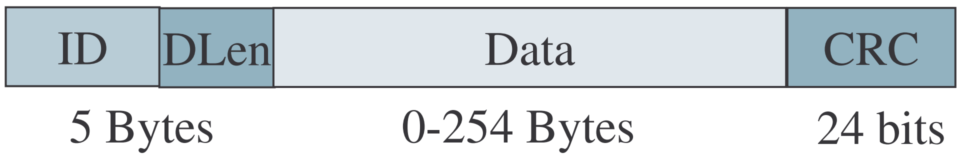
- Many industrial profiles / variants
  - Ethernet Powerlink
    - Master-Slave, similar to WorldFIP
  - EtherCAT
    - Open-ring with Master, specific 2-ports switch in each node, frames cross each node and are updated on the fly
  - PROFINET-IRT, TTEthernet
    - TDMA enforced by specific switch
  - Ethernet/IP, PROFINET-CBA, FF-HSE
    - Standard SE, with careful design of sources!
  - ...

FlexRay
www.flexray.com

- Created in the early 2000s by an industrial consortium from the **automotive domain.**
- **Aiming at** safety-critical **applications**
- Static **(TT - TDMA) and** dynamic **(ET - mini-slotting) isolated phases** (partitions)
- Two redundant channels **that can also be complementary**
- **Tx rate** up to **10Mbit/s**

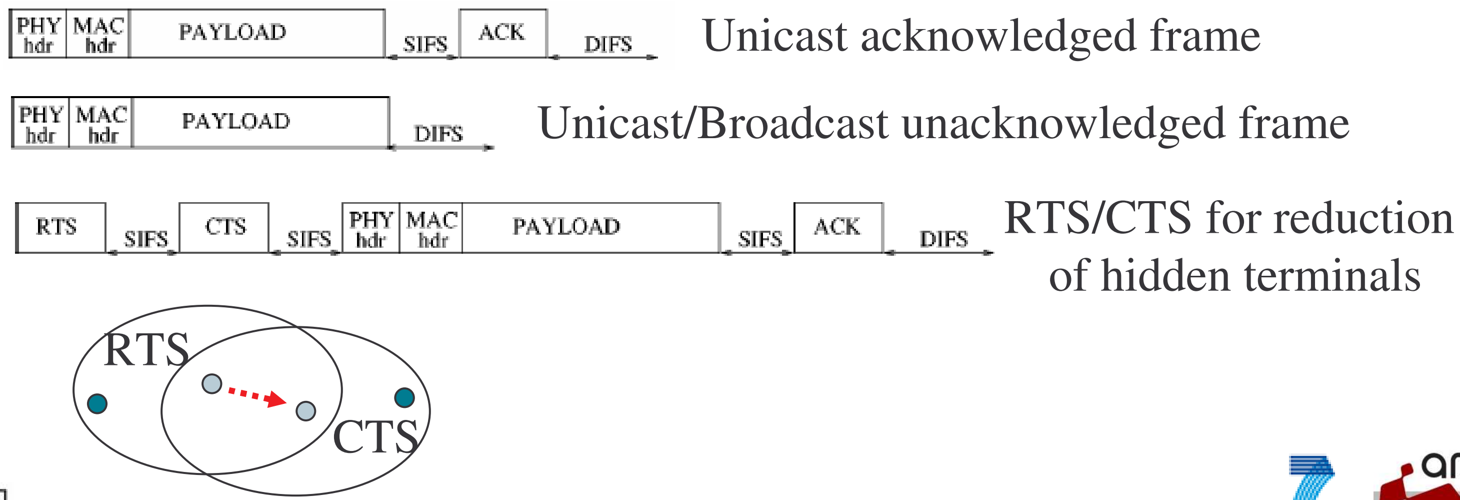






WiFi (802.11)
standards.ieee.org/getieee802/802.11.html

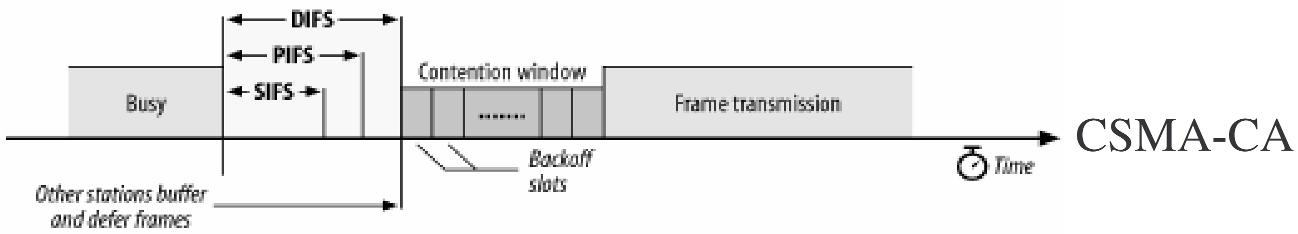
- Created in the 1990s for the SOHO domain and widely used for WLANs
- **3 modes/bands:** 802.11b/g (ISM-2.4GHz), 802.11a (5GHz)
- Scalable Tx rates between **1Mbit/s and 54Mbit/s** (or higher... 802.11n)
- Many mechanisms to **reduce collisions** and **hidden-terminals**
  - Retransmissions based on acknowledge



WiFi (802.11)

standards.ieee.org/getieee802/802.11.html

- Original MAC used different Inter-Frame Spaces to separate between:
  - Protocol packets (ACK, RTS, CTS, ...)
  - Contention-free access with master(AP)-slave (PCF) – *unused*
  - Contention access with CSMA-CA (DCF) – *standard*

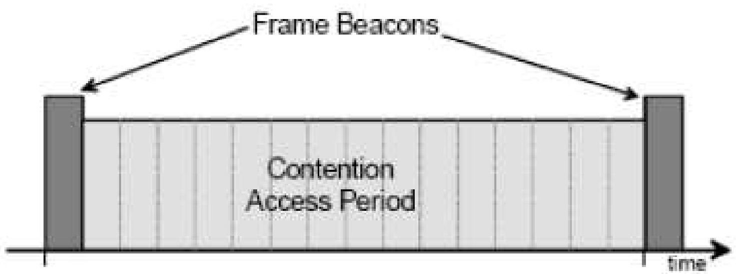


- Growing interest for **QoS support** (802.11e)
  - Separate Video and VoIP, from e-mail and general Internet access
  - EDCA (DCF with 8 different classes) + HCCA (AP schedules traffic)

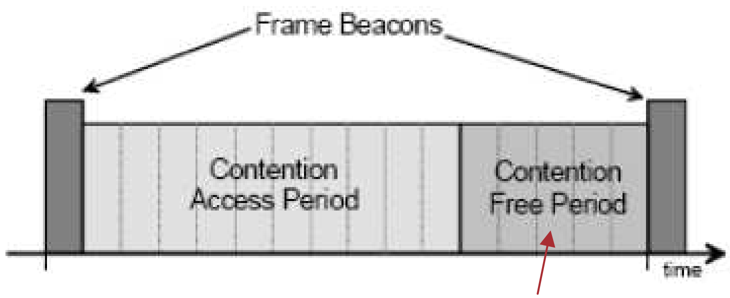
ZigBee

www.ZigBee.org

- Created in the early 2000s for **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 PAN coordinator to synchronize nodes, structure cells and QoS support. Also peer-to-peer mode. Routing support.
- Up to 65K nodes (full and reduced function devices)



Beacon mode with simple CSMA-CA



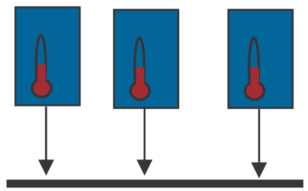
Superframe with GTS (Guaranteed Time Slot) scheduled in the beacon

# Communication technologies
for (Distributed / Networked) embedded systems

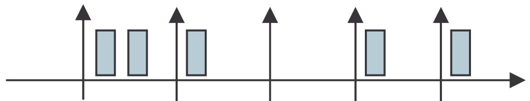
Standard middlewares

CANopen – a middleware for CAN
CAN-in-Automation
http://www.can-cia.com

- ## Key features:
  - **Process data** shared with the **producer/consumer** model
  - Standardized **device description** and **methods**
    - data, parameters, functions, programs



  - Standardized services for **device monitoring**
    - e.g. membership functions based on heartbeats
  - **System services**: synchronization message, central time-stamp message (e.g. synchronous data acquisition)



Sync messages

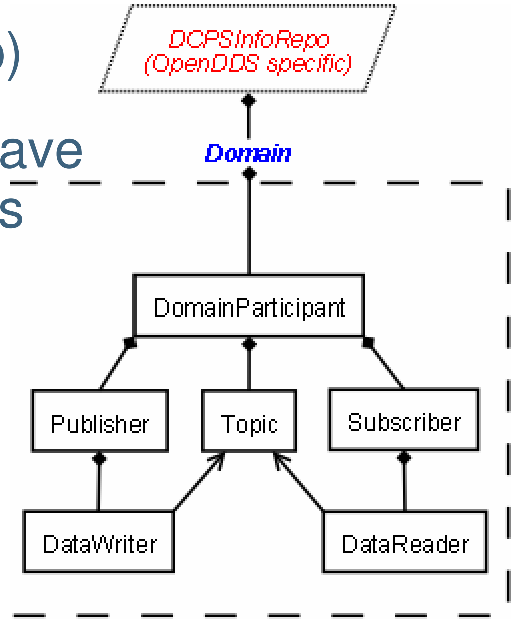
  - **Emergency messages**
  - Adopted by other protocols (e.g. Ethernet POWERLINK)

CANopen

- **Process Data Objects (PDO)**
  - Carry actual **application data**; broadcast, **producer/consumer** cooperation model, unacknowledged
    - Asynchronous PDOs (event-triggered)
    - Synchronous PDOs (time-triggered based on Sync Object)
- **Service data objects (SDO) -** device configuration
  - Read/write device OD entries, following sync client/server model
- **Network management (NMT)**
  - Node monitoring
  - Control their communication state

DDS - Data Delivery Service
portals.omg.org/dds

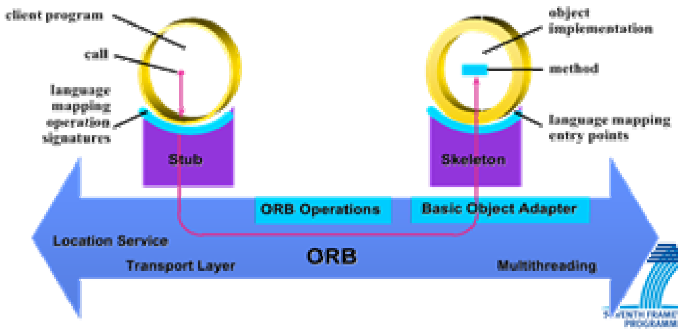
- **Real-Time Publisher-Subscriber** middleware for distributed real-time and embedded systems
- Standardized by **OMG** (Object Management Group)
- **DDS database shared among all nodes**, which have an holistic view on the communication requirements
- Publishers create "**topics**" (e.g. temperature) and publish "**samples**"
- Addressing, delivery, flow control, handled by DDS



CORBA - Common Object Request Broker Architecture

www.corba.org

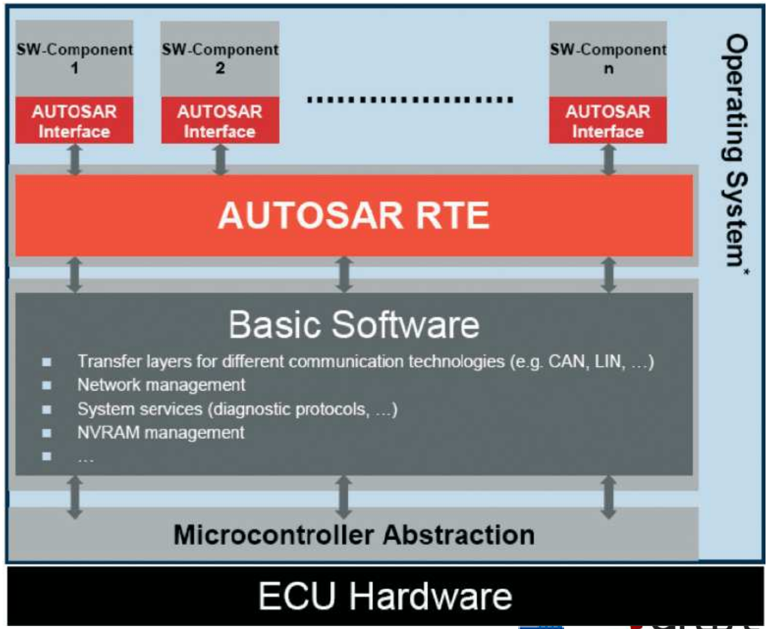
- **Open specification proposed by the OMG**
  - **Purpose:** Clients use remote objects as if they were local
- **Main features**
  - **Interoperability** between languages and platforms
    - Windows, Linux, Unix, MacOS, **QNX, VxWorks**, …
    - Ethernet, CAN, Internet, ...
    - Ada, C, C++, Java, Python, …
  - Multiple vendors (some are freeware products)
  - Many profiles
    - **Minimum CORBA, RT-CORBA**, FT-CORBA,



AUTOSAR

www.autosar.org

- **Proposed by a consortium of automotive industries**
- **Aims at separating functionality from execution HW**
  - Soft AUTOSAR components
- **Reduce number of active components and costs**
- **Improve efficiency in using system resources**
- **Manage complexity**
- **Give more design freedom to the OEM wrt subsystem providers**
- **Similar trends in avionics (IMA) and industrial automation (IEC 61499)**



Implementing the protocol stack

(by Paulo Pedreiras)

Protocol stack internals

- **Real-time protocol stacks may be based on:**
  - ✓ **Specialized communication** controllers and/or **custom device-drivers/stacks**
    - ✓ Low latency, high predictability, fine-grain control in queues, **but ...**
    - ✓ **Expensive**: non-COTS hardware, manpower for specific device-drivers development, longer development time, harder to debug, ...
  - ✓ **COTS** hardware and software
    - ✓ Cheap hardware, device drivers readily available for all the HW, full IP stacks supported, **but ...**
    - ✓ Potentially high **latency**, **unpredictability**, high **resource consumption** (memory and CPU), ...

Protocol stack internals

- **Using a TCP/IP stack:**
  - ✓ Easy **connection** to the **intra/Internet**, use of standard tools/apps, **easy development** (app. code independent from HW), but ...
  - ✓ "**standard**" TCP/IP stack:
    - ✓ **High CPU** and **memory** consumption (code and data memory in the order of hundreds of KB, multiple data copies); Not suitable to resource constrained embedded systems
  - ✓ "**lightweight**" TCP/IP stacks (e.g. lwIP, uIP):
    - ✓ **Code** size is around **10KB** and **RAM** size can be around **100's of B** (**suitable for 8/16 bit micro-controllers**), efficient buffer management (zero copy) , …
    - ✓ Some **limitations** (e.g. single interface, single connection)

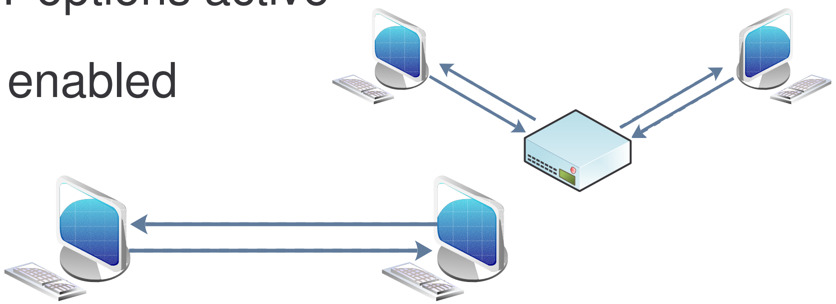
Protocol stack internals

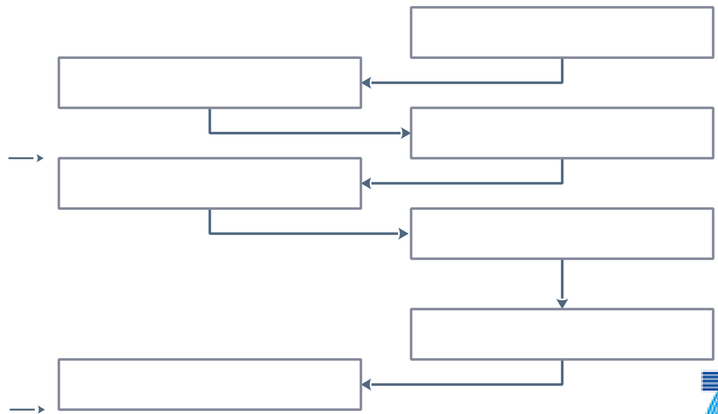
- **Issues with general-purpose Device-drivers**
  - ✓ DD **development** is **hard** and **costly**
    - ✓ Many real-time systems use general-purpose DD (GPDD), possibly with some adaptations
  - ✓ **Issues** with GPDD
    - ✓ optimized for throughput (**high latency/poor determinism**)
    - ✓ **FIFO queues** between the host memory and the NIC internal memory
    - ✓ DMA/IRQ **optimizations conflict** with **predictability**
      - Several messages can be buffered and generate a single INT/DMA transfer; Some operations have no defined time-bound
    - ✓ E.g. RTnet (Network stack for Xenomai and RTAI) DD for 3COM NICS (rt_3c59x.c) is stated as "**non real-time safe**"

Protocol stack temporal behavior

- **Round-trip measurements with Ethernet**
  (by Pedro Silva)

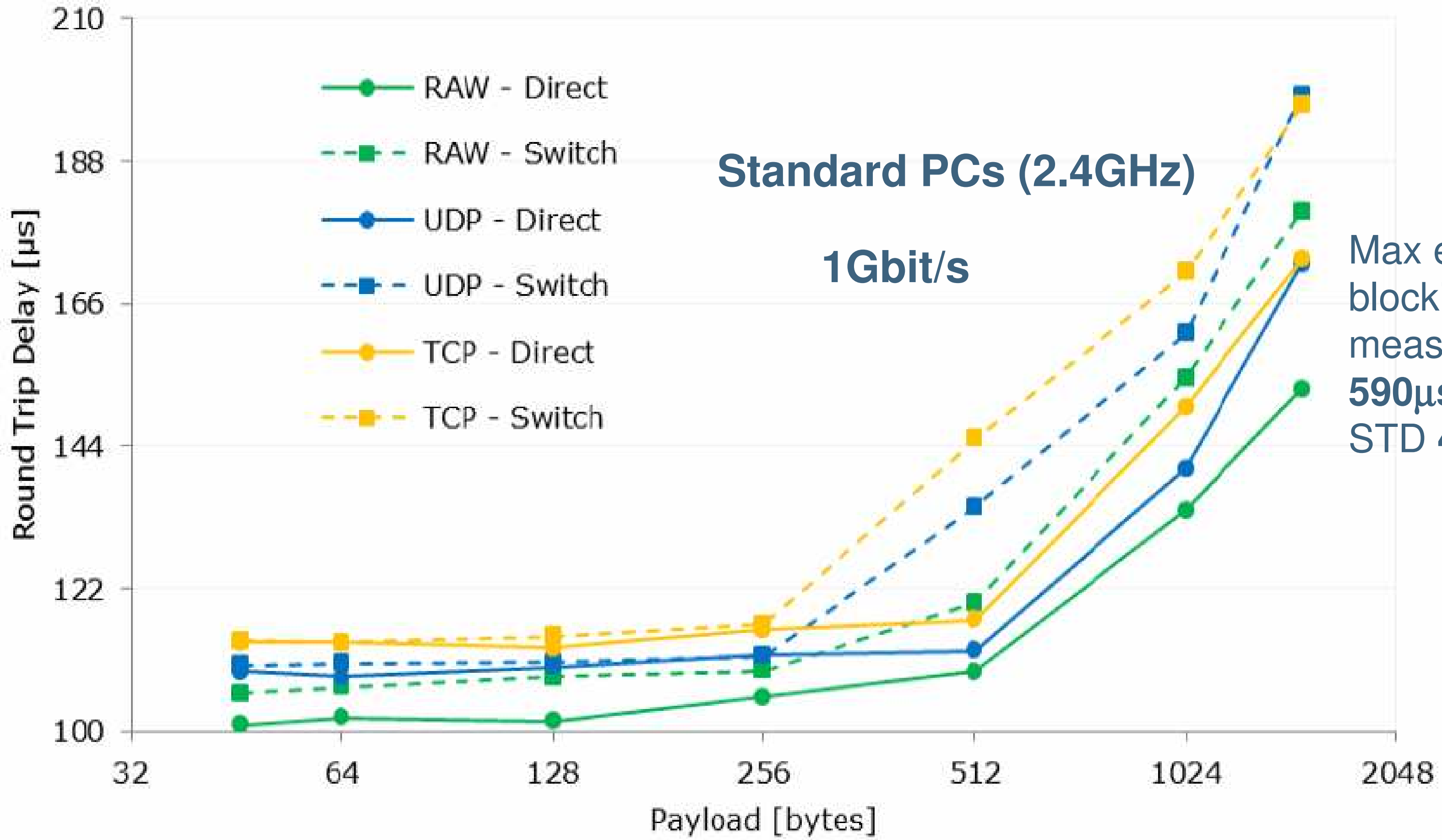
  - ✓ Standard Linux distribution (Ubuntu)
    - ✓ Plain (no RT features) and with RT options active
    - ✓ High-resolution timer, preemption enabled
  - ✓ An embedded Linux
    - ✓ IC NOVA AP7000 – with AVR32
  - ✓ Using sockets
    - ✓ RAW, UDP, TCP





Protocol stack temporal behavior

- Standard Linux (no RT) - average



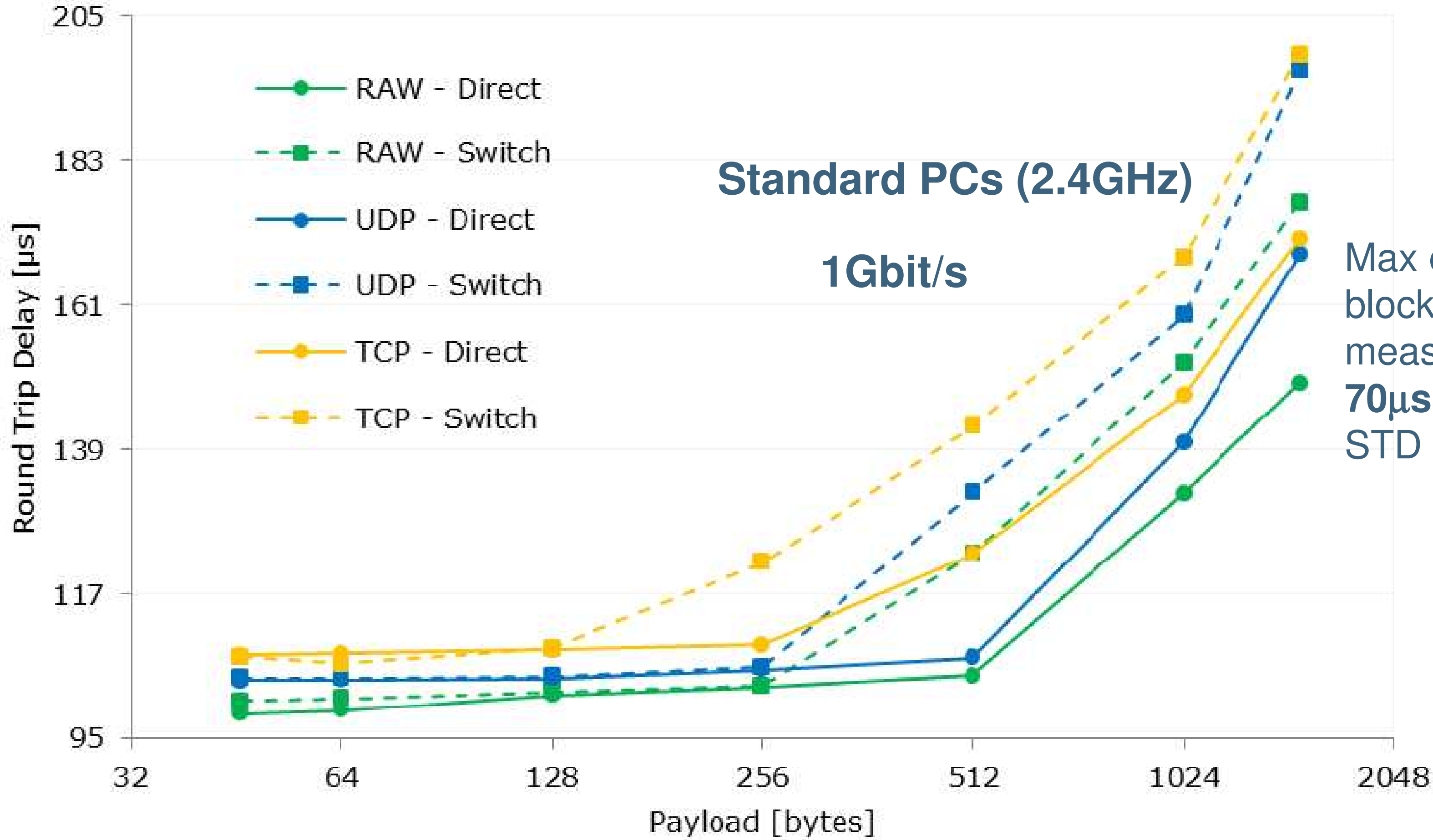
Standard PCs (2.4GHz)

