ARTIST2 Summer School 2008 in Europe
*Autrans (near Grenoble), France*
*September 8-12, 2008*

# Challenges of *Flexible* Real-Time Communication

Lecturer: **Luis Almeida**
Embedded Systems Lab
IEETA – University of Aveiro, Portugal

# Flexibility ?

- ***Flexibility*** – the capability of changing form
- "a ready capability to adapt to new, different, or changing requirements" (Merriam-Webster on-line)
- Within embedded systems, ***flexibility*** is a vast property that can apply to **off-line** or **on-line** features
  - a design flow that supports changes with minor redesign effort
  - systems prepared for different configurations at/after deployment
  - hardware that supports multi-functionality
  - systems that adapt to varying run-time conditions (load, users, resources...)
  - systems that reconfigure upon hazardous events or occasional operation of certain components
  - ...

# Flexibility – increasing interest

- Appears frequently as *Adaptability* and *Reconfigurability*
- Topic of an activity within ArtistDesign (NoE in FP7)
  - Other European projects (DySCAS,FRESCOR,RUNES,HiPEAC...)
- Several related events
  - NeRES 2007, APRES 2008, ARC 2005-8, ...
- EU FP7 Embedded Systems Call
  - "The engineering of more robust, context-aware and easy-to-use ICT systems that **self improve and self-adapt within their respective environments**."
  - "Heterogeneity; composability; predictability of extra-functional properties; **adaptivity for coping with uncertainty**; and unification of approaches from computer science, electronic engineering and control."

# In this talk

- Focus on **Operational Flexibility** (on-line) in
  - Distributed (networked) Embedded Systems
- What we want
- Two possible approaches
- Some challenges
- Our recent research inline with these approaches
- Conclusion & general issues

# What we want

To be able to **connect any component** to the system, **on-line**, being sure that:

- **Nothing bad** will happen
- The system will **do its best to integrate** the new component:
  - It can **accept** the new component without any adjustment
  - It can **accept** it upon system **adjustment**
  - It can **reject** the new component

Reconfigurability

# What we want

Allow the system to **adjust on-line** according to **effective instantaneous needs** or **resource availability**:

- **Free and reuse the resources** of subsystems that **operate occasionally** or **fail** when off
- **Adapt resources required** by each subsystem on-line to:
  - **Minimize the resources** used (e.g. BW, energy, …)
  - **Maximize the service** delivered with a fixed level of resources
  - Cope with **varying levels of resource availability** (e.g. links with variable BW, decreasing available energy, …)

Adaptivity

# What we want

Focus on **Distributed Embedded Systems**:

- ✓ Work on **Flexible scheduling**, **Feedback scheduling**, **Elastic scheduling**, **Resource usage optimization** in control systems, ..., has mainly addressed uniprocessors
- ✓ There have been few **extensions to distributed** embedded systems
  - ✓ In these, **the network is central** and **must support** the desired level of **flexibility**

# Two approaches

**to achieve flexibility in distributed (networked) embedded systems**

- **Law enforcement**
  - Negotiate resource needs and enforce their proper use
  - Requires good level of control over resources
    - Admission controller, QoS manager, ...

Hard real-time

- **Voluntary cooperation**
  - Assume nodes will cooperate for proper resource usage
  - No enforcement → unavoidable limitations caused by non-cooperating nodes, interference, ...

Best effort

# Some challenges

- **Law enforcement**
  - **How strong/robust** is the **enforcing** of proper resource usage?
  - **Control over resources** and **flexibility management** imply **extra resource needs** (BW, CPU, energy ...) !
  - Also imply **extra complexity**! With potential for **lower reliability**!
  - **Extra state information** is needed to manage flexibility.
    - Is it **replicated**? If so, it must be consistent...
    - Is there a **safe state** in case of inconsistency?
- **Voluntary cooperation**
  - Some kind of **guarantees (probabilistic)** would be welcome!
    - Characterization of **operational environments**
    - Varying **communication links**
  - **Resource usage** optimizations

# Some challenges

- **Flexibility management**
  - How to **distribute free BW** among a set of users?
    - Elastic models
    - (m,k)-firm model
    - Greedy models ...
    - Act on C, on T, on both...
  - **When and how to trigger** adaptation / reconfigurations?
  - **Flexible mode changes**...
  - How to **relate resource usage and QoS**? Or even better, **QoE**?