1Gbit/s

Max extra blocking measured: **590µs**
STD **4-6µs**

Protocol stack temporal behavior

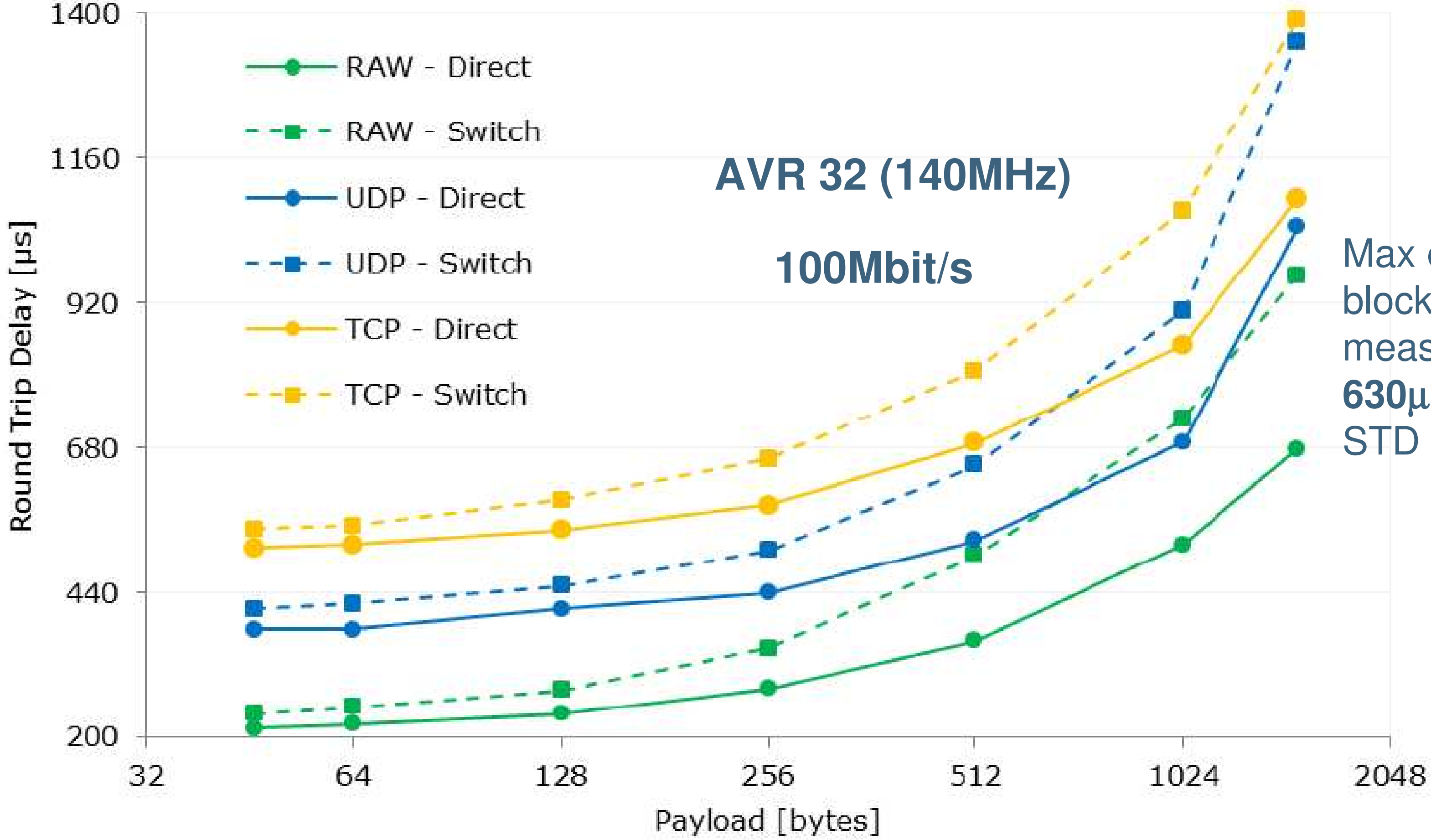
- ## Standard Linux (RT) - average



Max extra blocking measured: **70µs**
STD **2-4µs**

Protocol stack temporal behavior

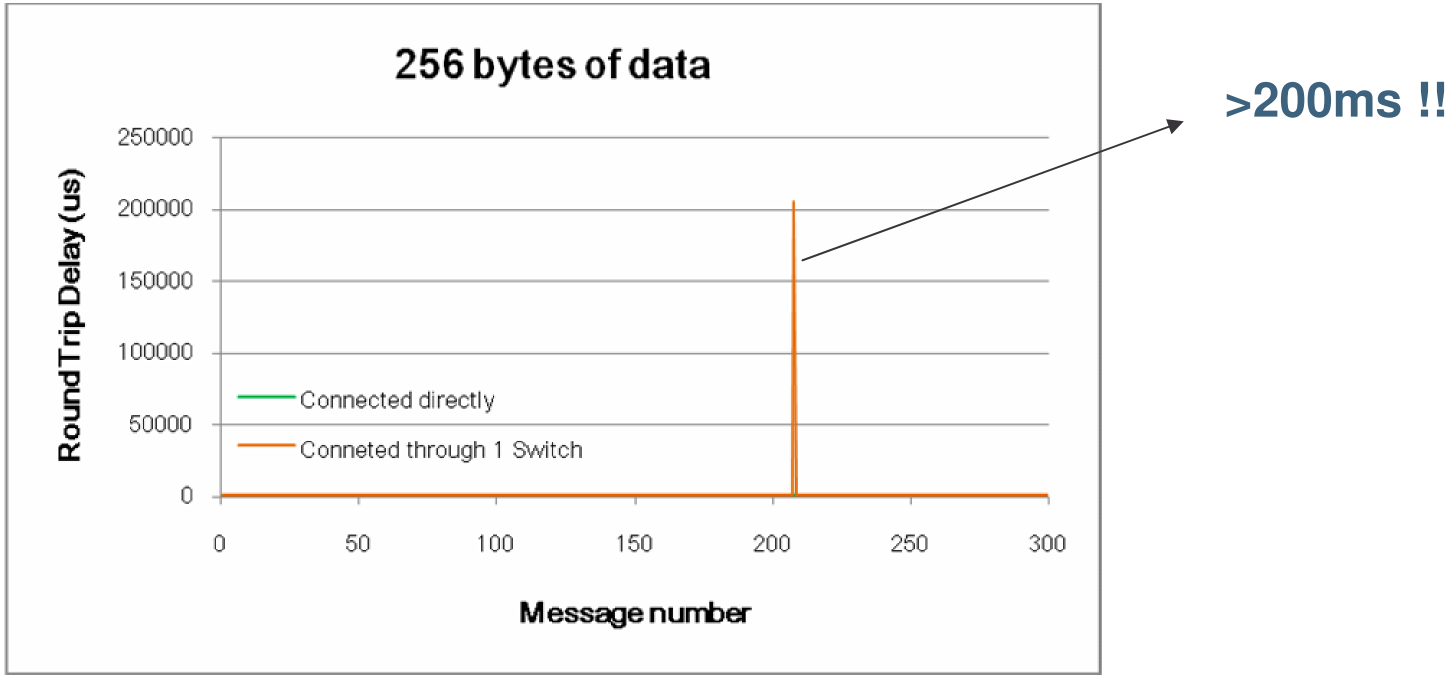
- Embedded Linux - average



Max extra blocking measured: **630µs**
STD **8-10µs**

Protocol stack temporal behavior

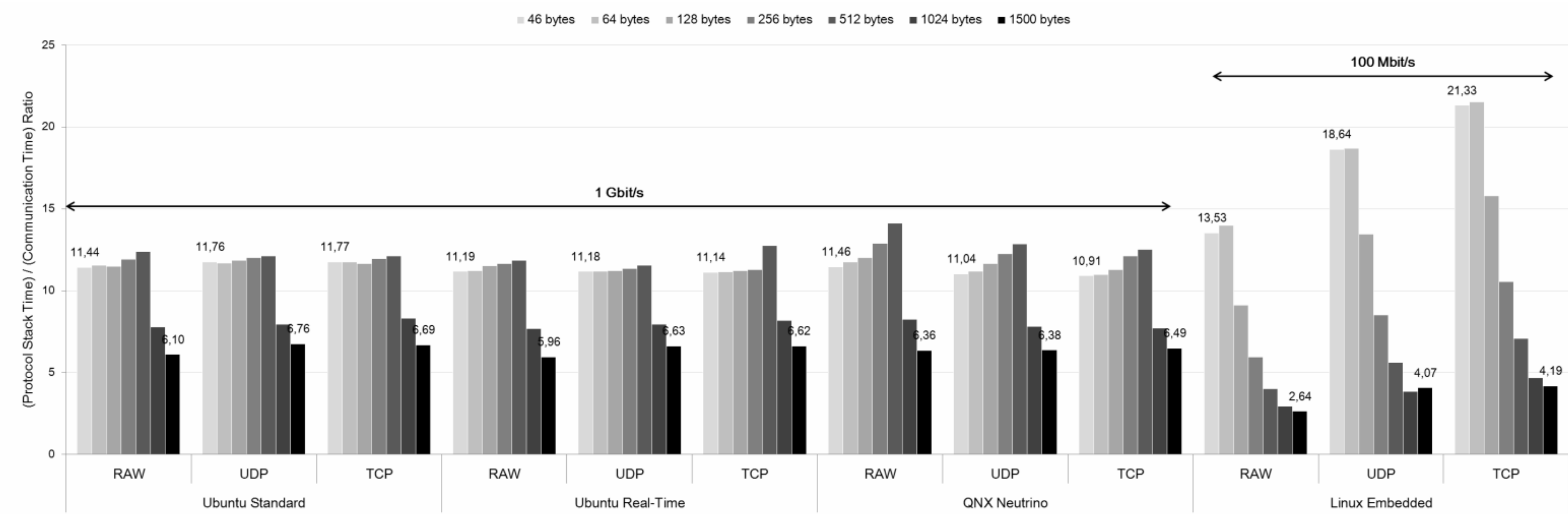
- **Problem of TCP...**
  - Long, poorly bounded, retries...



Embedded Linux

Protocol stack temporal behavior

- **Overhead of the network protocol stack**
  for different OS/platforms



$$\frac{\text{Protocol Stack Time}}{\text{Communication Time}}$$

for each OS, type of socket used and payload length

Protocol stack temporal behavior

- **Round-trip measurements with Ethernet**
  - ✓ If the platform is powerful, it is not worth using RAW sockets
    - ✓ Better exploiting the higher abstraction of IP communication
  - ✓ For low computing power platforms the difference might be significant.
  - ✓ TCP connections may incur very long delays caused by the error recovery mechanisms

Some open issues

Some open issues

- **In wired networks (DES or U/NES)**
  - Combining RT and nonRT nodes in a bandwidth efficient manner
  - Improving composability and robustness
    - Explore the use of stars to increase the robustness wrt errors and temporal misbehaviors and enforce partitions (virtual channels)
  - How to guarantee non-functional properties with HW-agnostic software components?
  - …

Some open issues

- **In wireless networks (U/NES)**
  - Improve synchronization for better RT behavior
    - For channel spatial reuse, systolic data routing
  - MACs & routing to further improve energy savings
  - Virtual channels in large scale networks
    - Adaptive resource reservation
  - Combination of reactive and proactive routing
    - To get the best of both worlds, low latency + adaptation
  - Resilience to interference
    - Improving frequency multiplexing
    - Exploiting new techniques, e.g., cognitive radio…
  - Improve bandwidth efficiency with new coding techniques
    - E.g., Network coding
  - …

Conclusion of the first day

- The **network is a fundamental component** within a distributed or networked system (supports **system integration**)
- **Real-time coordination** in a distributed / networked system requires **time-bounded communication**
  - appropriate protocols must be used
- We have seen a brief overview of the **techniques and technologies** used in the **networks and middlewares for embedded systems**
- Still many **open issues** remain in trying to **improve the timeliness**, **robustness** and **efficiency** of the communication
- **Tomorrow:**
  - **Traffic schedulability analysis**
  - **Case studies of (flexible) real-time communication**

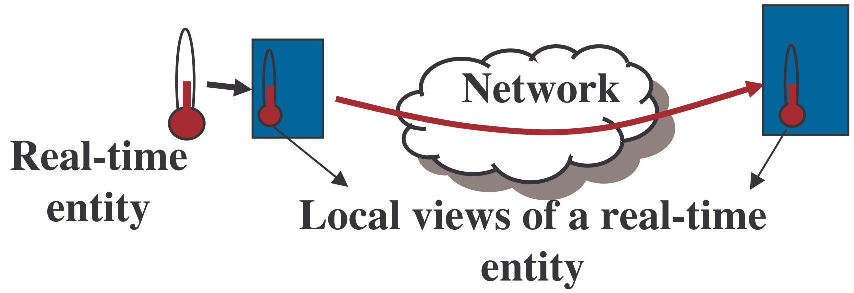
Traffic models
for scheduling issues

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

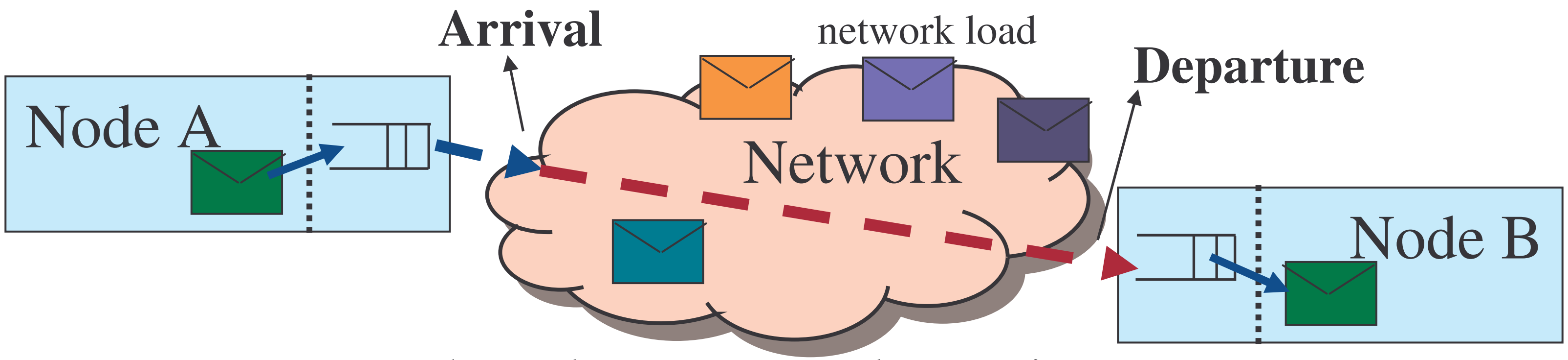
- Real-time messages can have **event or state semantics**



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**

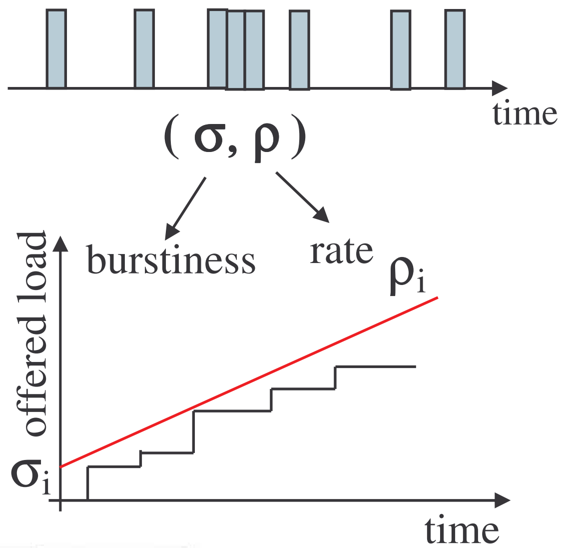
Traffic models



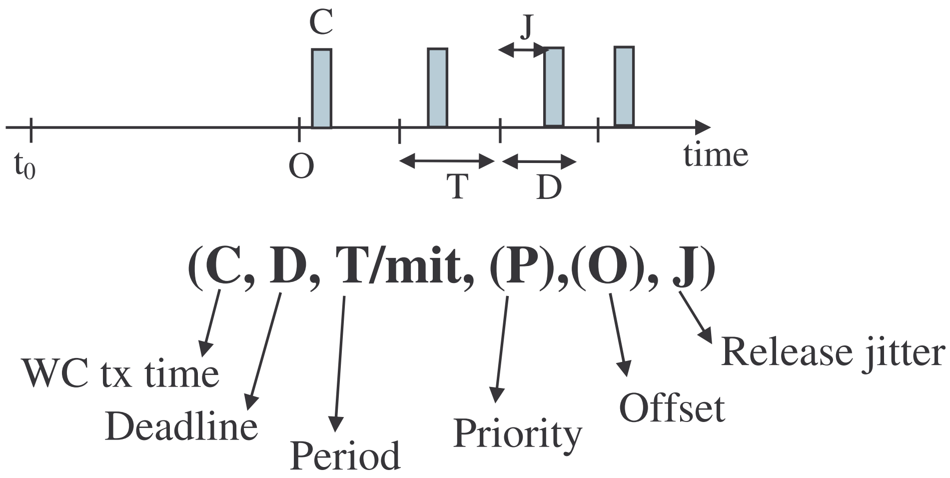
Throughput ≤ Network capacity

(σ,ρ)-model

(Cruz, 1991)

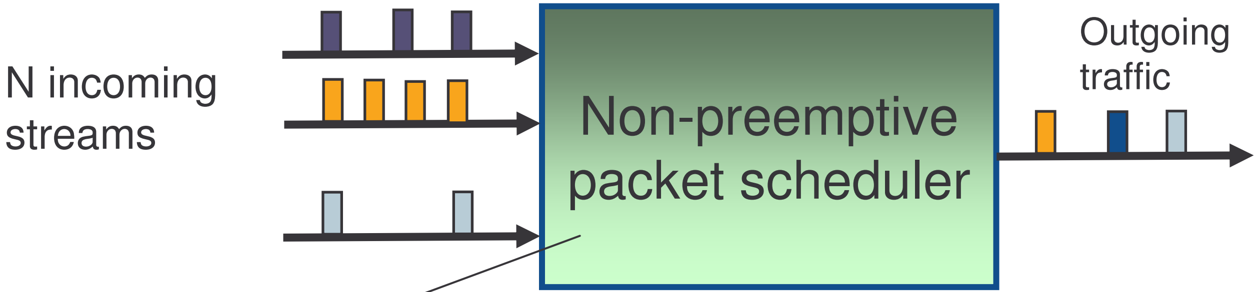


Recurrent model



Traffic scheduling model

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



Scheduling policies

- EDF
- RM
- DM
- FCFS
- WFQ
- …

Knowing the **scheduling policy** and the **arrival pattern of the incoming** flows allows determining the **departing pattern** (including the network delays) of the **outgoing flow**

Scheduling policies

- Static priorities
  - RM – Rate Monotonic
  - DM – Deadline Monotonic
  - FP – Arbitrary Fixed Priorities
- Dynamic priorities
  - FCFS – First Come First Served
  - EDF – Earliest Deadline First
  - WFQ – Weighted Fair Queuing
- Sub cases
  - D=T, D<=T, arbitrary deadlines (D>T)
  - J>0, J=0 → release jitter, deviation from exact periodic release
  - B>0, B=0 → blocking by lower priority messages

Scheduling policy limitations

- The **traffic scheduling** is frequently imposed by the **network protocol**

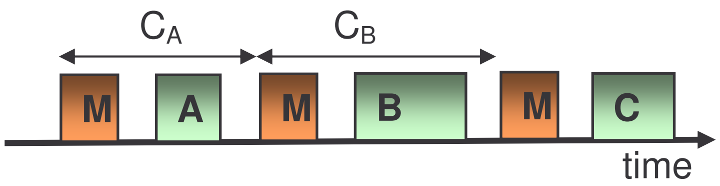
- **The following analysis applies mainly to**
  - **Master-slave systems**
    - The traffic scheduling can be any!
  - **TDMA systems**
    - The traffic scheduling inside each slot can also be any!
  - **Priority-based arbitration mechanisms**
    - Such as CAN or mini-slotting (FlexRay – dynamic part)

Computing the Tx time

- **Transmission time *C***
  - *Maximum_number_of_bits / bit_rate + InterFrame_Space*
    - **e.g., TTP/C:**

$$C = \frac{(SOF(3) + header(4) + data + CRC(16))}{2\text{Mbit/s}} + IFG(10 - 100\mu s)$$

  - In **Master-Slave** systems consider *master token* + *slave answer*

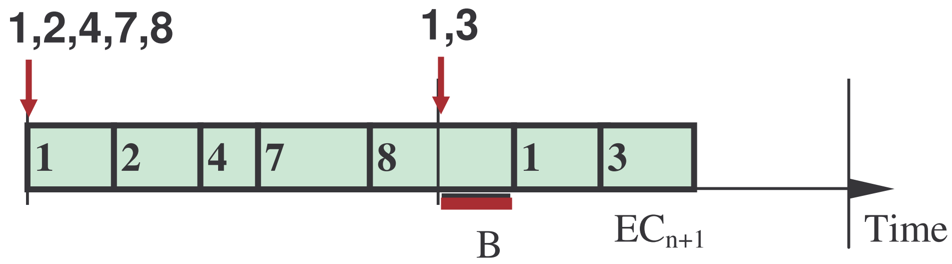


  - Attention:
    - **Number of bits might vary** with the **data value** (CAN)
    - **Bit_rate might vary dynamically** with errors in the channel (WiFi)

Utilization-based schedulability tests
(recurrent model)

- **Based on** bandwidth utilization $(U_i=C_i/T_i)$
  - **EDF: Earliest Deadline First** (Dynamic priorities, D=T, J=0, B>0)
  - **RM: Rate-Monotonic** (Static priorities, *idem*)

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



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

Single utilization test

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

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

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

Utilization-based schedulability tests
(recurrent model)

- **Based on** bandwidth utilization **($U_i=C_i/T_i$)**
  - **EDF: Earliest Deadline First** (Dynamic priorities, D=T, J=0, B>0)
  - **RM: Rate-Monotonic** (Static priorities, *idem*)

N tests
(more accurate)

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

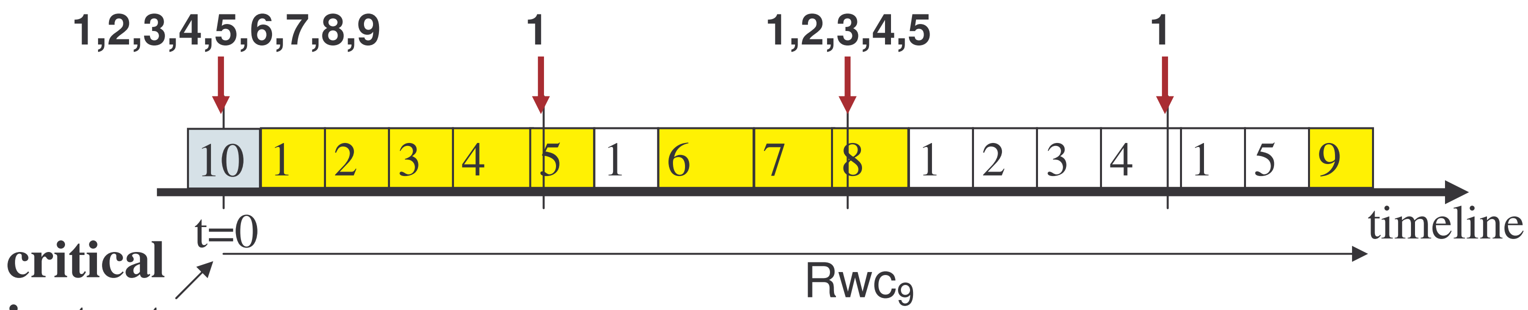
EDF: $$\forall_{i=1..N} \sum_{j=1}^{i} \frac{C_j}{T_j} + \frac{B_i}{T_i} < 1$$

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

Response time-based schedulability tests
(recurrent model)

- **Based on** a response time upper bound **(Rwc_i)**
  - **FP: Arbitrary fixed priorities** ($D \leq T$, $J=0$, $B>0$)