# Our recent research

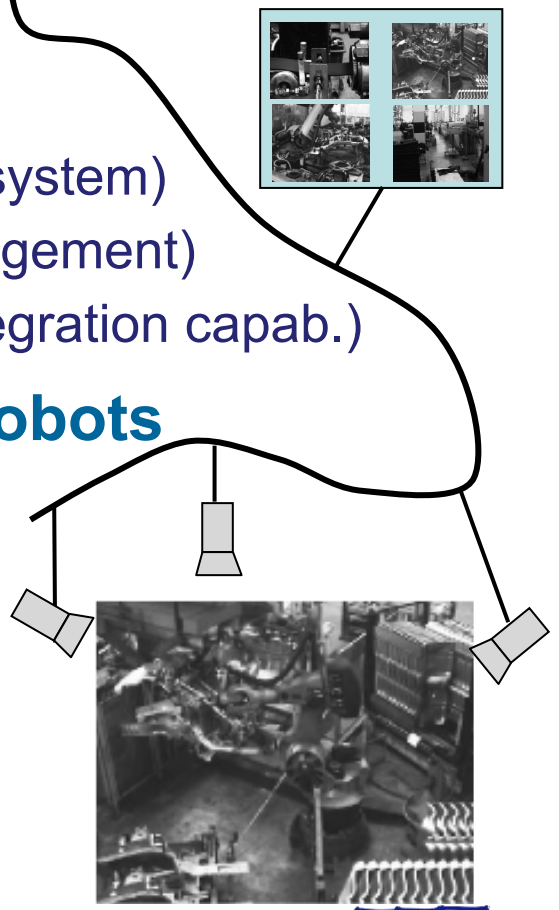
F-T-T-

Flexible Time-Triggered architecture

- **FTT framework – new developments**
  - FTT-SE (enhanced aperiodic communication)
  - Dynamic QoS management (distributed video system)
  - Server-SE (server-based communication management)
  - Enhanced switch (boosting robustness and integration capab.)
- **Wireless communication for teams of robots**
  - Adaptive-TDMA framework
  - Round reconfiguration

Law enforcement

Voluntary cooperation

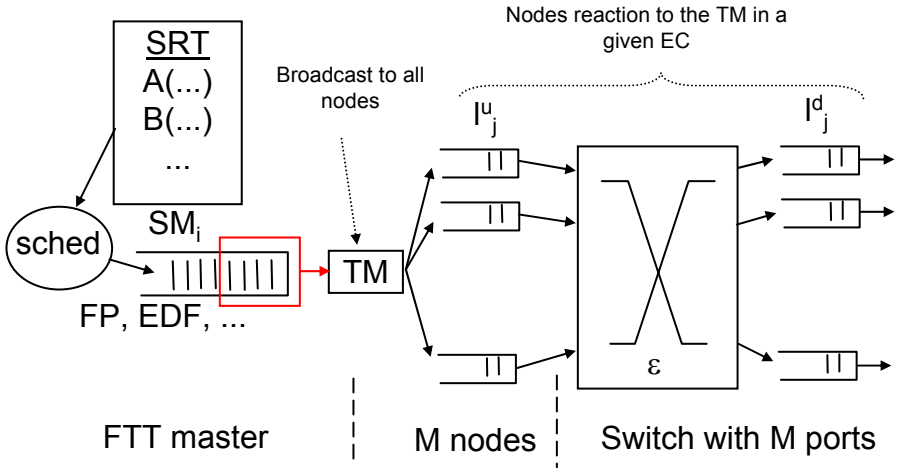



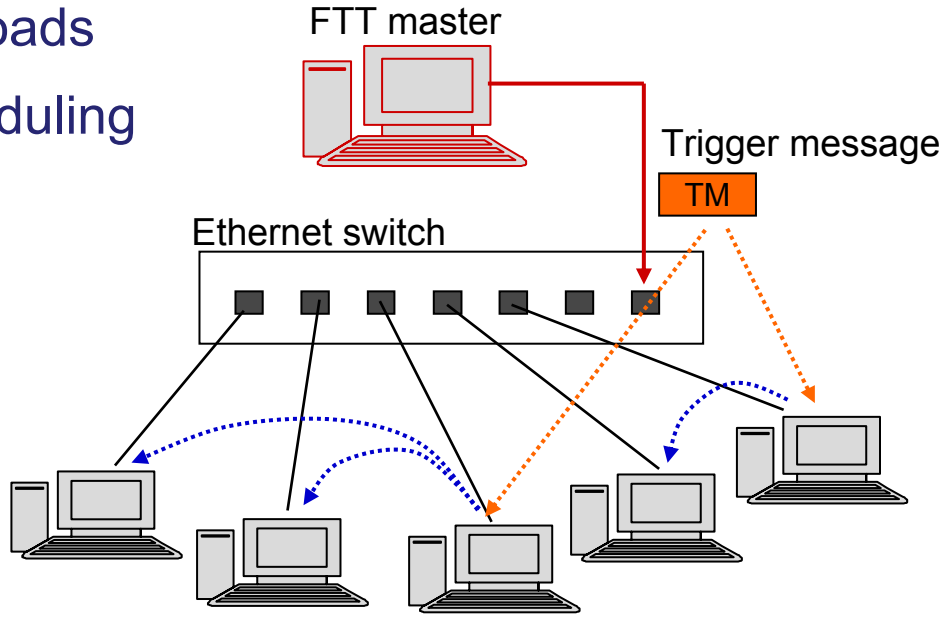
# FTT-SE

## Flexible Time-Triggered communication over 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

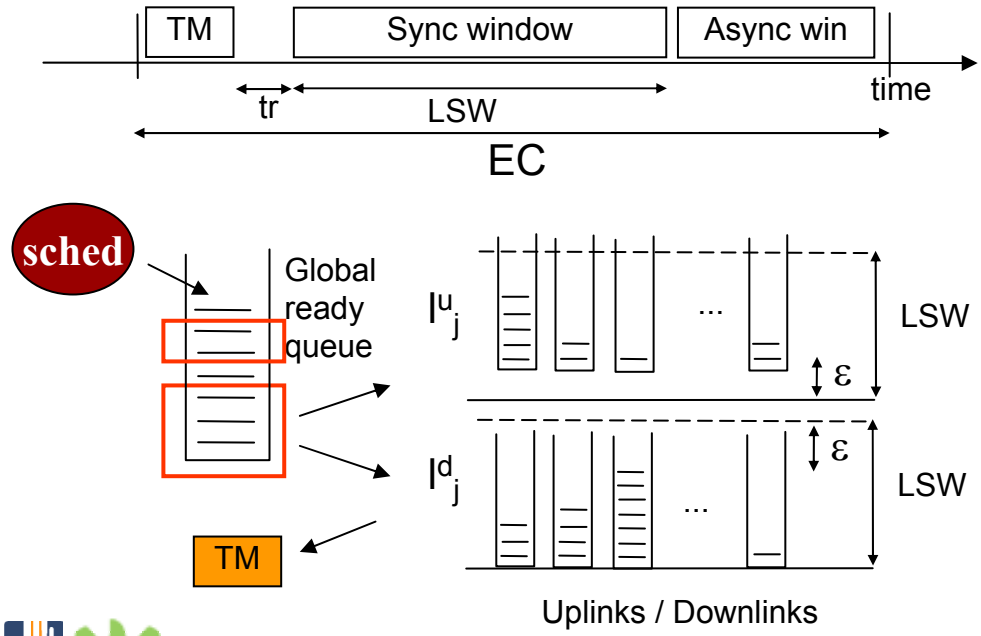


# FTT-SE

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

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




- **Scheduling equation**

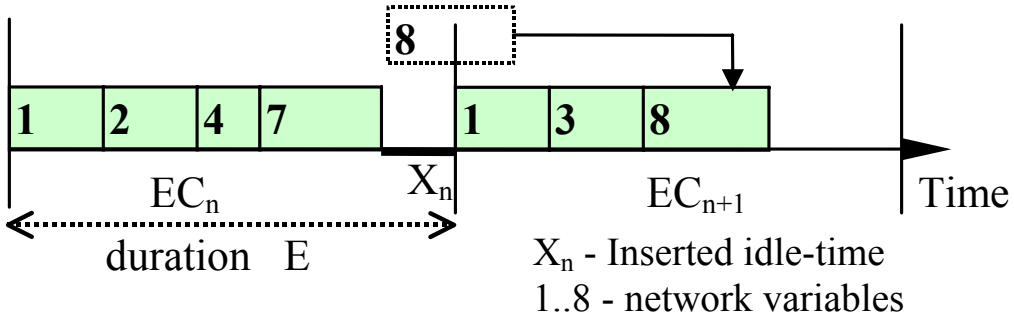
$$\begin{cases} \max\limits_{j} \left( \sum\limits_{SM_i \in l^u_j} C_i \right) \le LSW - \max\limits_{SM_i \in