Consider a 10 messages set with $T_1=1$, $T_{2..5}=2$, $T_{6..10}>3$ (t.u.)



$$Rwc_i = w_i + C_i$$

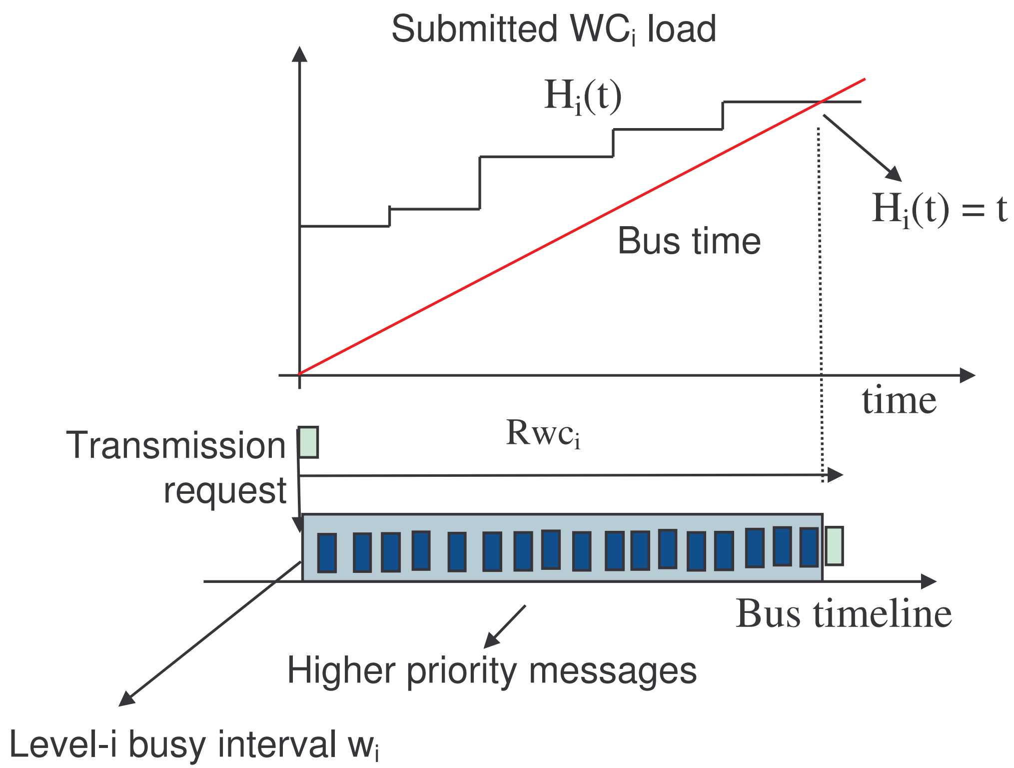
$$w_i = B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i + \tau}{T_j} \right\rceil * C_j$$

bit time (τ)

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

Computing the response time upper bound
(recurrent model)

- **Demand** versus **supply** approach



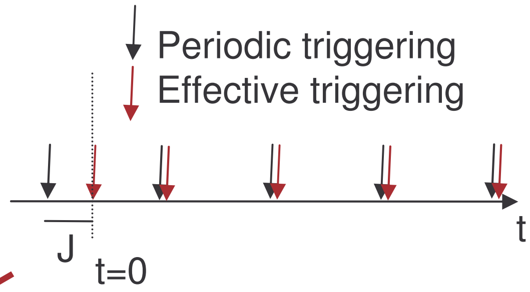
$$Rwc_i = w_i + C_i$$

$$w_i = \min\left(t_{>0}\right) : H_i(t) = t$$

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

Response time with release jitter
(recurrent model)

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



Depends on time origin for Rwc

$$Rwc_i = (J_i) + w_i + C_i$$

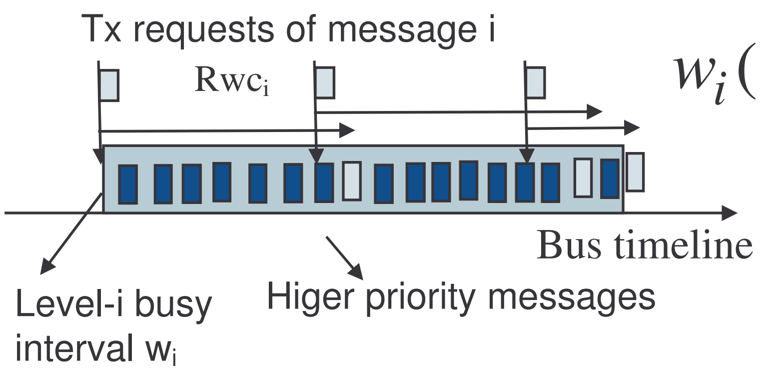
$$w_i = B_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{J_j + w_i + \tau}{T_j} \right\rceil * C_j$$

Response time with arbitrary deadlines
(recurrent model)

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

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

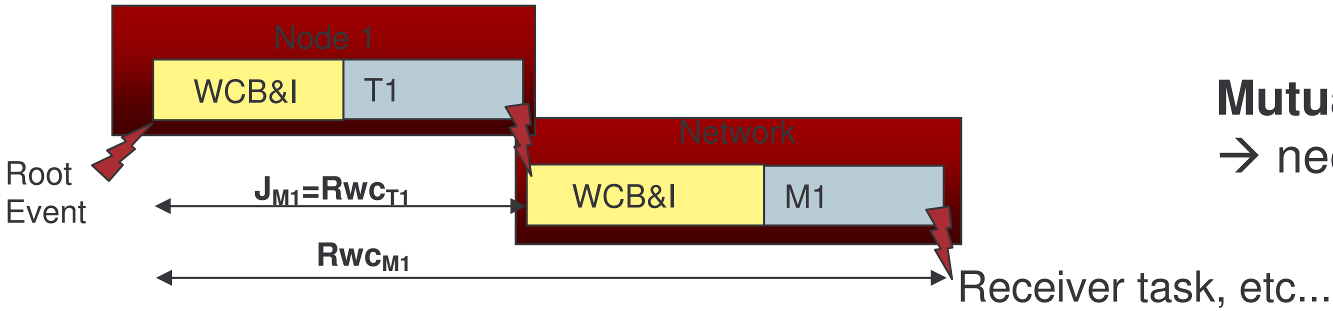
$$w_i(q) = B_i + qC_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{J_j + w_i(q) + \tau}{T_j} \right\rceil C_j$$



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

Computing the release jitter

- Depends on how it is generated (normally inherited)
- Case of periodic tasks
  - **1st approach: J=Rwc** of sender task
  - **better approach: J=Rwc-Rbc** of sender task (worst-best case)



Mutual dependencies
→ needs to be iterated

Computing the exact WCRT might be too costly...
One possibility: *consider WCRT=D*

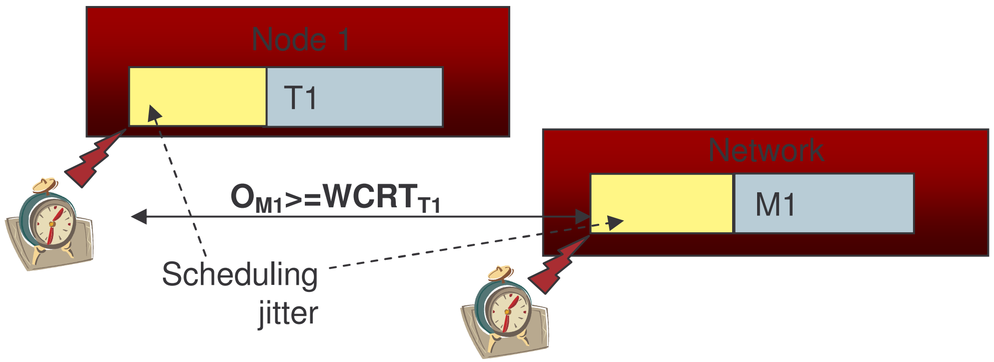
$$J_{M_k} = J_i + B_i + C_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{J_j + D_i + \tau}{T_j} \right\rceil * C_j$$

Interference by tasks in the node of the sender task (i)

Getting rid of release jitter

✓ Break immediate dependencies

- ✓ Use a time-triggered model:
- ✓ ***Periodic triggering with offsets***

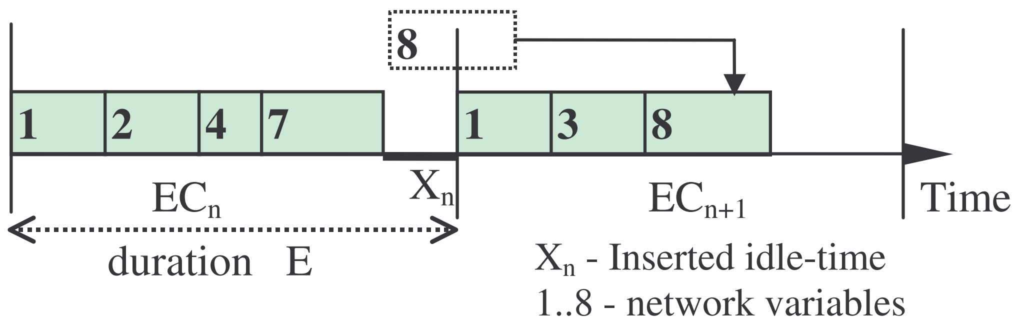


Computing minimal offsets is equivalent to computing release jitter !!

Simplifying approaches
- Use offsets = D of sender
- For local schedulability analysis, consider critical release (no offsets)

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**! (B=0)
  - In this case, **any analysis** for **preemptive** scheduling can be used with inflated transmission times (C')



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

This kind of scheduling is **not strictly work-conservative**

Inserted idle-time compensation factor

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 (B=0)
  - With Release Jitter (J>0)

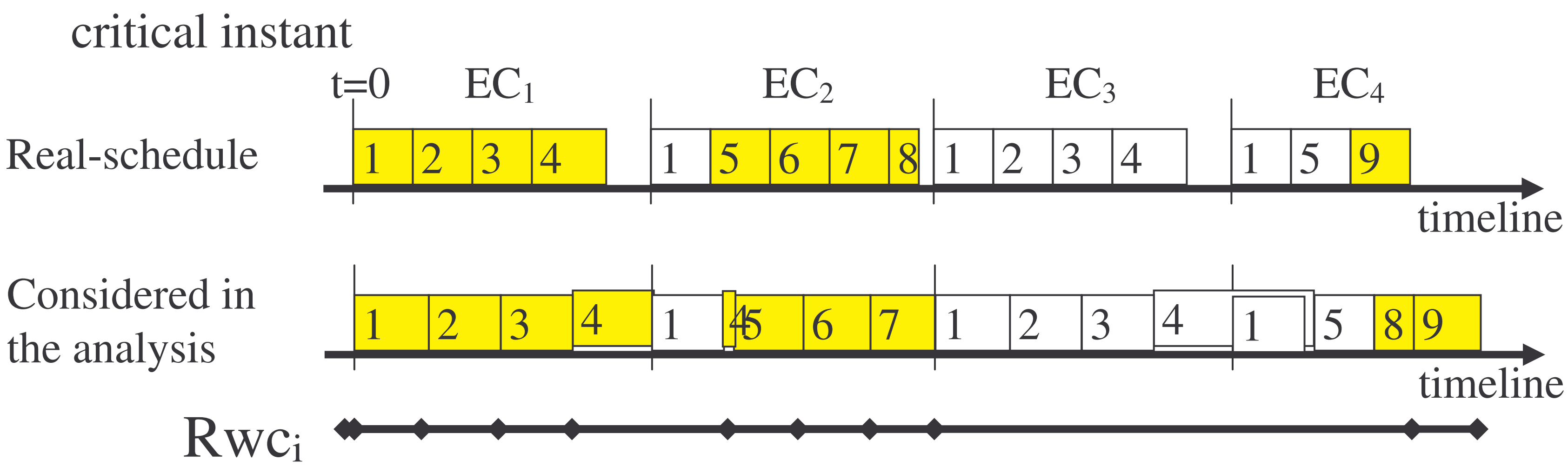
$$\text{RM:}\quad \forall_{i=1..N} \sum_{j=1}^{i} \frac{C'_j}{T_j} + \frac{\max_{j=1..i}(J_j)}{T_i} < i(2^{1/i} - 1)$$

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

Can also be simplified to **one single test** for each policy
(but attention to pessimism!)

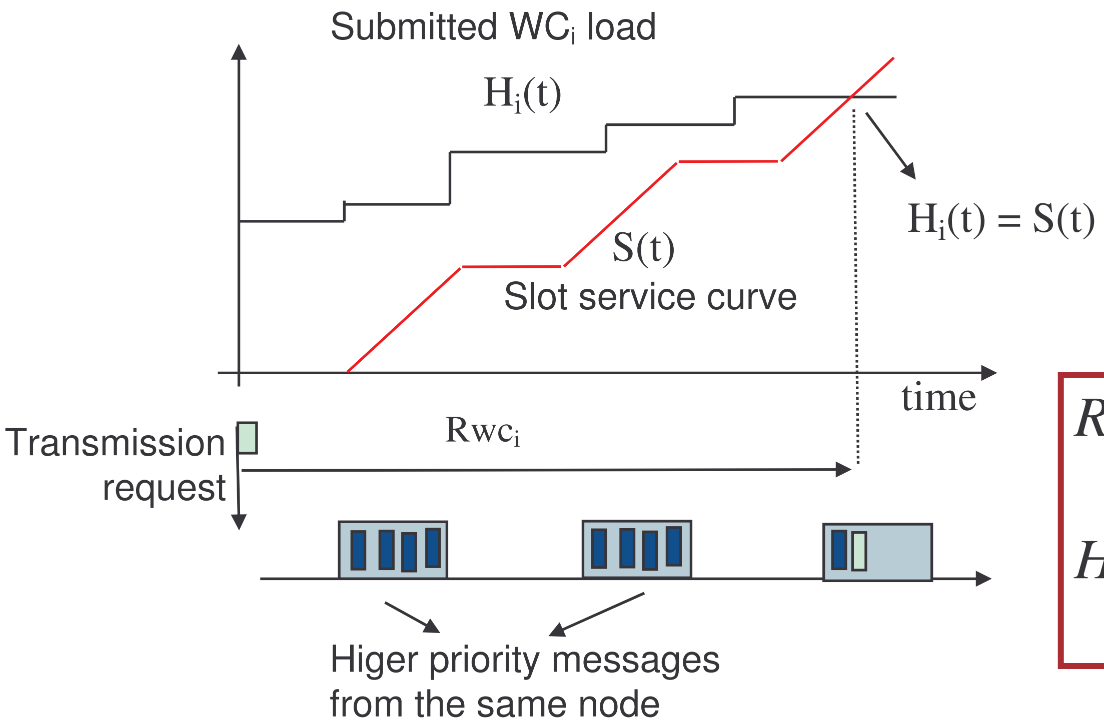
Scheduling within slots/cycles
(recurrent model)

- 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)

- **Demand** versus **supply** approach



Note: remember that activations are synchronous with the cycles

$$Rwc_i = \min(t) : H_i(t) = S(t)$$

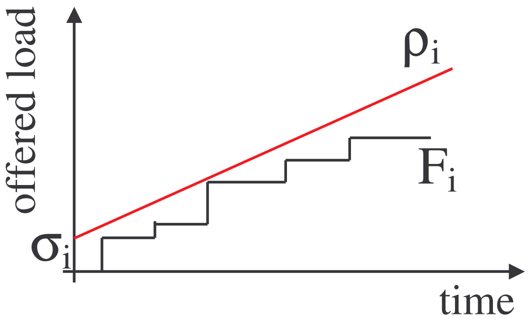
$$H_i(t) = C'_i + \sum_{j\ in\ hp(i)} \left\lceil \frac{t+\tau}{T_j} \right\rceil * C'_j$$

Note2: also remember that this analysis (with C') considers preemption

Network Calculus
((σ,ρ)-model)

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

(σ_i,ρ_i) → TSPEC



in bits

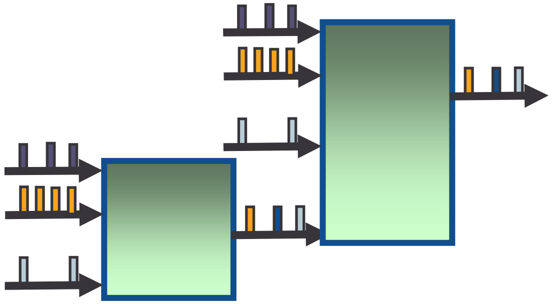
$$D_i = \frac{\sum_{j=1}^{i}\sigma_j + \max_{i+1\le j\le N}(C_j)}{c - \sum_{j=1}^{i-1}\rho_j} \quad , \quad c \ge \sum_{j=1}^{N}\rho_j$$

Stability condition

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

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



Using scheduling tables

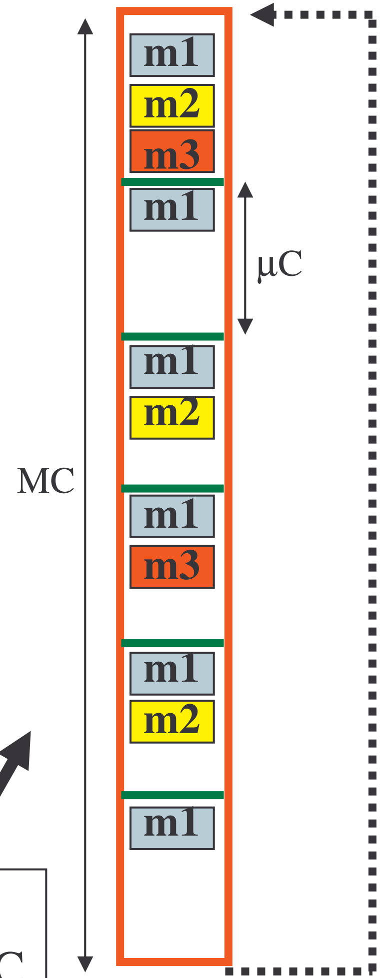
- Some systems use **static table-based** scheduling **(e.g., WorldFIP, TTP/C…)**
- The schedule is **built off-line** and it is typical to use optimization techniques to **optimize the schedule**
  - e.g. wrt to **jitter** or **end-to-end delays**

  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*)

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

GCD=5ms → μC
LCM=30ms → MC
MC={6μC}

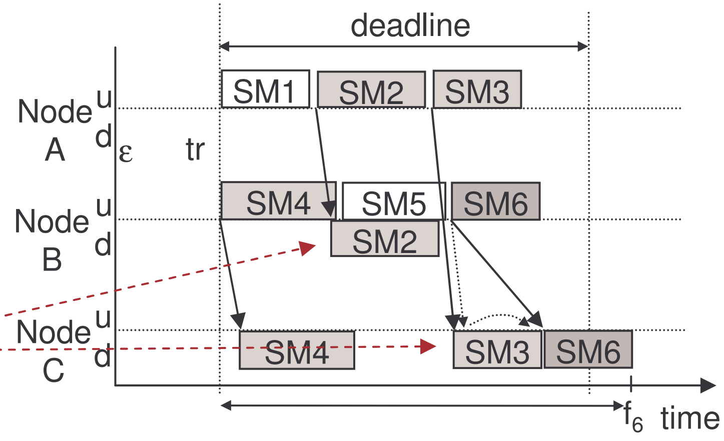


Constraints imposed by the MAC

- **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**
  - to handle aperiodic communication requests
    - e.g. Master-Slave, Token-Passing

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

Some open issues

- **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
- **Release jitter computation**
- **Jitter control at the device-drivers level**
- **Composition of segments**

Rewinding

Rewinding...

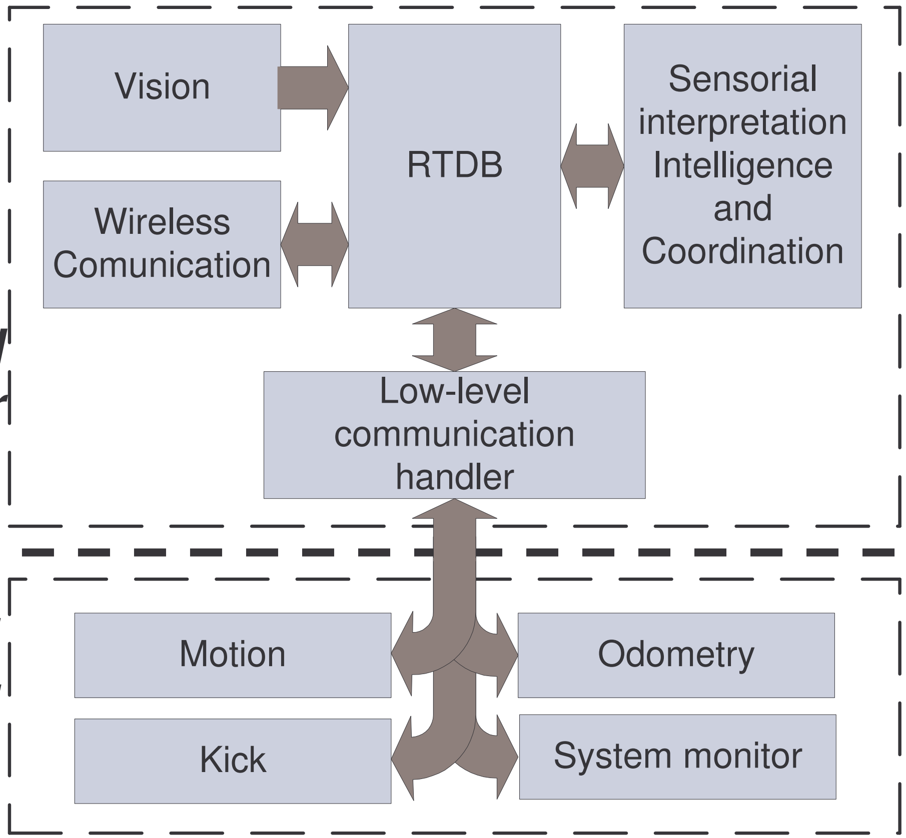
- Computing the **network delays** requires knowledge of the **protocols used**
  - Different traffic scheduling policies require different analysis
- Many of the **existing analysis** are still **pessimistic** → lead to **low efficiency**
  - Mainly **utilization bounds**
    - And worse with **blocking, release jitter and offsets**
- **The whole** protocol stack **must be revisited for** improved temporal behavior

Some practical examples

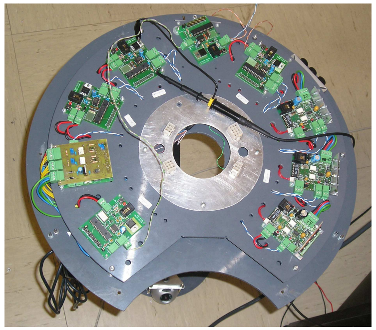
CAMBADA
A RoboCup MSL soccer team

http://www.ieeta.pt/atri/cambada/

Robots functional architecture



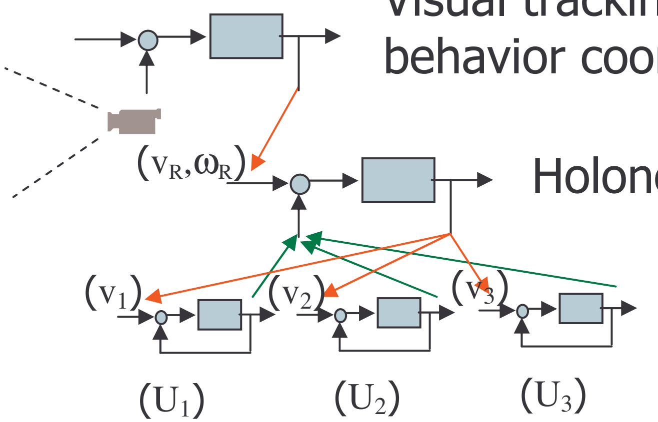
Higher-level Coordination layer



Lower-level sensing&actuation layer
(**CAN**-based distrib. system)

Robots architecture

Control architecture

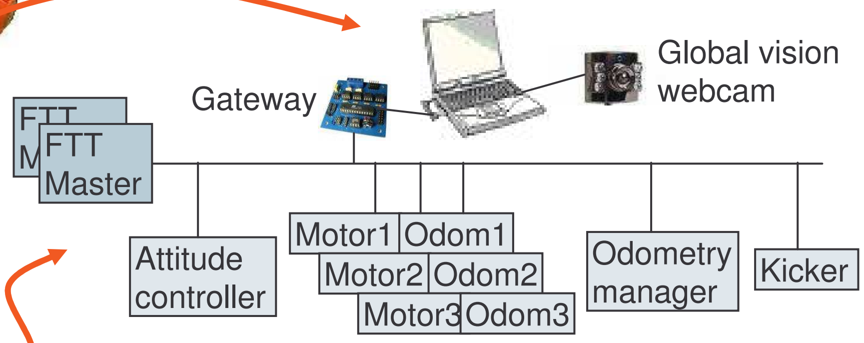


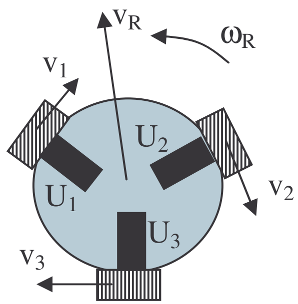
Visual tracking & behavior coordination

Holonomic control

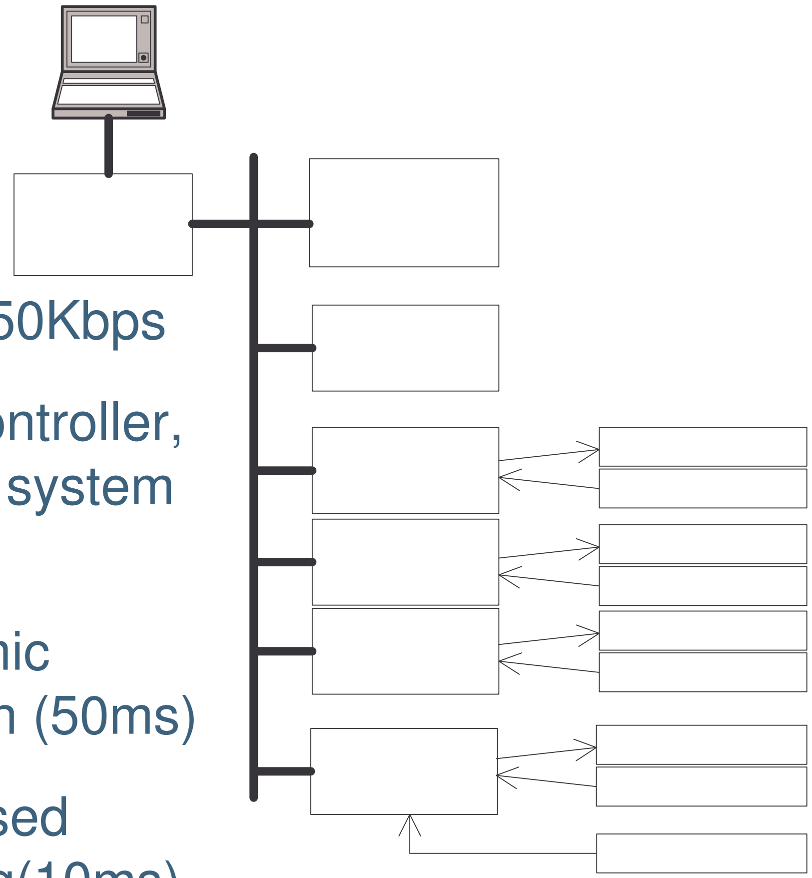
Motor controllers

Hardware architecture





Lower-level sensing and actuation layer

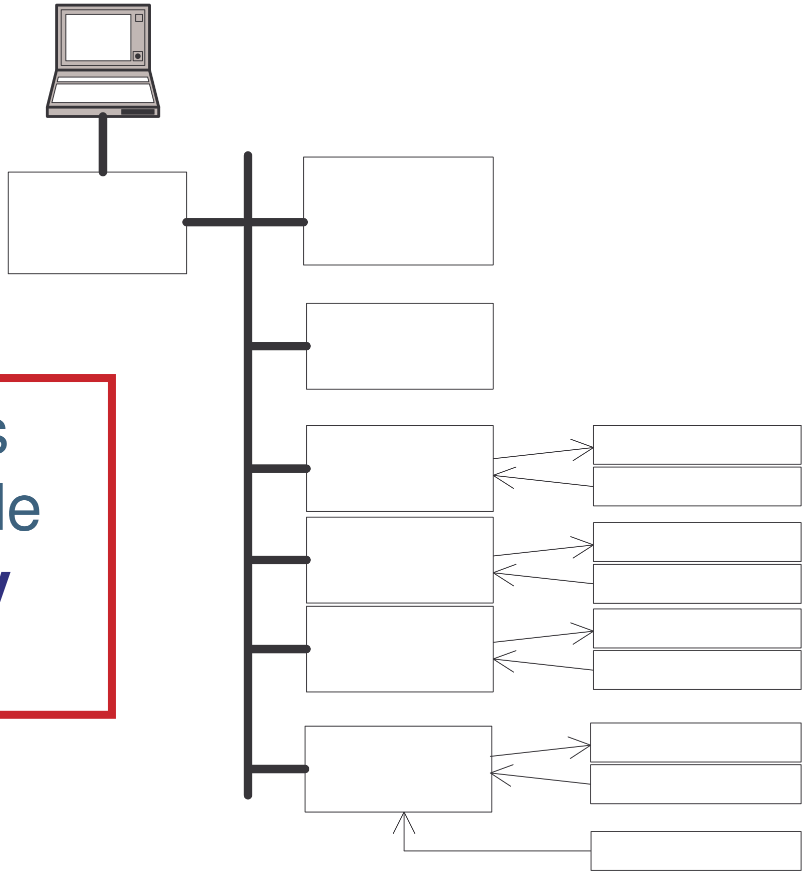


Distributed control system

- Controller Area Network (**CAN**) at 250Kbps
- **3 DC motor** drives, 1 **holonomic** controller, 1 **odometry** manager, 1 **kicker** and system monitor, 1 **gateway**
- **2 main information flows:** holonomic motion (30ms), odometry information (50ms)
- Local cyclic activities: DC-motor closed loop control (5ms), encoders reading(10ms)
- **Unsynchronized chained cycles** may cause large end-to-end delay and jitter

J. L. Azevedo, B. Cunha, L. Almeida. Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. ETFA 2007. Patras, Greece, September 2007.

Lower-level sensing and actuation layer

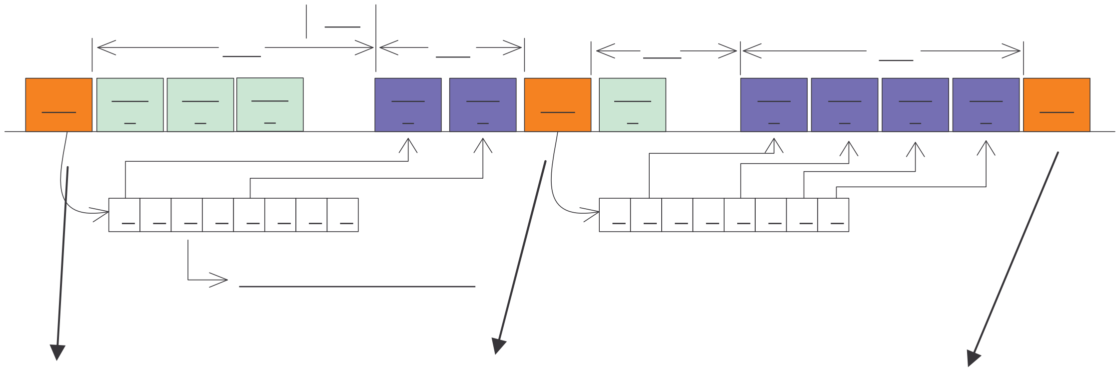


How to **synchronize** activities and communications to provide **bounded end-to-end latency** with **low jitter**?

Lower-level sensing and actuation layer

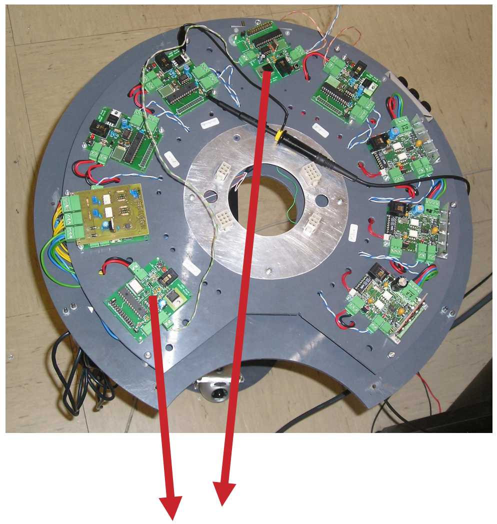
Two implementations

- **Unsynchronized direct** use of Controller Area Network (send/receive)
- **Synchronized framework** based on **FTT-CAN (Flexible Time-Triggered CAN)**



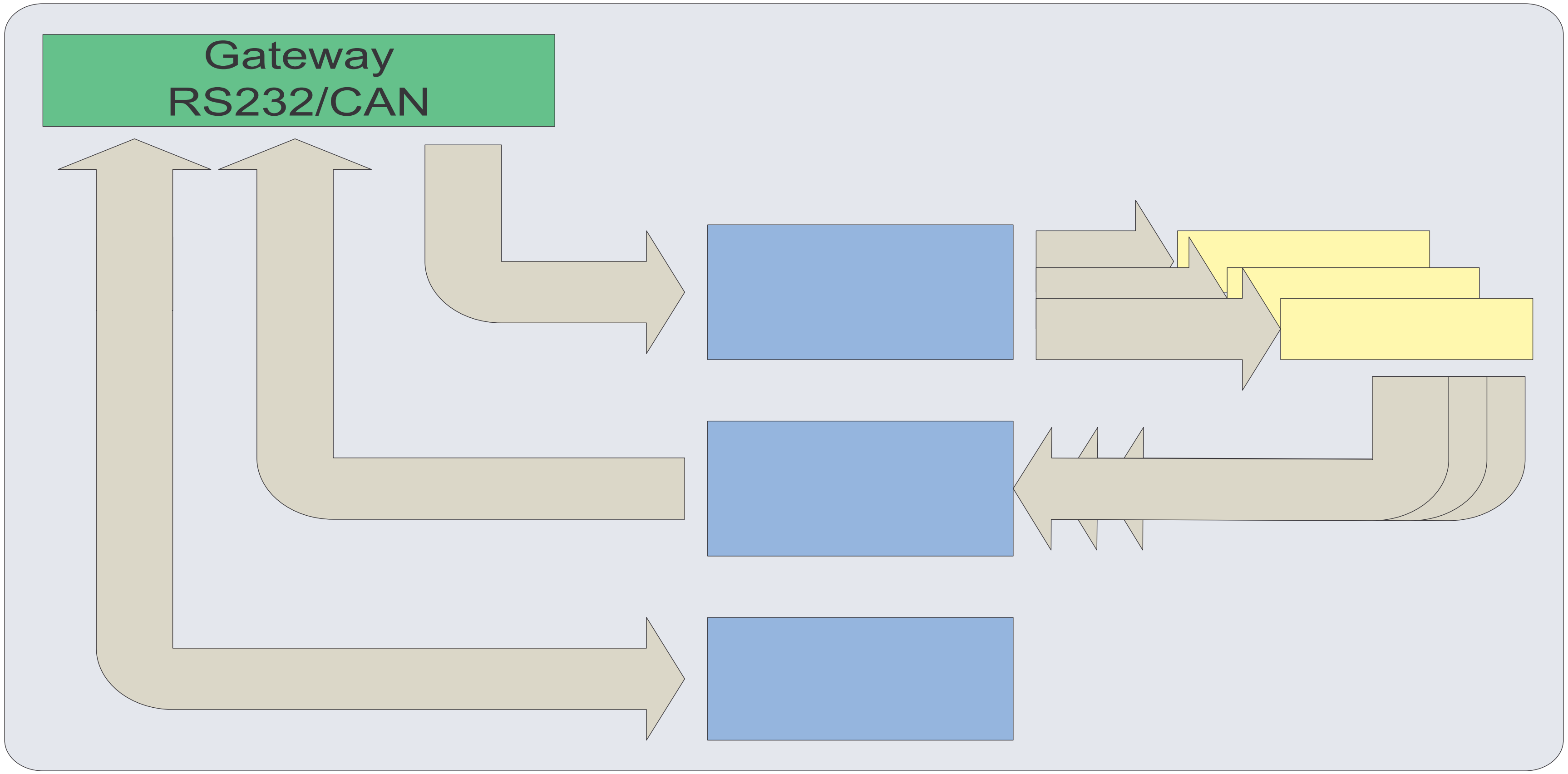
Trigger message sent regularly by the Master every Elementary Cycle:

Triggers synchronous **messages and tasks**



FTT-CAN replicated masters

Information Flows



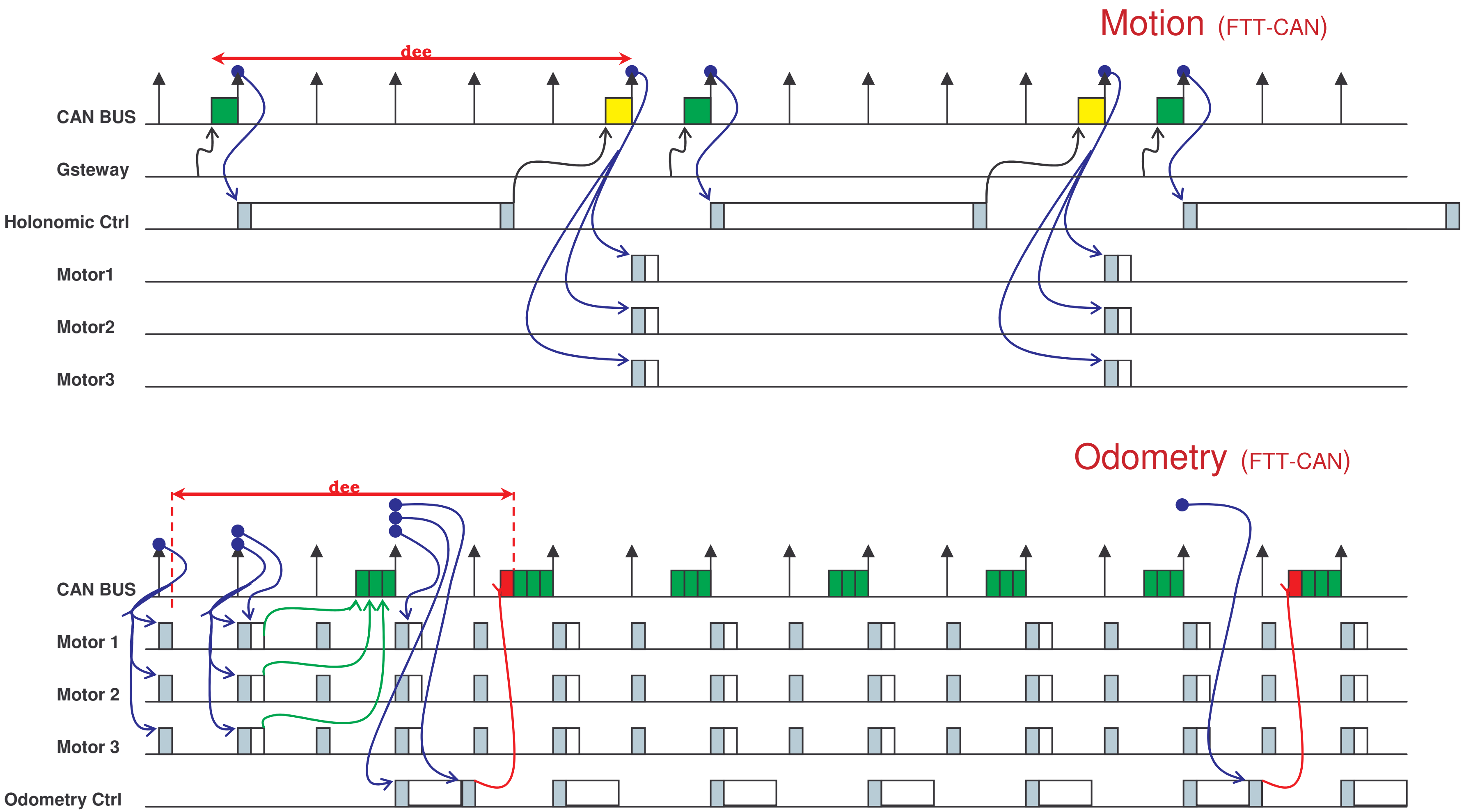
Low level communication requirements

ID	Source	Target	Type	Period/ mit (ms)	Size (B)	Short description
M1	Holonomic ctrl	Motor node [1:3]	Periodic	30	6	Agregate motor speeds setpoints
M2	Kicker	Gateway	Periodic	1000	2	Battery status
M3.1– M3.3	Motor node [1:3]	Odometry node	Periodic	5 to 20	3+3	Wheels encoder values
M4.1– M4.2	Odometry node	Gateway	Periodic	50	7+4	Robot Position+orientation
M5.1– M5.2	Gateway	Odometry node	Sporadic	500	7+4	Set/Reset robot position+orientation
M6.1– M6.2	Gateway	Holonomic ctrl	Periodic	30	7+4	Velocity vector (linear+angular)
M7	Gateway	Kicker	Sporadic	1000	1	Kicker actuaction
M8 – M12	Every node	Gateway	Sporadic	1000	5*2	Node hard reset

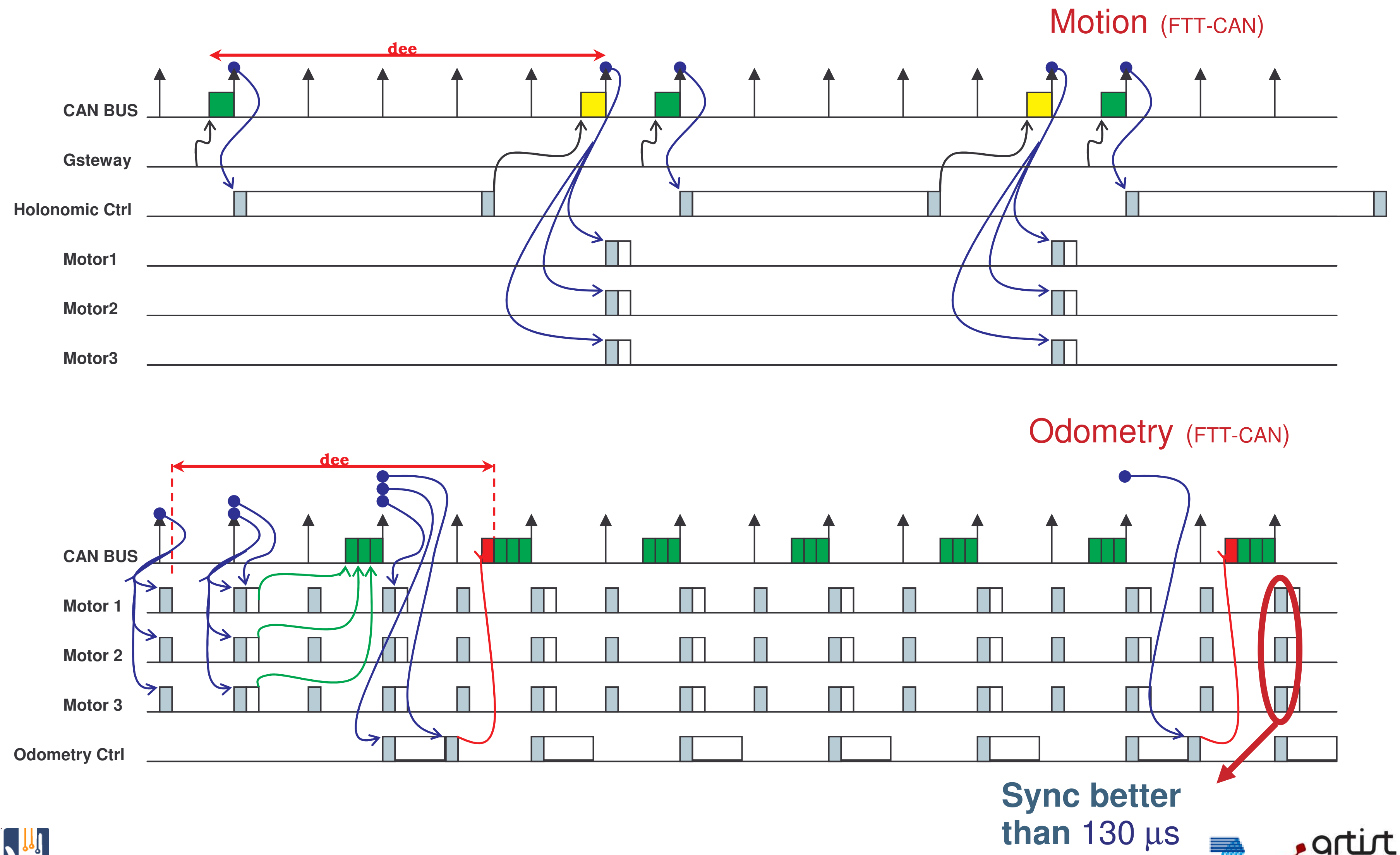
Low level communication requirements

ID	Period/mit (ms)	Prio (RM)	Size (B)	MaxSize (bit)	C (ms) @125kbit/s	C (ms) @70kbit/s	C (ms) @50kbit/s	Rwc(i) (ms) @50kbit/s
M3.1	10	1	3	85	0.68	1.2	1.7	4.2
M3.2	10	2	3	85	0.68	1.2	1.7	5.9
M3.3	10	3	3	85	0.68	1.2	1.7	7.6
M1	30	4	6	115	0.92	1.6	2.3	9.9
M6.1	30	5	7	125	1	1.8	2.5	12.4
M6.2	30	6	4	95	0.76	1.4	1.9	19.4
M4.1	50	7	7	125	1	1.8	2.5	21.9
M4.2	50	8	4	95	0.76	1.4	1.9	28.9
M5.1	500	9	7	125	1	1.8	2.5	31.4
M5.2	500	10	4	95	0.76	1.4	1.9	49.6
M7	1000	11	1	65	0.52	0,9	1.3	50.5
M8	1000	12	2	75	0.6	1.1	1.5	78.4
M9	1000	13	2	75	0.6	1.1	1.5	79.9
M10	1000	14	2	75	0.6	1.1	1.5	81.4
M11	1000	15	2	75	0.6	1.1	1.5	88.0
M12	1000	16	2	75	0.6	1.1	1.5	89.5
M2	1000	17	2	75	0.6	1.1	1.5	91.0
				Utot	33.6%	60%	84%	
				Utot_BT	40.4%	72%	101%	

Cambada – Information flow with FTT



Cambada – Information flow with FTT



Results

- Virtual **elimination** of periodic message **jitter**
- **Shorter end-to-end delay** for message with longer periods (Holonomic motion flow)
- **Acquisition** of the 3 wheel encoders are **synchronized** within 130 µs

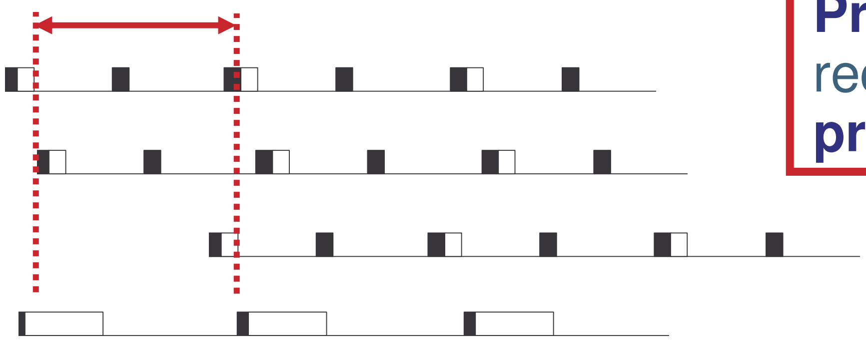
Measure	Without FTT (ms)	With FTT (ms)
Setpoints from Gateway to actuation on motors	38.8 to 64.4	26.7 to 27.7
Encoders acquisition to Gateway reception of actual position	12 to 21	21.6 to 21.7

V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, J. Fonseca. Implementing a distributed sensing and actuation system: The CAMBADA robots case study. IEEE ETFA 2005, Catania Italy, Sept 2005.