l^u_j} \varepsilon_i \\ \max\limits_{j} \left( \max\limits_{SM_i \in l^d_j} (f_i) \right) \le LSW \end{cases}$$

- **Memory bounds**

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

# 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')



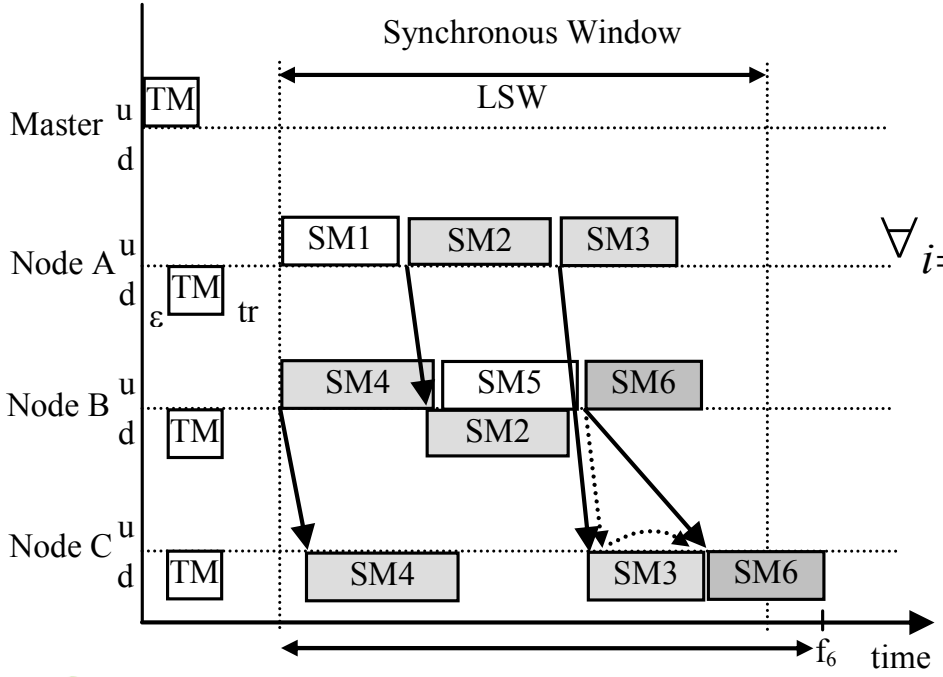

$$C_i' = C_i * \frac{E}{E - X_{max}}$$

Inserted idle-time compensation factor

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

# FTT-SE

- **Testing schedulability of periodic traffic**
  - Interference in the uplinks appears at the 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:**