The value of synchronization

- The **odometry** manager uses the values of **all 3 encoders** to update the current robot position and orientation.
- If the encoder information is **not sampled synchronously** there will be an **extra error**
- **10ms** of difference, when moving at **2m/s**, may induce (per sec) an error of **2cm** in the **linear displacement** of each motor (~1cm in the robot displacement and ~4.6º in orientation)
- **Jitter** in this difference (e.g. as caused by drift in the nodes clocks) causes this **error to be incremental**

Unsynchronized streams

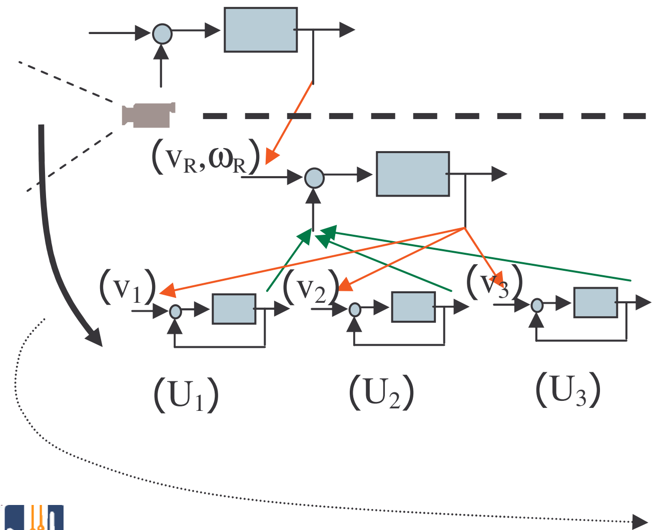


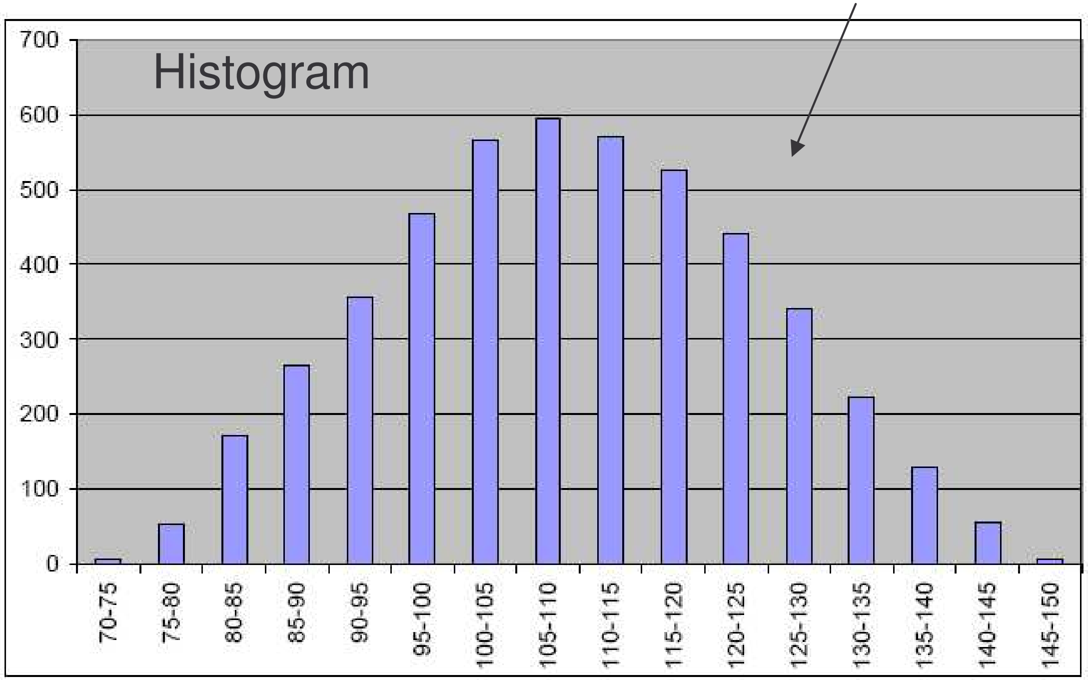
Precise movement control requires **precise synchronization**

Synchronizing the higher and lower levels

- The higher-level coordination layer and the lower-level sensing and actuation layer are still not synchronized
  - Camera frames and lower-level cycle are not synchronized
  - Causes large and variable delay
  - Being addressed…

We can save ~70ms

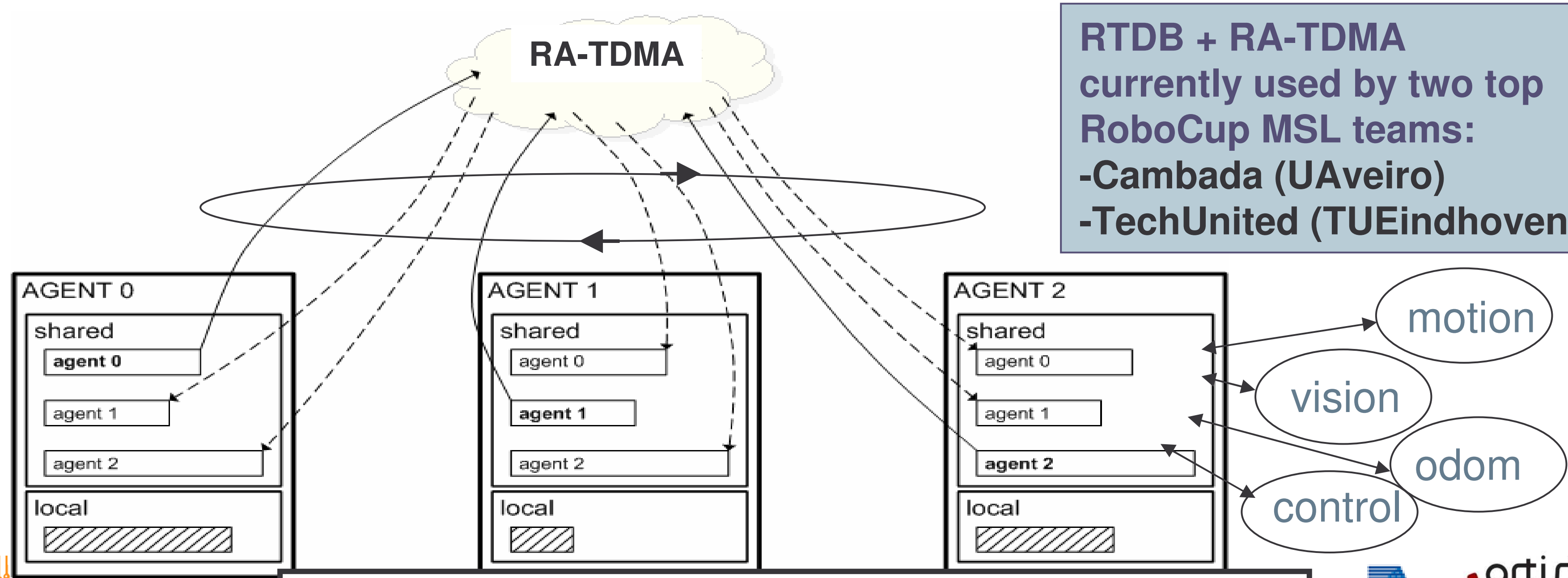




End-to-end delay (ms)

The RTDB middleware

- **Nodes share data** with a **Real-Time DataBase** that holds
  - **Local sensor/state** data gathered from local processes
  - **Images of remote data** updated transparently with RA-TDMA
  - **Remote data used as local**, transparently, with **age information**

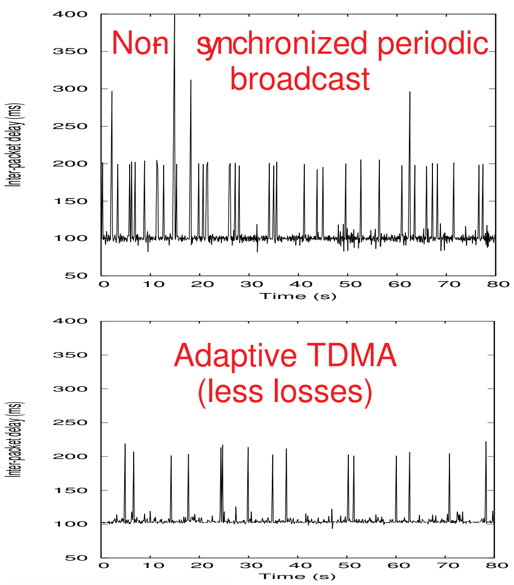


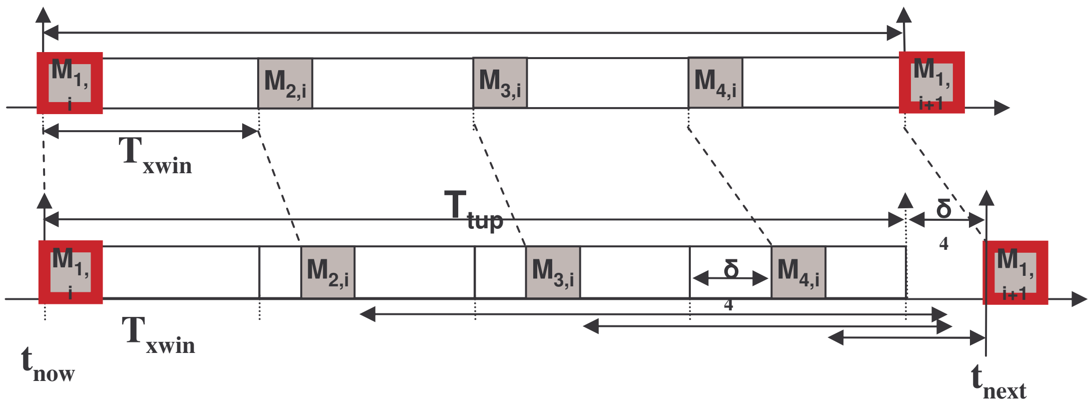
RTDB + RA-TDMA currently used by two top RoboCup MSL teams:
-Cambada (UAveiro)
-TechUnited (TUEindhoven)

F. Santos, L. Almeida, P. Pedreiras, L. S. Lopes. A real-time distributed software infrastructure for cooperating mobile autonomous robots. ICAR 2009. Munich, Germany, 24-26 June 2009.

Adaptive (A-)TDMA

- <u>Team members transmit in sequence</u>
  - **TDMA set on top of CSMA-CA** of IEEE802.11
  - Virtually **eliminates collisions** among **team members**
  - **Fully distributed synchronization** based on frame receptions
  - Shifts phase of TDMA round to match periodic interference
  - **Time constraints → TDMA round period Ttup**





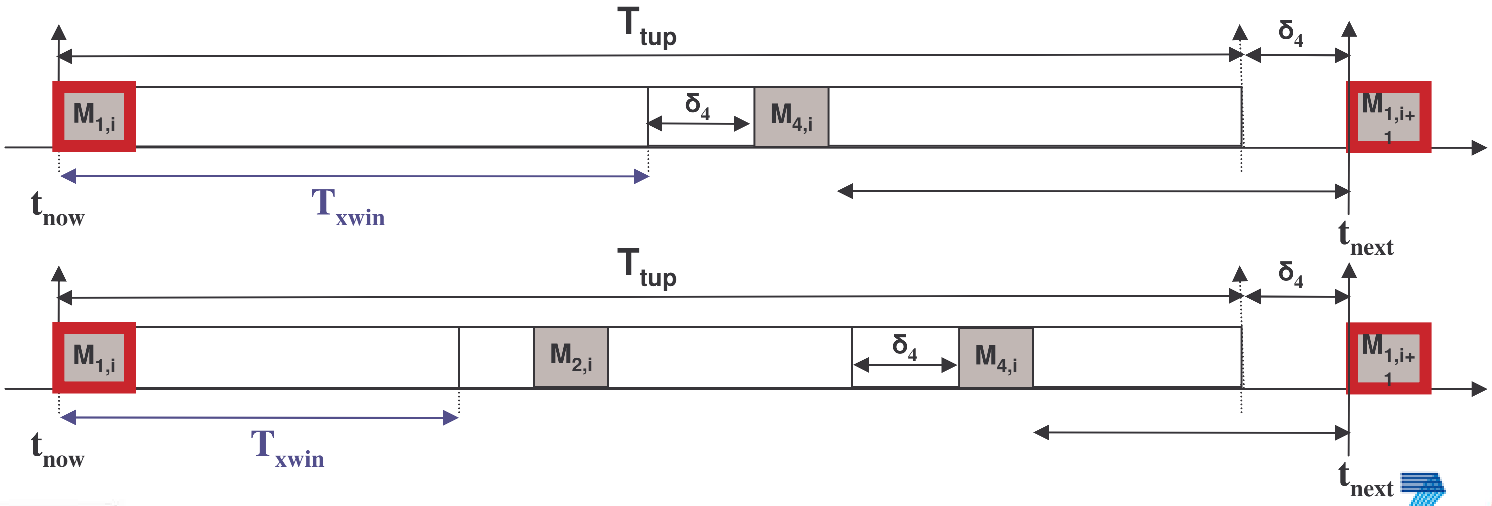
If $0<\delta_k<\Delta$ then $t_{next} = t_{now}+T_{tup}+max_k(\delta_k)$
round period will be within $[\ T_{tup}\ ,\ T_{tup}+\Delta\ ]$

Reconfigurable and Adaptive (RA-)TDMA

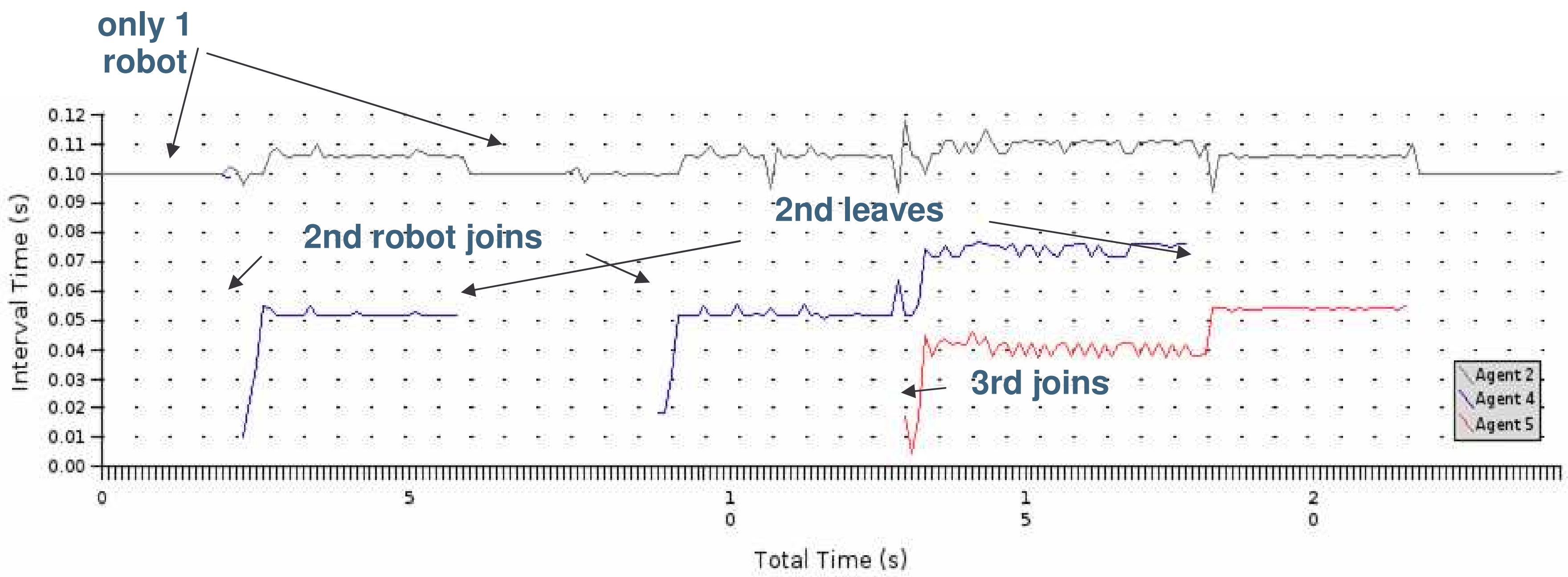
- Dynamic reconfiguration of the slot structure
  - Robots join and leave dynamically
    - crash, maintenance, movements...
  - Slot structure of TDMA round need not be predefined
    - Number of slots continuously adjusted as needed
  - **Fully distributed – minimal a priori knowledge**

2 running robots:

3 running robots:



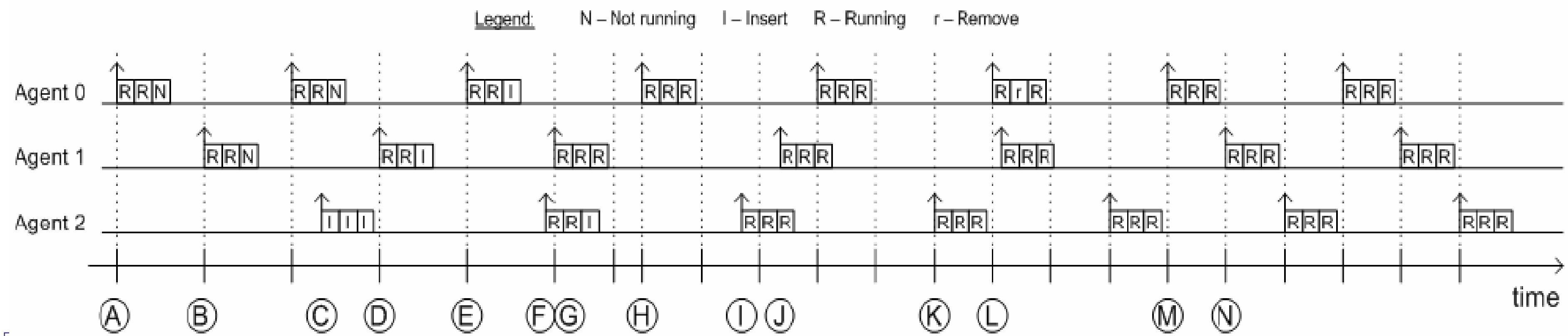
Reconfigurable and Adaptive (RA-)TDMA



F. Santos, L. Almeida, L. S. Lopes. Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots. ETFA 2008. Hamburg, Germany, 15-18 September 2008.

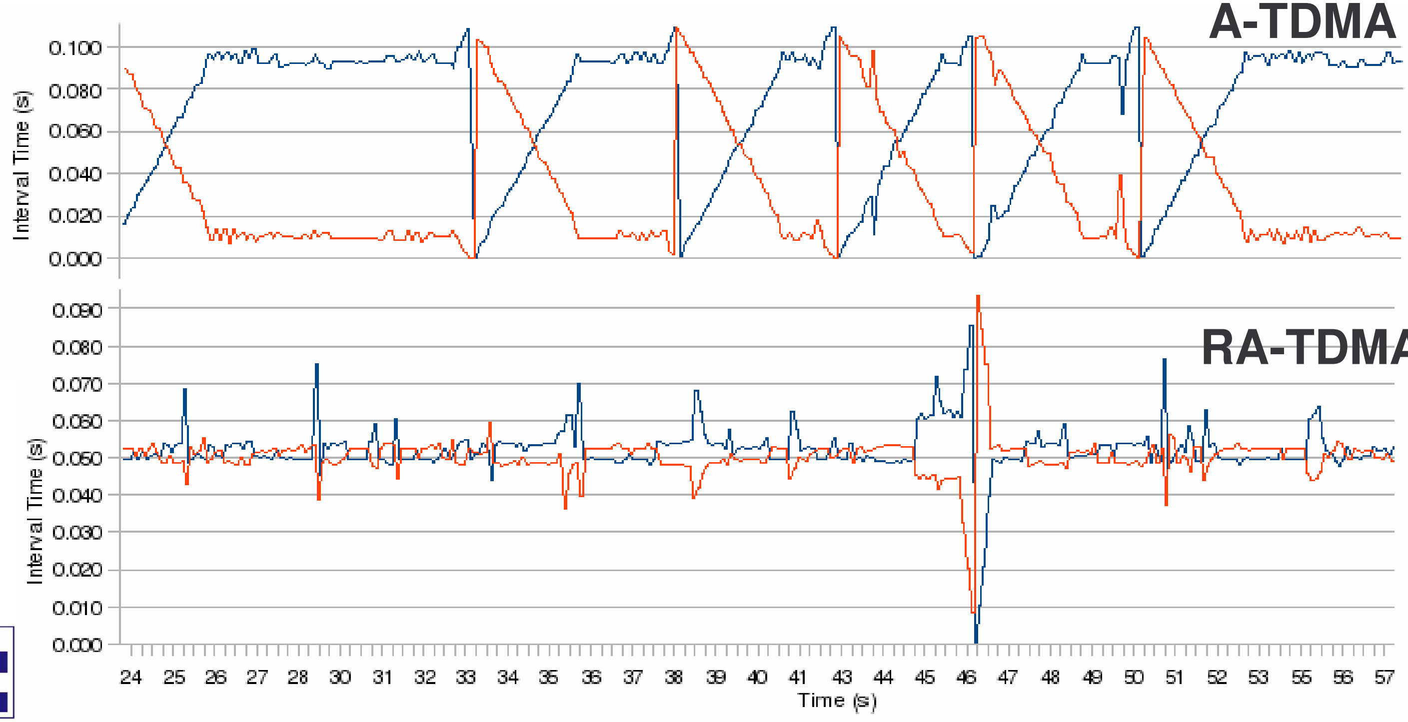
Reconfigurable and Adaptive (RA-)TDMA

- Using an AP simplifies team membership definition **and** speeds up the agreement process **for reconfigurations**
  - **Topology** becomes virtually **fixed**
  - **Agreement** takes about **one TDMA round**
    - For all nodes to reach consensus on the reconfiguration to be done
  - **Synchronization** takes a **few more** rounds (**bounded**)
    - For all nodes to synchronize with the new round structure



Reconfigurable and Adaptive (RA-)TDMA

- **Main advantages**
  - Absence of a fixed TDMA round structure
  - Fully distributed startup procedure with minimal configuration
  - Further contribution to reduce collisions