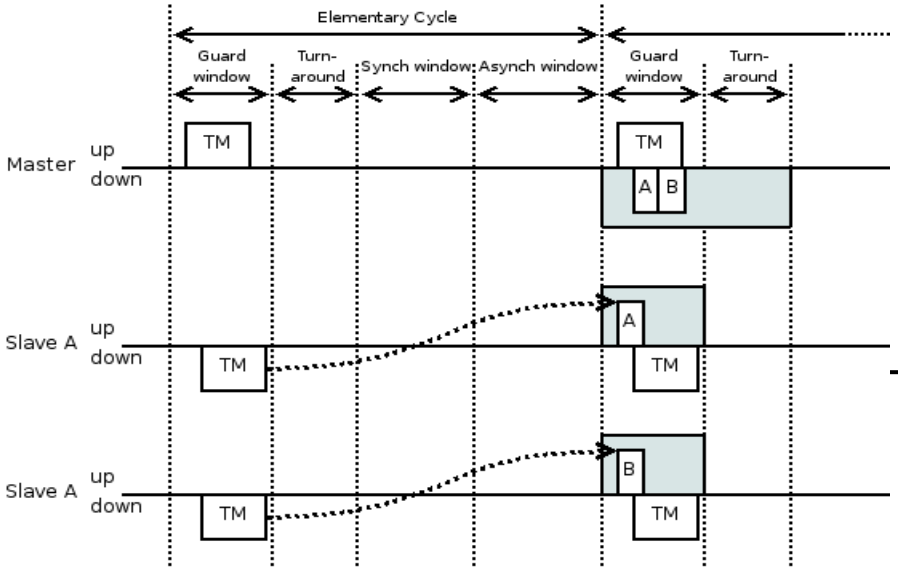
$$\forall_{i=1..n} \sum_{j=1}^{i} \frac{C_j}{T_j} + \frac{\max_{j=1..i} J_j}{T_i} \leq U^{\text{lub}}_{RM,EDF}(i)$$

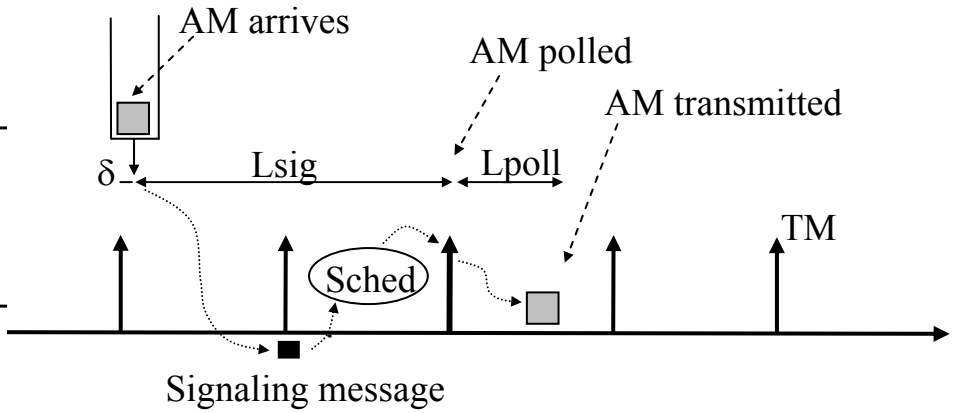
$$\sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{\max_{i=1..n} J_i}{T_1} \leq U^{\text{lub}}_{RM,EDF}(n)$$

# 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 the async one

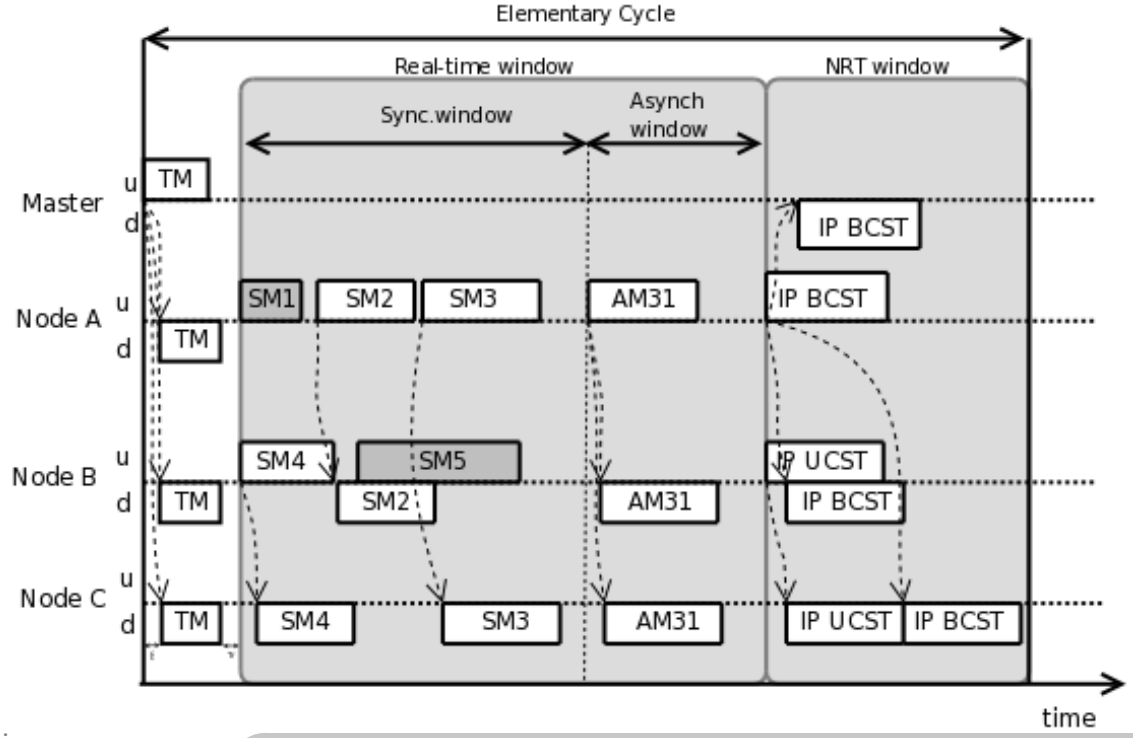



$$AM\_Rt_i = Lsig + Lsch_i + Lpoll$$

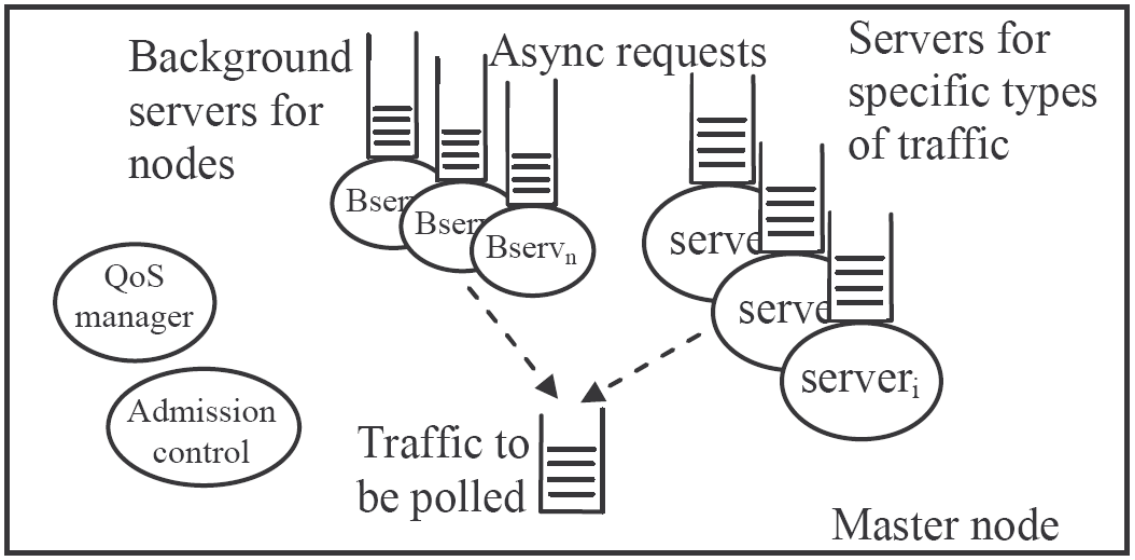
# FTT-SE

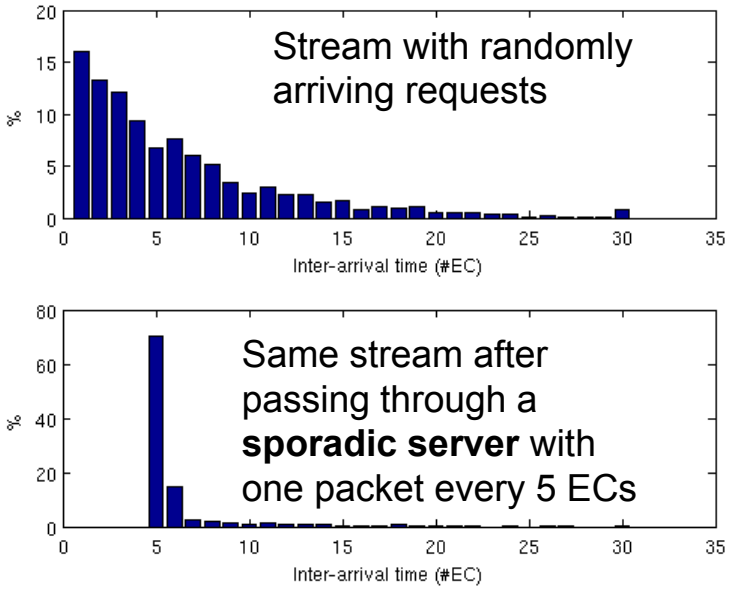
- **Same triggering for all traffic**