A-TDMA (2 nodes/10 slots)

in sync: **50%**

lost packets:**1,2%**

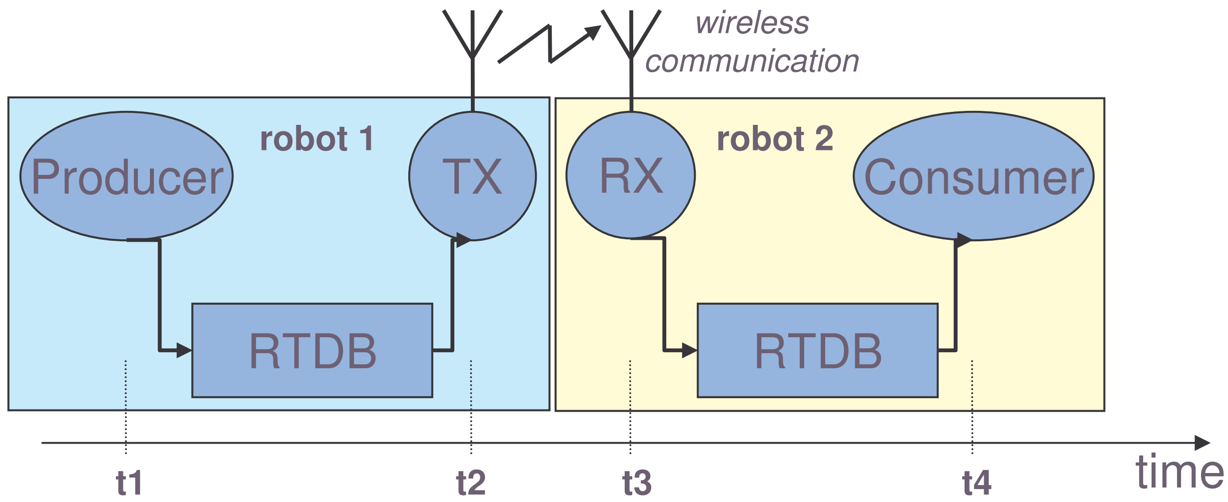
RA-TDMA (2 nodes)

in sync: **97%**

lost packets: **0,3%**

RTDB – Age of the data

- **No global clock synchronization → use age (relative)**

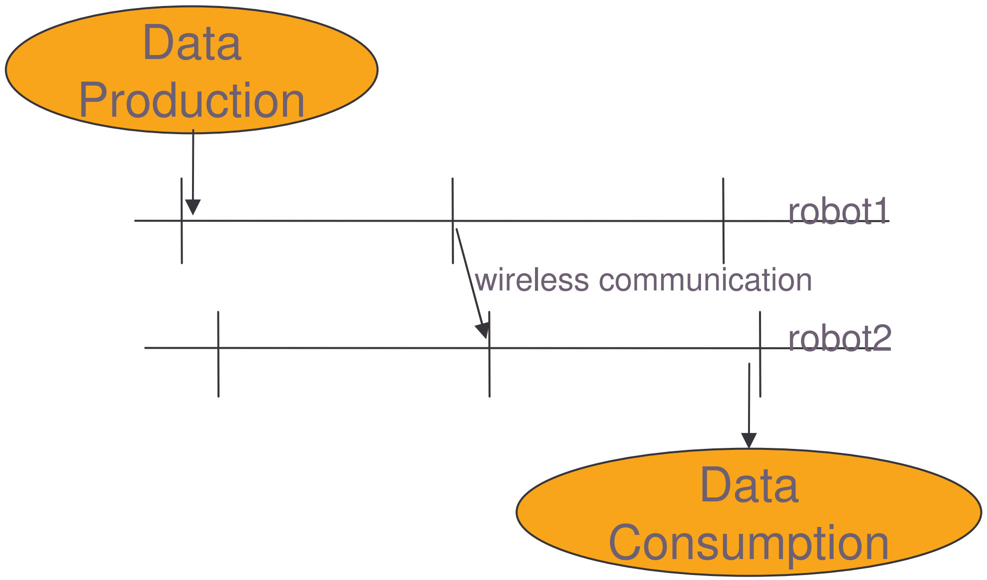


(tx – local timestamps)
t1 – Robot 1 produces and writes data into the RTDB
t2 – Communication protocol fetches data and also sends age (*t2 – t1)*
t3 – Robot 2 writes robot 1 data into the RTDB and subtracts (*t2 – t1*) to *t3*
t4 – Consumer reads data and total data age

age = t4 – (t3 – (t2 – t1)) + wireless_communication_delay

RTDB – Age of the data

- **Maximum age (worst-case)**



Data produced just after a communication broadcast

Data consumed just before a new reception from robot1, with updated info

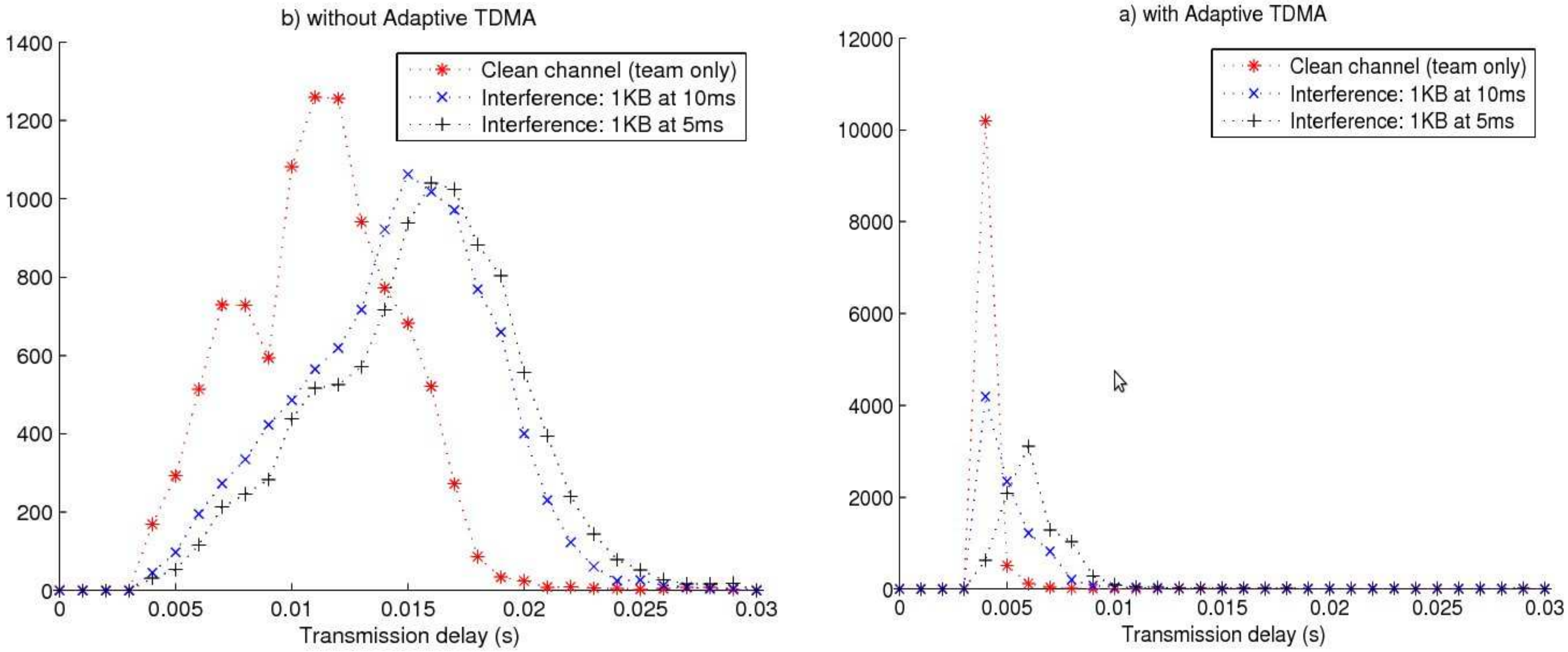
$$max\ data\ age = min(T_{rcpp}, T_{dup} * T_{tup}) + T_{wt} + (T_{dup} * T_{tup})$$

Wireless communication delay

Producer period

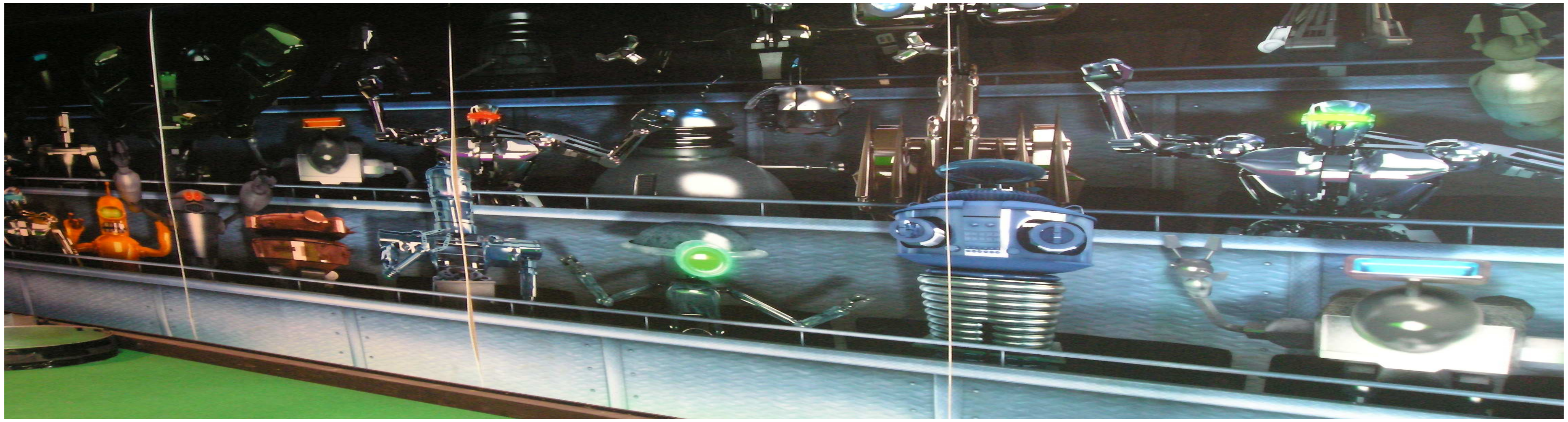
Network period for this item

Wireless communication delay T_{wt}



Configurations: Managed communication; 379 bytes packet size;
4 robots; T_{tup} = 50ms
All robots start communication at the same time using an external trigger signal

The CAMBADA robotic soccer team



Videos…

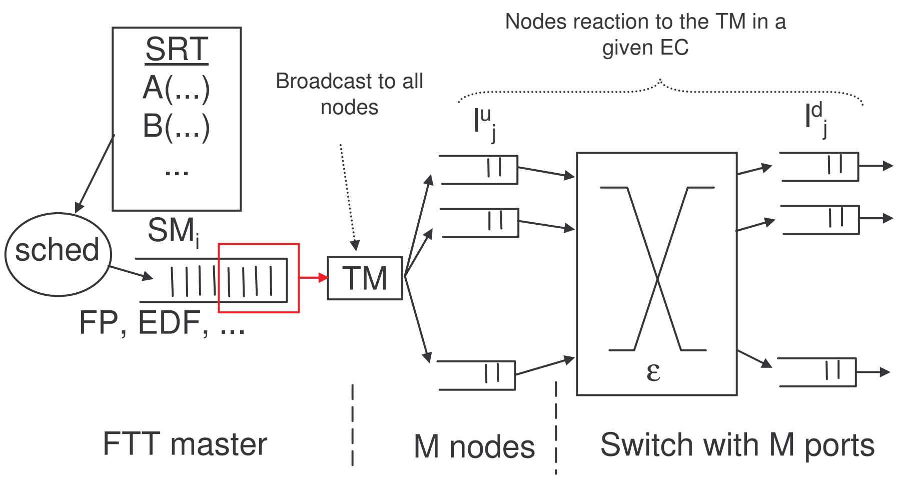
CAMBADA

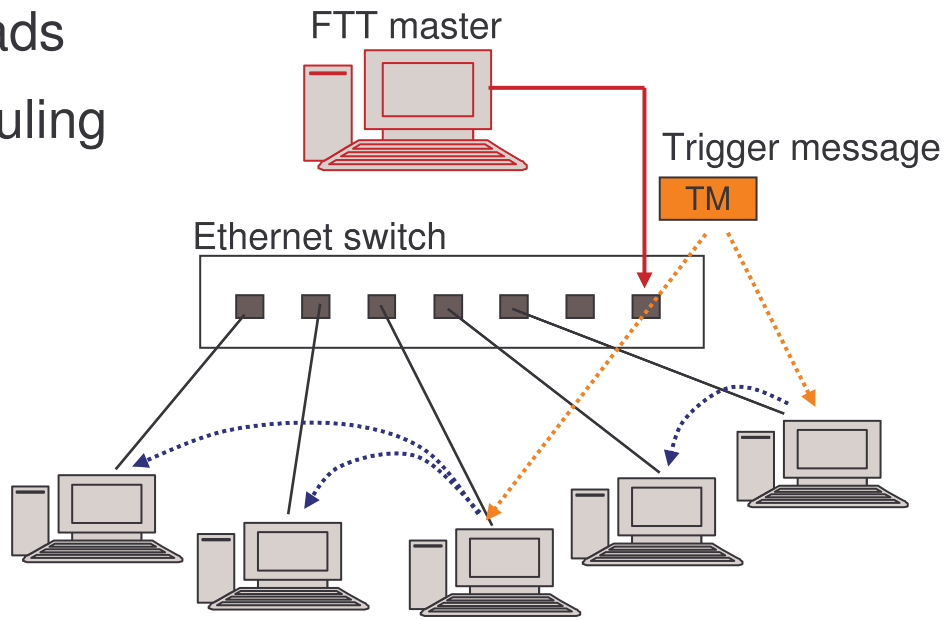
http://www.fe.up.pt/ftt/

FTT-SE
Flexible Time-Triggered Switched Ethernet

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



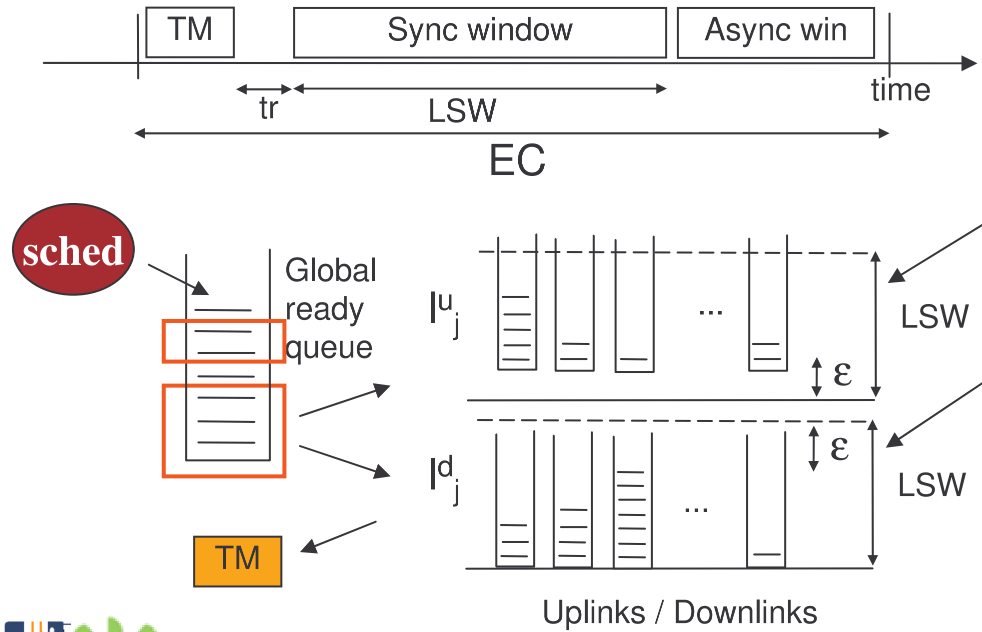


R. Marau, L. Almeida, P. Pedreiras. Enhancing Real-Time Communication over COTS Ethernet switches. WFCS 2006, IEEE 6th Workshop on Factory Communication Systems, Turin, Italy. June 2006.

FTT-SE

- Scheduling model for periodic traffic
  - Set of periodic streams (**synchronous** traffic)

    SRT = $\{SM_i: SM_i(C_i, D_i, T_i, O_i, Pr_i, S_i, \{R^1_i .. R^{k_i}_i\}), i=1..N_s\}$
  - Scheduling with multiple queues
  - Strictly confined to the Synchronous Window per EC



Scheduling equation (building ECs)

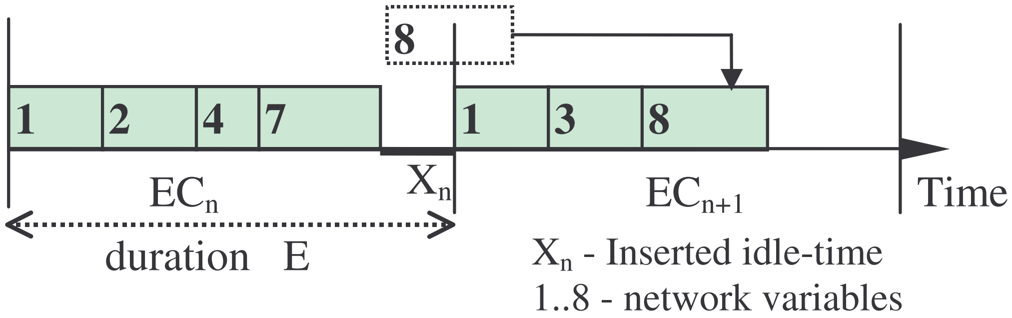
$$\begin{cases} \max_j \left(\sum_{SM_i \in l_j^u} C_i \right) \le LSW - \max_{SM_i \in l_j^u} \varepsilon_i \\ \max_j \left(\max_{SM_i \in l_j^d} (f_i) \right) \le LSW \end{cases}$$

Memory bounds

$$\max_{j=1..M}(\mu^n_j, \mu^p_j) < (LSW - \varepsilon) * r/8$$

FTT-SE

- Testing schedulability of periodic traffic
- **Basic scheduling model:**
  - **Schedule within partitions** with **strict time bounds**
  - Use **inserted idle-time** (X)
    - There is **no blocking**
    - **Any analysis for preemptive scheduling** can be used with **inflated transmission times** (C')



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

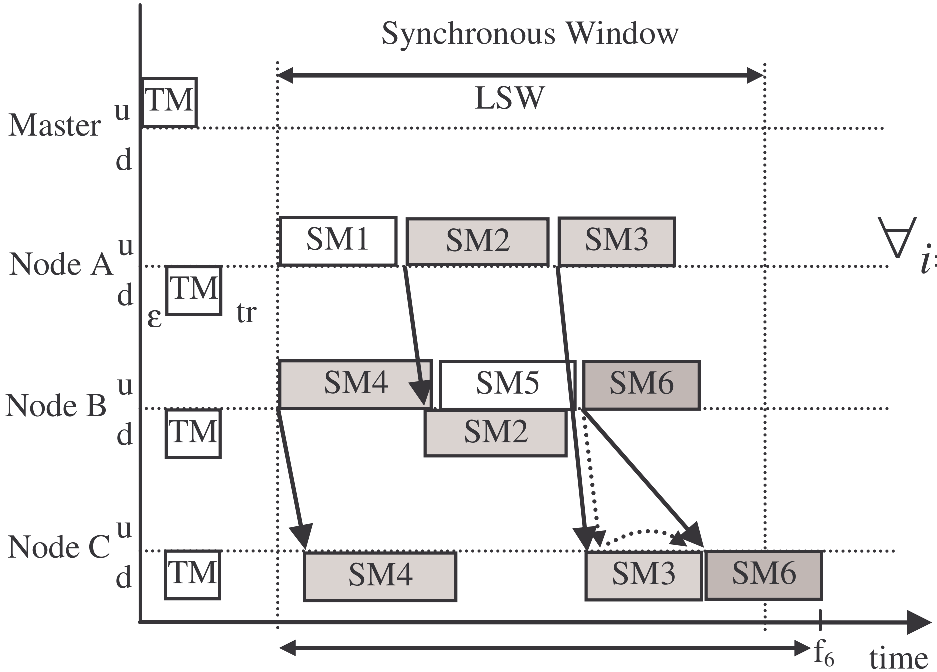
Inserted idle-time compensation factor

We will consider C' as C and use **preemptive analysis** in the following

L. Almeida, J. Fonseca. Analysis of a Simple Model for Non-Preemptive Blocking-Free Scheduling. Proceedings of ECRTS'01, EUROMICRO Conf. on Real-Time Systems, Delft, Holland, June 2001.

FTT-SE

- Testing schedulability of periodic traffic
  - Interference in uplinks appears at downlinks as **release jitter** (J)
  - **Utilization bounds** are important for **on-line QoS management**
    - With release jitter they can be **applied to each link separately**



For each link:

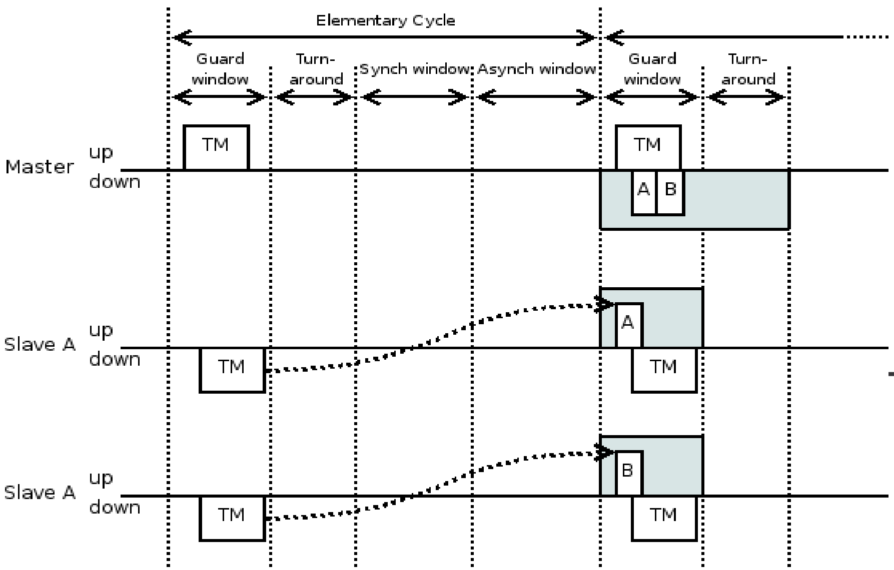
$$\forall_{i=1..n} \sum_{j=1}^{i} \frac{C_j}{T_j} + \frac{\max_{j=1..i} J_j}{T_i} \le U^{lub}_{RM,EDF}(i)$$

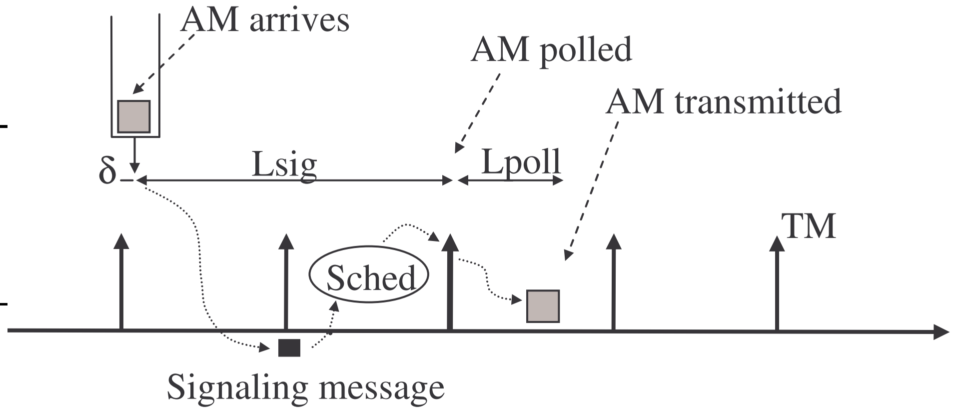
$$\sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{\max_{i=1..n} J_i}{T_1} \le U^{lub}_{RM,EDF}(n)$$

R. Marau, L. Almeida, P. Pedreiras, K. Lakshmanan, R. Rajkumar. Utilization-based Schedulability Analysis for Switched Ethernet aiming Dynamic QoS Management. IEEE ETFA 2010, Bilbao, Spain, Sept 2010.

FTT-SE

- ## Aperiodic traffic
  - Set of sporadic streams (**asynchronous** traffic)
    $ART = \{AM_i: AM_i(C_i, D_i, mit_i, Pr_i, S_i, \{R^1_i .. R^{ki}_i\}), i=1..N_a\}$
  - Scheduled after the synchronous traffic
  - **Non-real-time** traffic (IP…), scheduled after asynchronous one



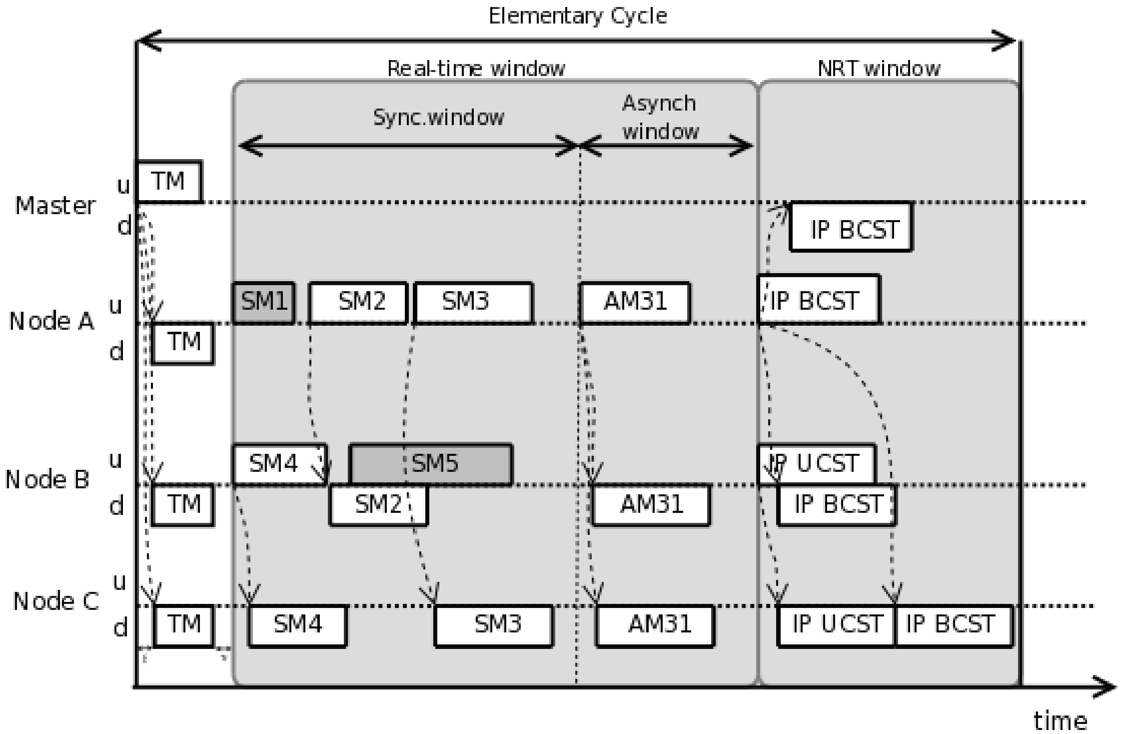


$$AM\_Rt_i = Lsig + Lsch_i + Lpoll$$

R. Marau, P. Pedreiras, L. Almeida. Asynchronous Traffic Signaling over Master-Slave Switched Ethernet protocols. RTN'07, 6th Workshop on Real-Time Networks, (satellite of ECRTS'07), Pisa, Italy, July 2007

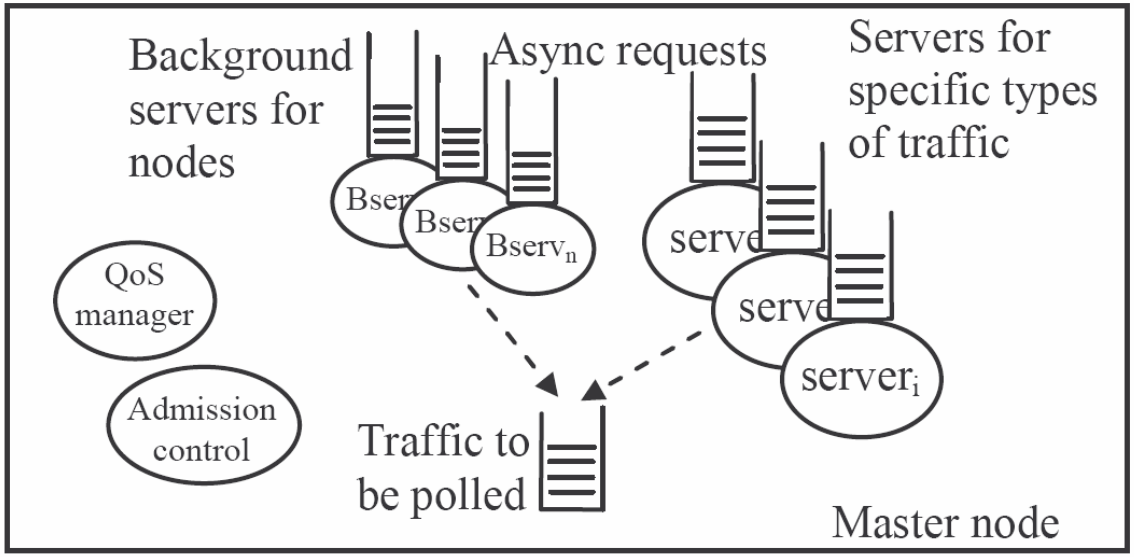
FTT-SE

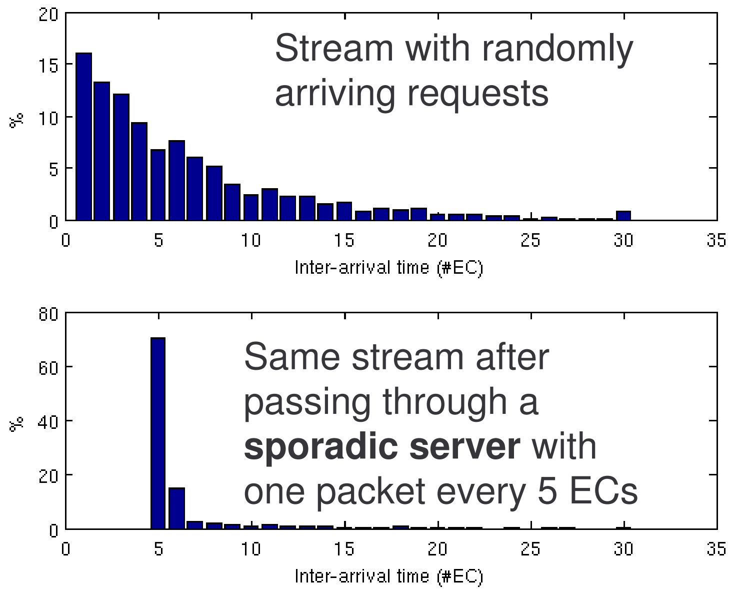
- Same triggering for all traffic
  - Aperiodic traffic is signaled to the Master
  - All traffic scheduled in an integrated way
    - Synchronous + asynchronous RT + Non-RT
    - **Everything encoded in the TM**



Server-SE (server-based FTT-SE)

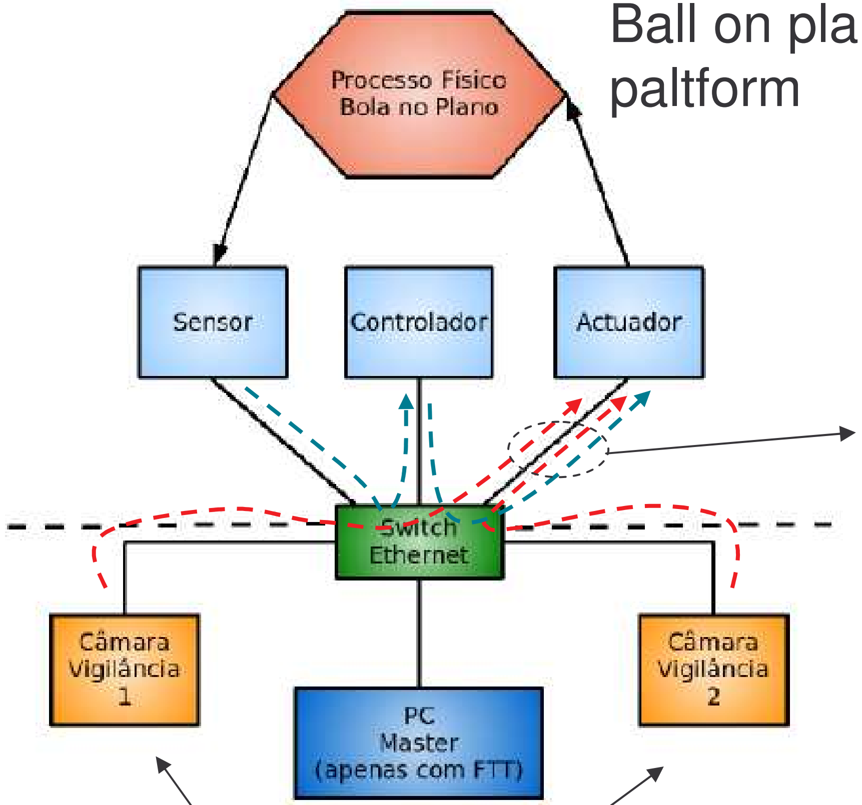
- All aperiodic
  - Uses aperiodic mechanism of FTT-SE
  - All traffic handled through servers
    - **Servers controls encoded in the TM**
  - Aims at managing servers dynamically





Ricardo Marau, Luıs Almeida, Paulo Pedreiras, Thomas Nolte. Towards Server-based Switched Ethernet for Real-Time Communications. Work-In-Progress Session, ECRTS 2008, Prague, Czech Republic, 2-4 July 2008.

Server-SE



Ball on plane paltform

Experiments with FTT-SE / Server-SE and plain SE

(see videos)

Total load varied between 18Mbit/s and 91Mbit/s

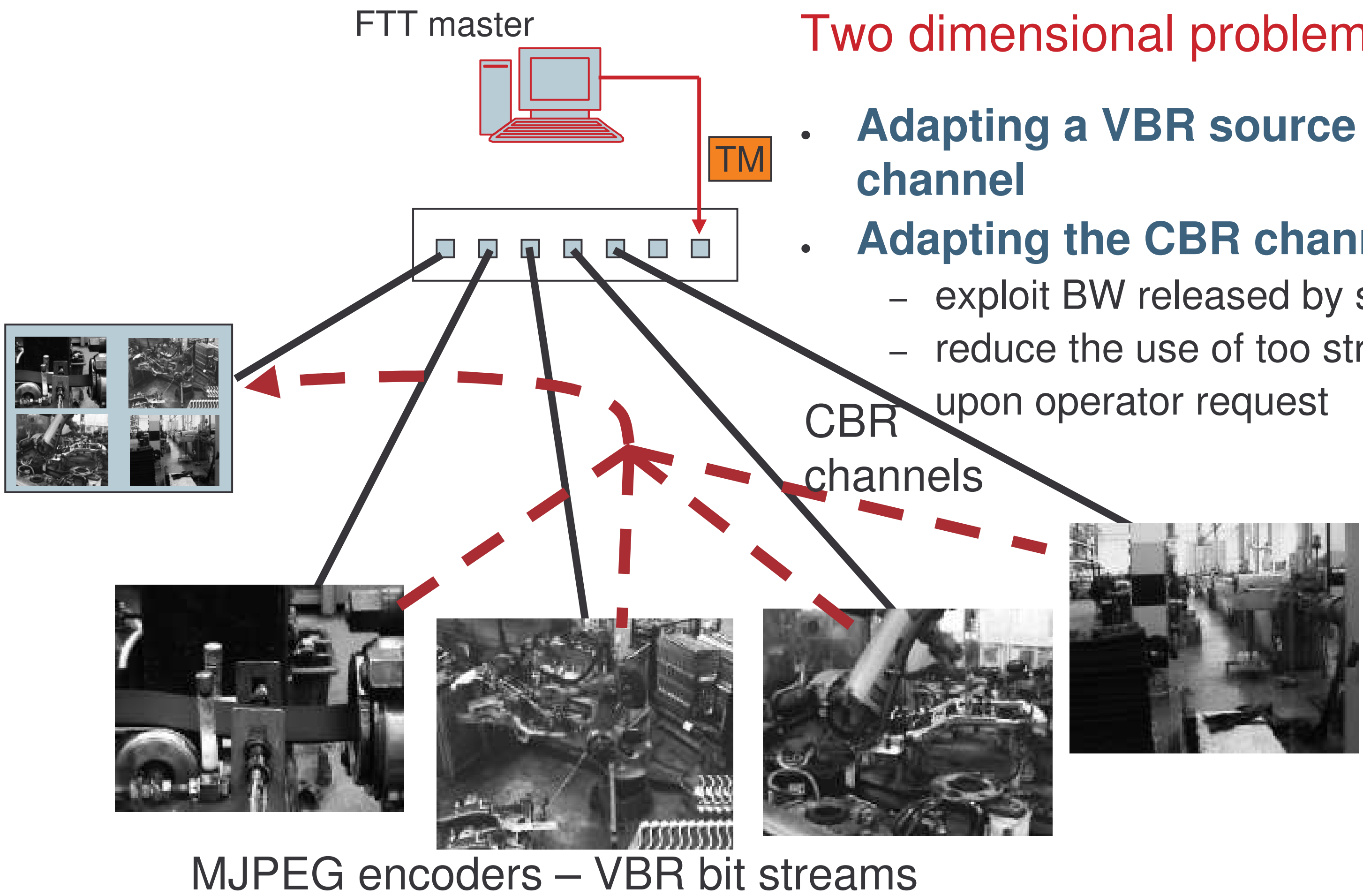
1 short control packet together with 2 long video frames

Without isolation, too high load causes control packet losses and large jitter

Surveillance cameras

Nuno Figueiredo. Controlo Distribuído de Plataformas para Experiências de Mecatrónica. Master Dissertation. University of Aveiro, July 2008. (in Portuguese)

QoS adaptation with FTT-SE



Two dimensional problem:

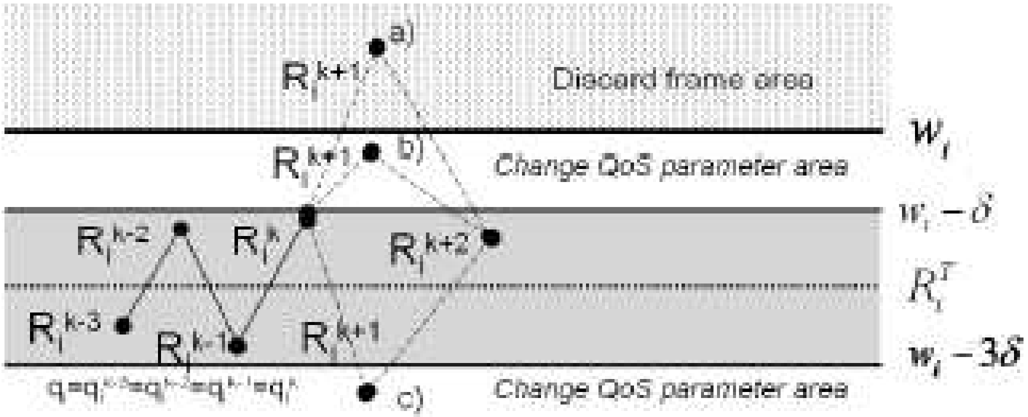
- **Adapting a VBR source to a CBR channel**
- **Adapting the CBR channels bandwidth**
  - exploit BW released by streams that are off
  - reduce the use of too strong compression upon operator request

J. Silvestre, R. Marau, P. Pedreiras , L. Almeida. On-line QoS Management for Multimedia Real-Time Transmission in Industrial Networks . IEEE Transactions on Industrial Electronics, 58(3), March 2011

A distributed monitoring system

- VBR → CBR adaptation
  - *q* is the compression parameter
  - It determines the size of each frame
  - Typical model of stream BW (*R*) and *q*

$$R(q) = \alpha + \frac{\beta}{q^{\lambda}}$$



a) Frame discarded. Reduce q, T or both

b) Frame above target coding bit rate, reduce q, T or both

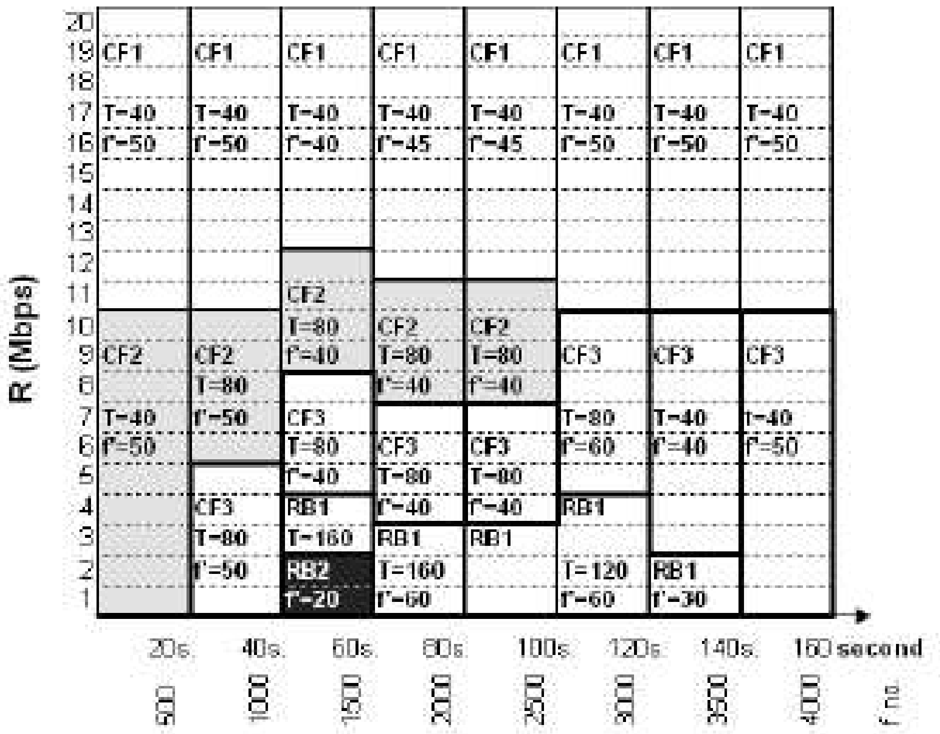
c) Frame below target coding bit rate, increase q, T or both

$$q_i^{k+1} = \left(\frac{R_i^{k+1} - \alpha_i}{\beta_i^{k+1}} \right)^{(1/-\lambda)}$$

$$\beta_i^{k+1} = \Delta R_i^{k+1,k} \left(q_i^k \right)^{\lambda} + \beta_i^k$$

A distributed monitoring system

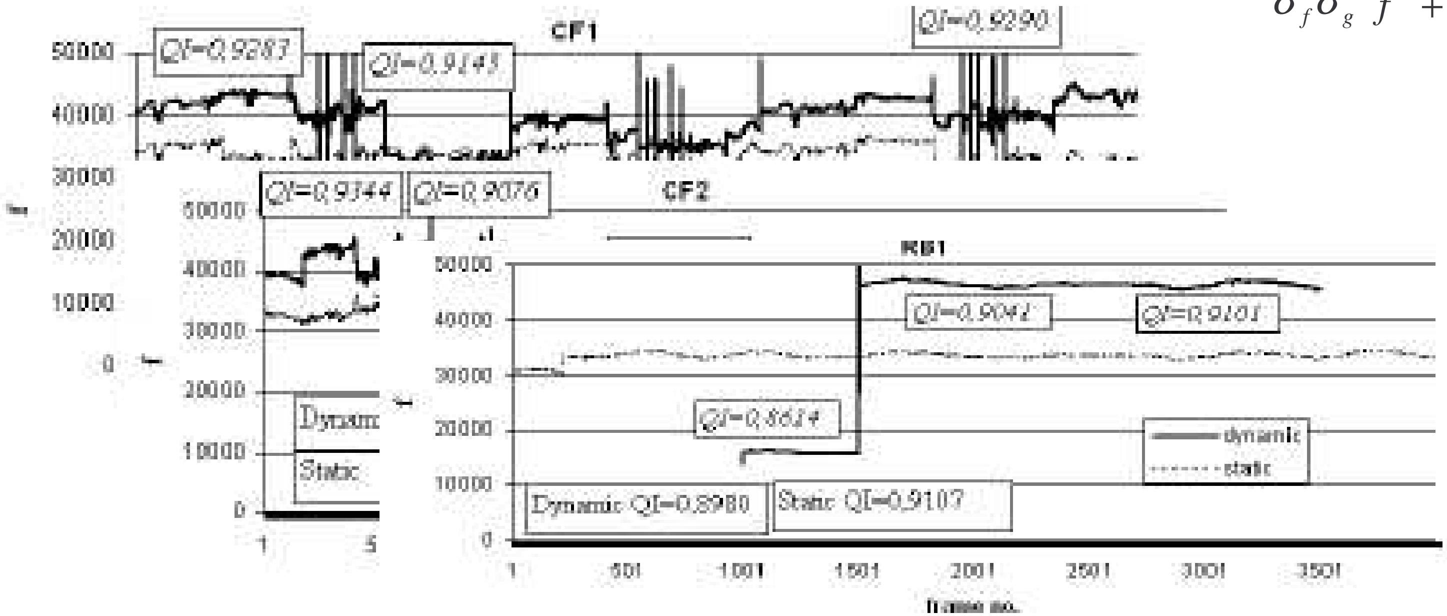
- Adapting multiple CBR channels
  - Streams are not always ON
  - Maximize total BW usage



A distributed monitoring system

- Adapting multiple CBR channels
  - Evolution of the Quality Index (QI) comparing to statically allocated channels

$$QI = \frac{\sigma_{fg}}{\sigma_f \sigma_g} \frac{2\hat{f}\hat{g}}{\hat{f}^2 + \hat{g}^2} \frac{2\sigma_f \sigma_g}{\sigma_f^2 \sigma_g^2}$$



A distributed monitoring system

5 Mbit/s **1 Mbit/s**

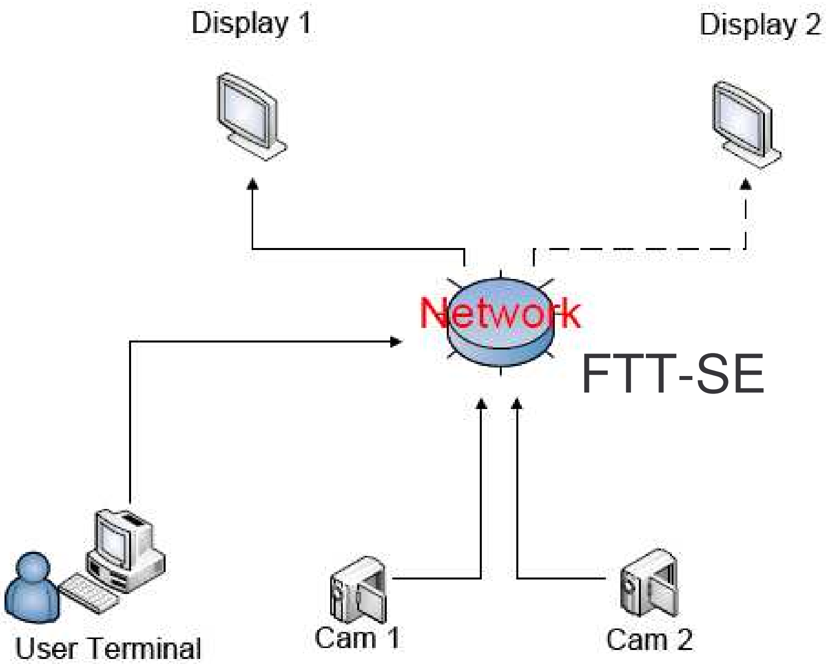
V **after 20s** **V**

1 Mbit/s 6Mbit/s at all times **5 Mbit/s**

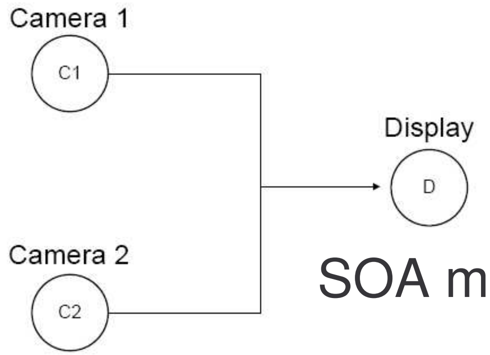
The iLAND project: dynamic reconfiguration

http://www.iland-artemis.org/

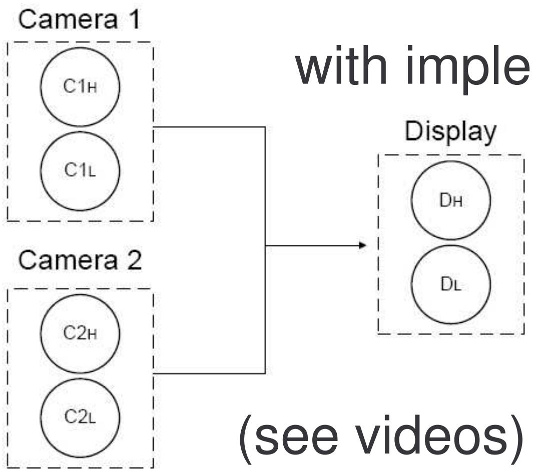
- **Service-oriented real-time** middleware for deterministic and dynamically **reconfigurable applications**



Initial prototype



SOA modeling



with implementations

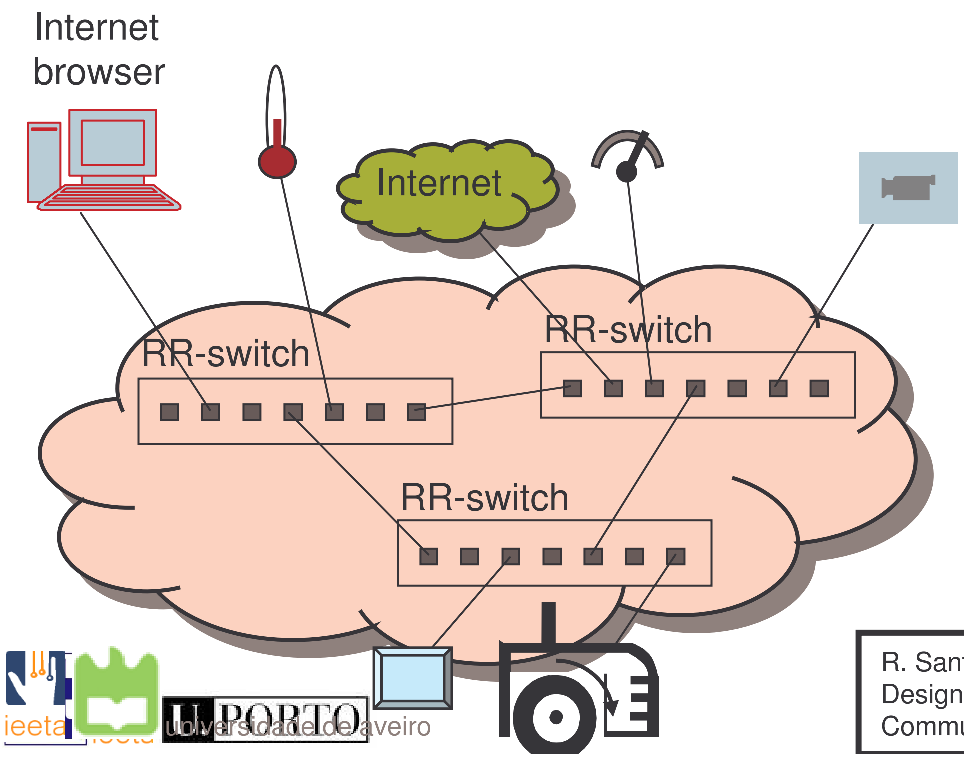
(see videos)

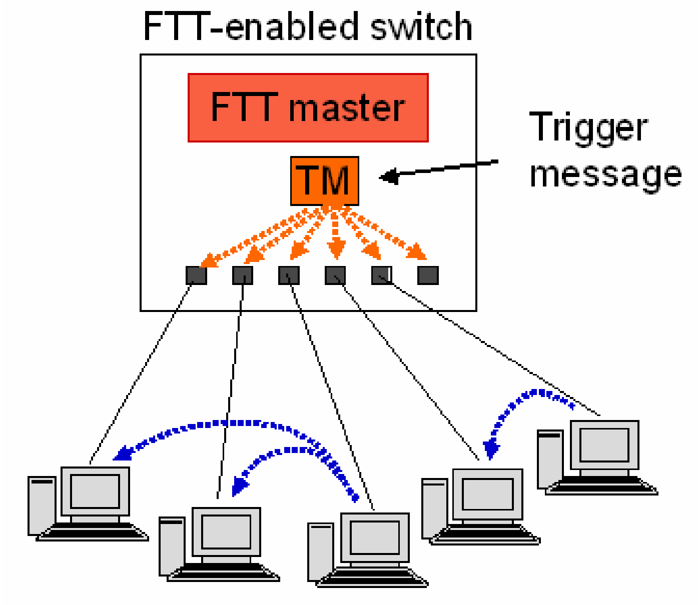
FTT-enabled switch

HaRTES
Hard Real-Time Ethernet Switching
http://www.ieeta.pt/lse/hartes

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**)

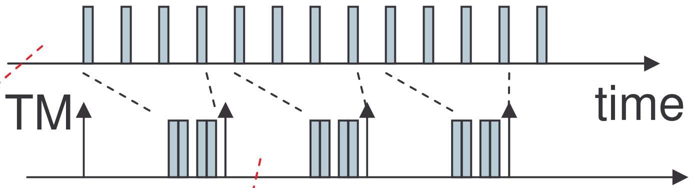


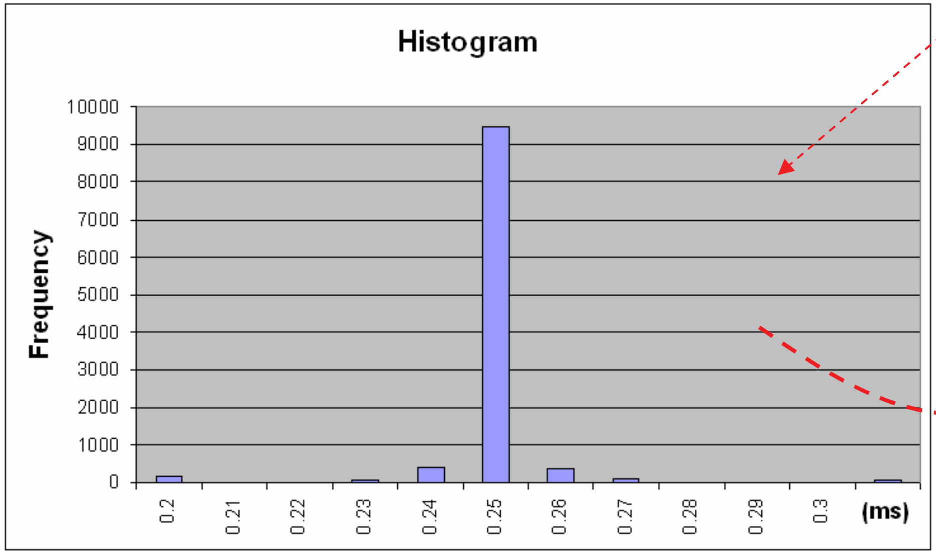


R. Santos, R. Marau, A. Oliveira, P. Pedreiras, L. Almeida. Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication. WFCS 2008. Dresden, Germany. 21-23 May 2008.

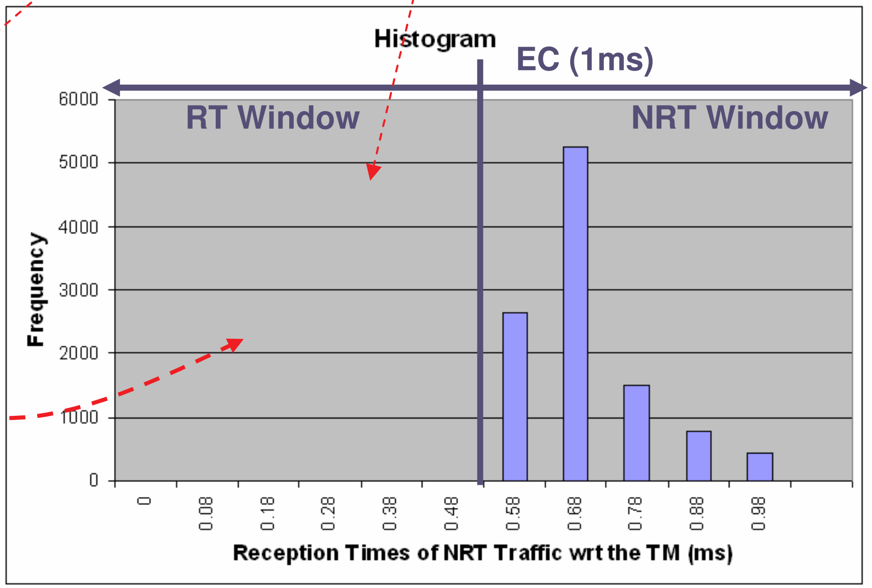
FTT-enabled switch

- Aperiodic traffic confinement





Submitted traffic
1000B packets, T_{avg}=250µs



Outgoing traffic
Offset wrt the previous TM
NRT window - 50% EC

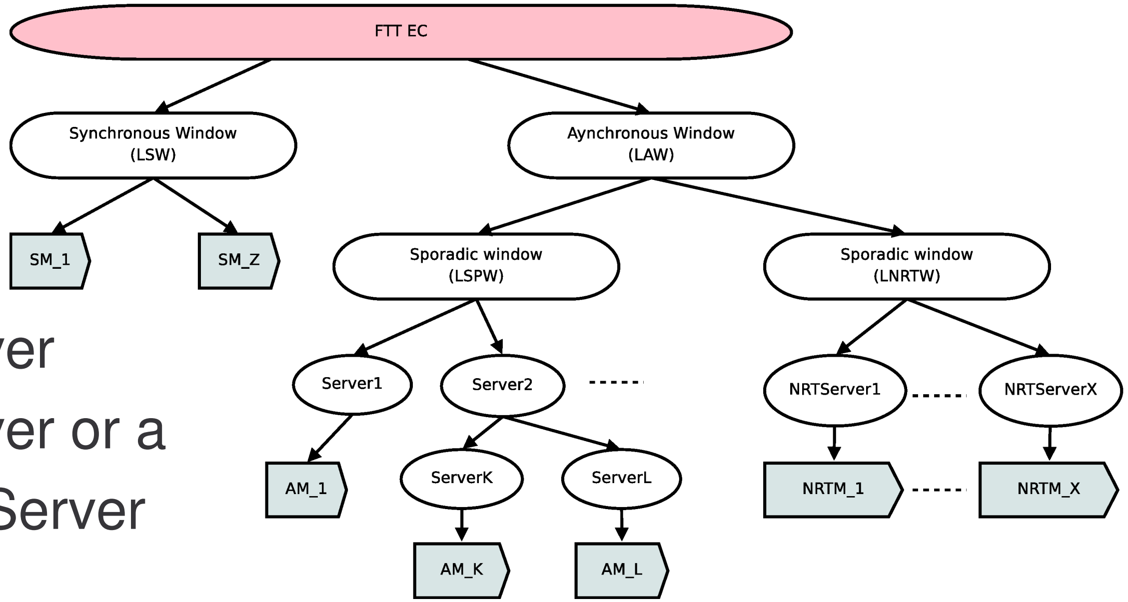
Regularity of the TM	
	T_TM_{avg} = 1,000ms
	T_TM_{max} = 1,0003ms
	T_TM_{min} = 0,99998ms
	STD_TM = 138ns

FTT-enabled switch

- Traffic scheduling and management
  - Supports online admission control and dynamic QoS management
  - Allows arbitrary traffic scheduling policies
  - Reduction in the switching latency jitter
- Traffic classification, confinement and policing
  - Seamless integration of standard non-FTT-compliant nodes without jeopardizing the real-time services
  - Asynchronous traffic is autonomously triggered by the nodes
  - Unauthorized transmissions can be readily blocked at the switch input ports, thus not interfering with the rest of the system

Hard Real-Time Ethernet Switching

Hierarchical traffic scheduling



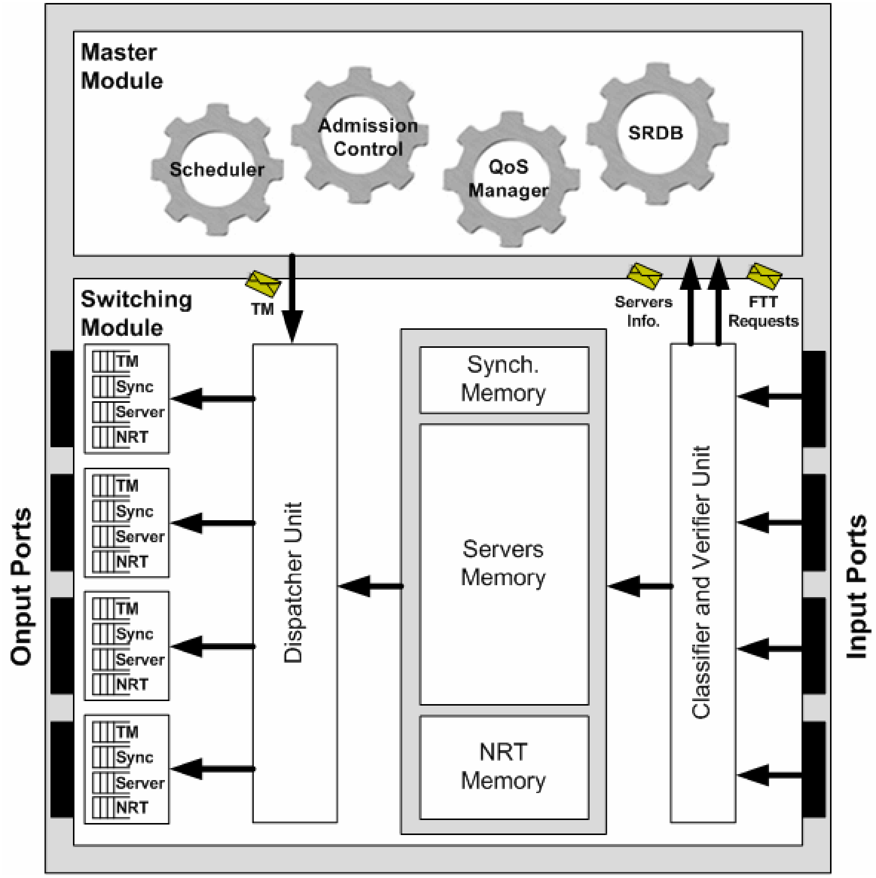
- First Level
  - SW – Polling Server
  - AW – Polling Server or a Deferrable Server
- Second Level
  - Manages the sporadic and the NRT traffic
- Third Level
  - Implements specific servers, constituing virtual channels

R. Santos, A. Vieira, P. Pedreiras , A. Oliveira, L. Almeida , R. Marau, T. Nolte.
Flexible, Efficient and Robust Real-Time Communication with Server-based Ethernet Switching.
WFCS 2010. Nancy, France. 18-21 May 2010.

HaRTES
Hard Real-Time Ethernet Switching

Proposed functional architecture

- **Master Module (MM)**
  - Implemented in Software
- **Switching Module (SM)**
  - Implemented in Hardware

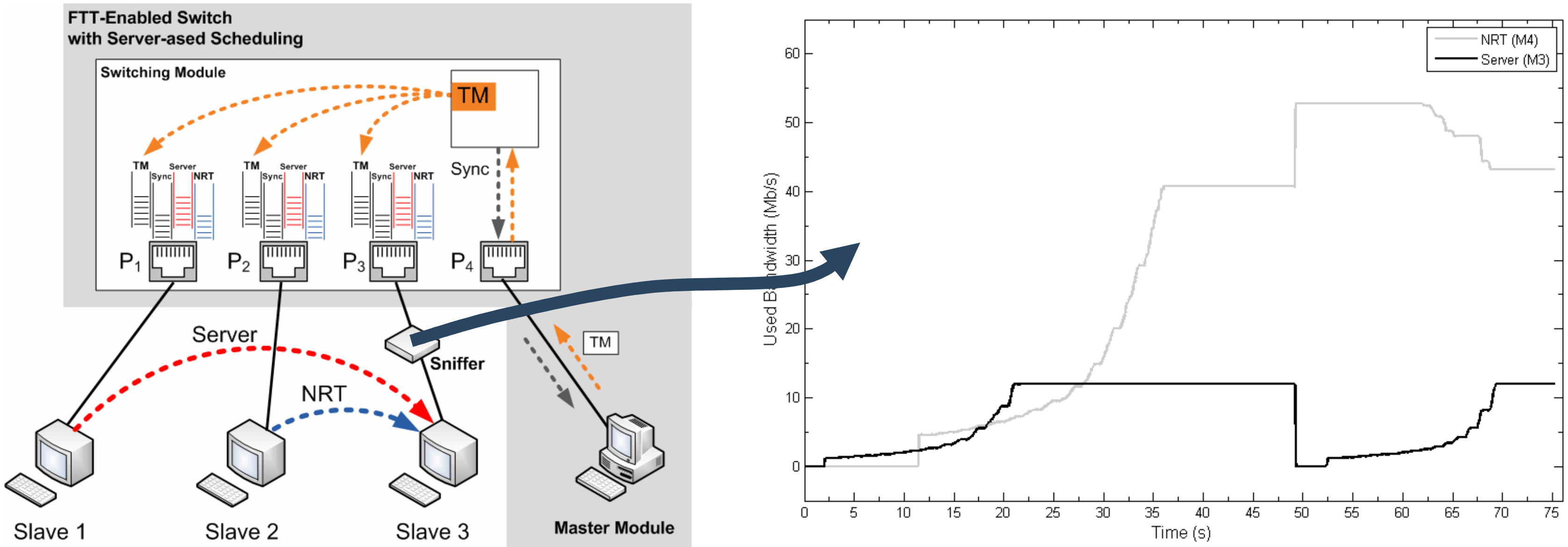


- SM informs at the beginning of each EC the MM which server messages have been received in previous EC.
- MM computes the scheduling and communicates it back to the SM
- The SM enforces the respective transmissions

HaRTES Hard Real-Time Ethernet Switching

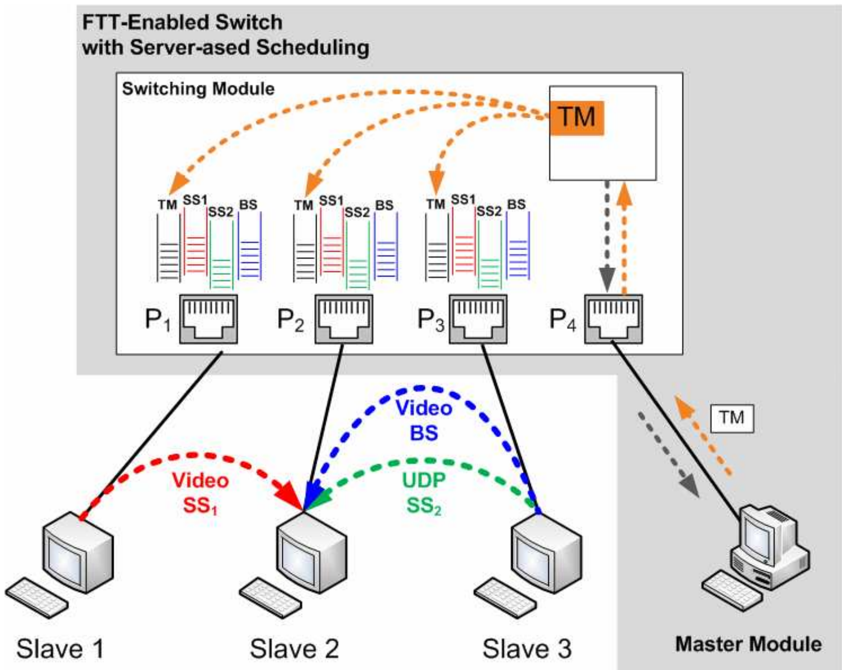
Experimental results

- Elementary Cycle = 1ms; Asynchronous Window = 54%
- Implemented a Sporadic Server with C = 3000B and T = 2 EC
- Slave1 sends 150B size RT messages handled by the sporadic server.
- Slave 2 sends 600B size NRT messages

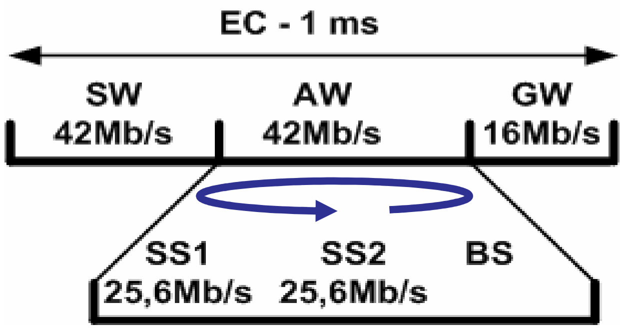


Experimental results

- Elemantary Cycle = 1ms; Asynchronous Window = 42%
- SS1, SS2 – sporadic server with C = 3200B and T= 1ms
- BS – backgound server uses the remaining bandwidth

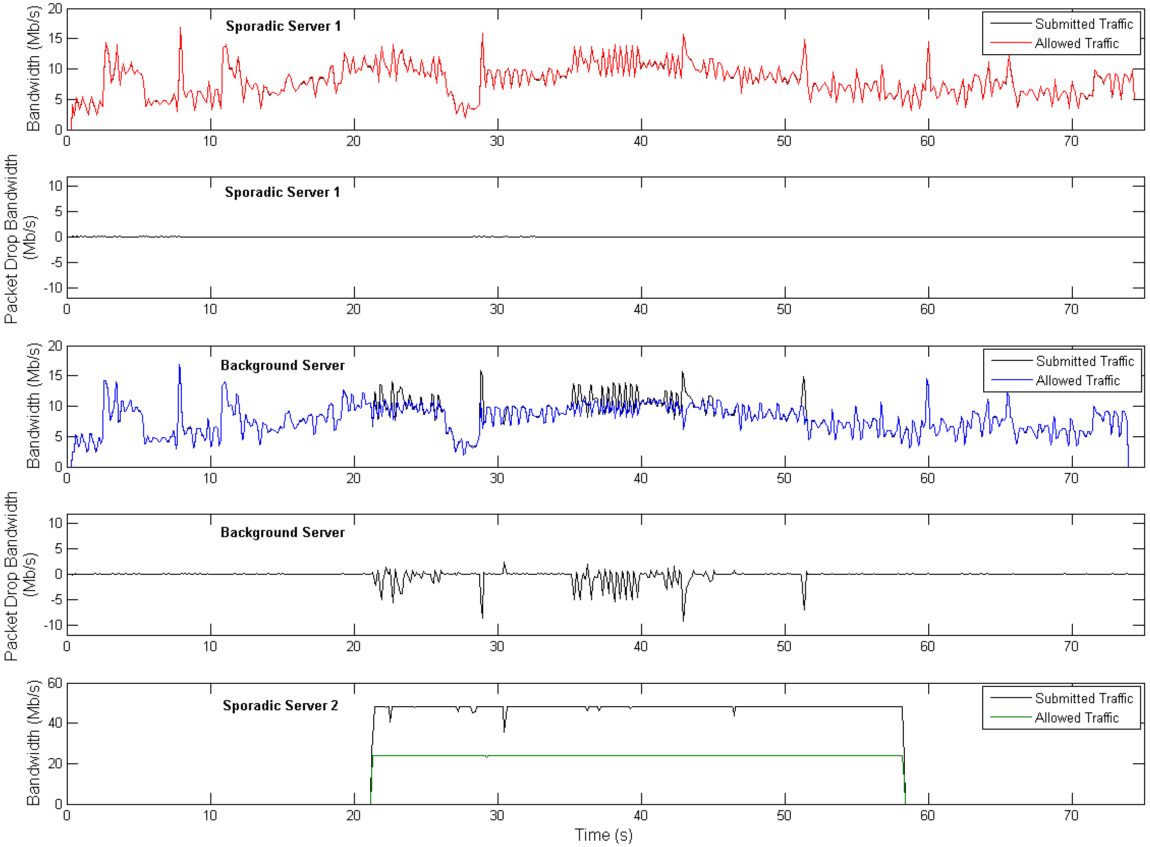


- Video SS1 equal to Video BS
  - Peak load = 21.9 Mbps
- UDP SS2
  - Average load = 48.9Mbps



Hard Real-Time Ethernet Switching

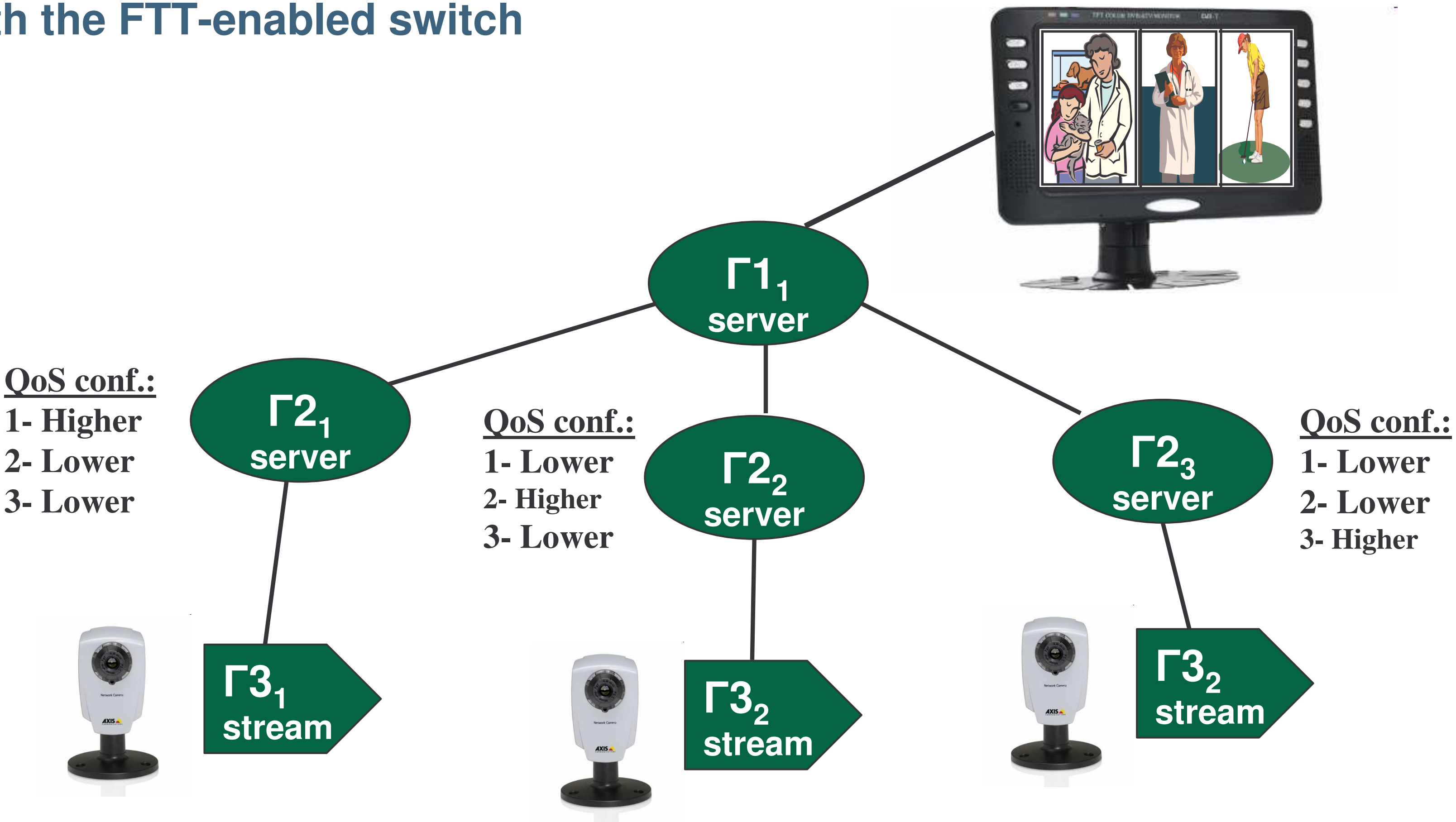
Experimental results



Video

Adaptive surveillance system

- QoS adaptation **of communication channels with the FTT-enabled switch**



See video

Wrapping up – Global conclusion

Conclusion

- The **network is a fundamental component** within a distributed or networked system (supports **system integration**)
- **Real-time coordination** in a distributed / networked system requires **time-bounded communication**
  - appropriate protocols must be used
- We have seen a brief overview of the **techniques and technologies** used in the **networks and middlewares for embedded systems**
- Still many **open issues** remain in trying to **improve the timeliness**, **robustness** and **efficiency** of the communication

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.**