  - Aperiodic traffic is signaled to the Master
  - All traffic scheduled in an integrated way
    - Synchronous + asynchronous RT + Non-RT
    - **Everything encoded in the TM**



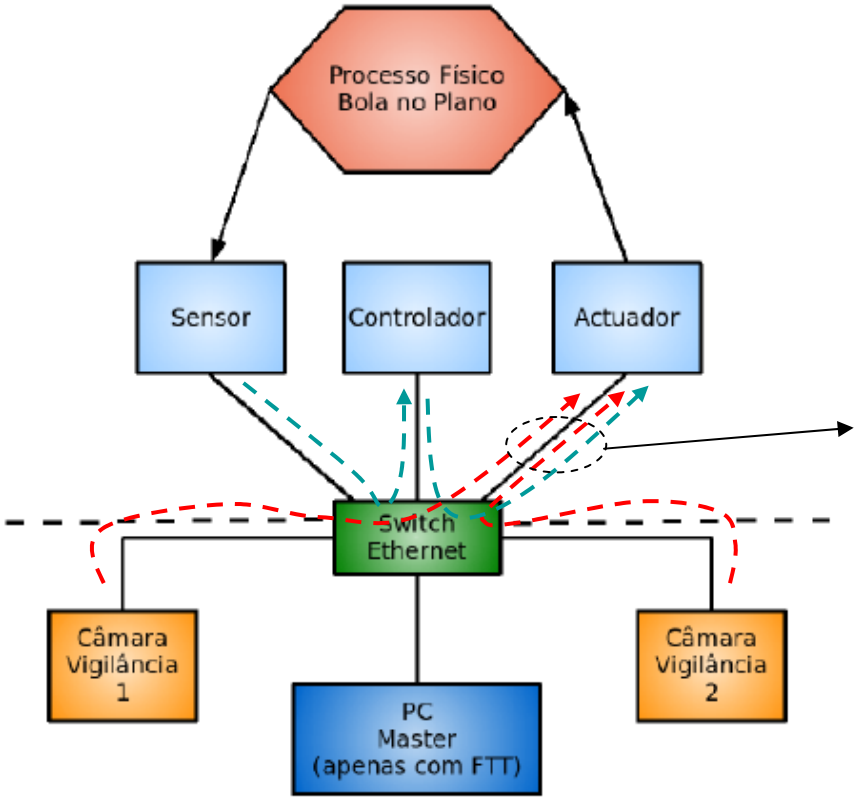
# Server-SE

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



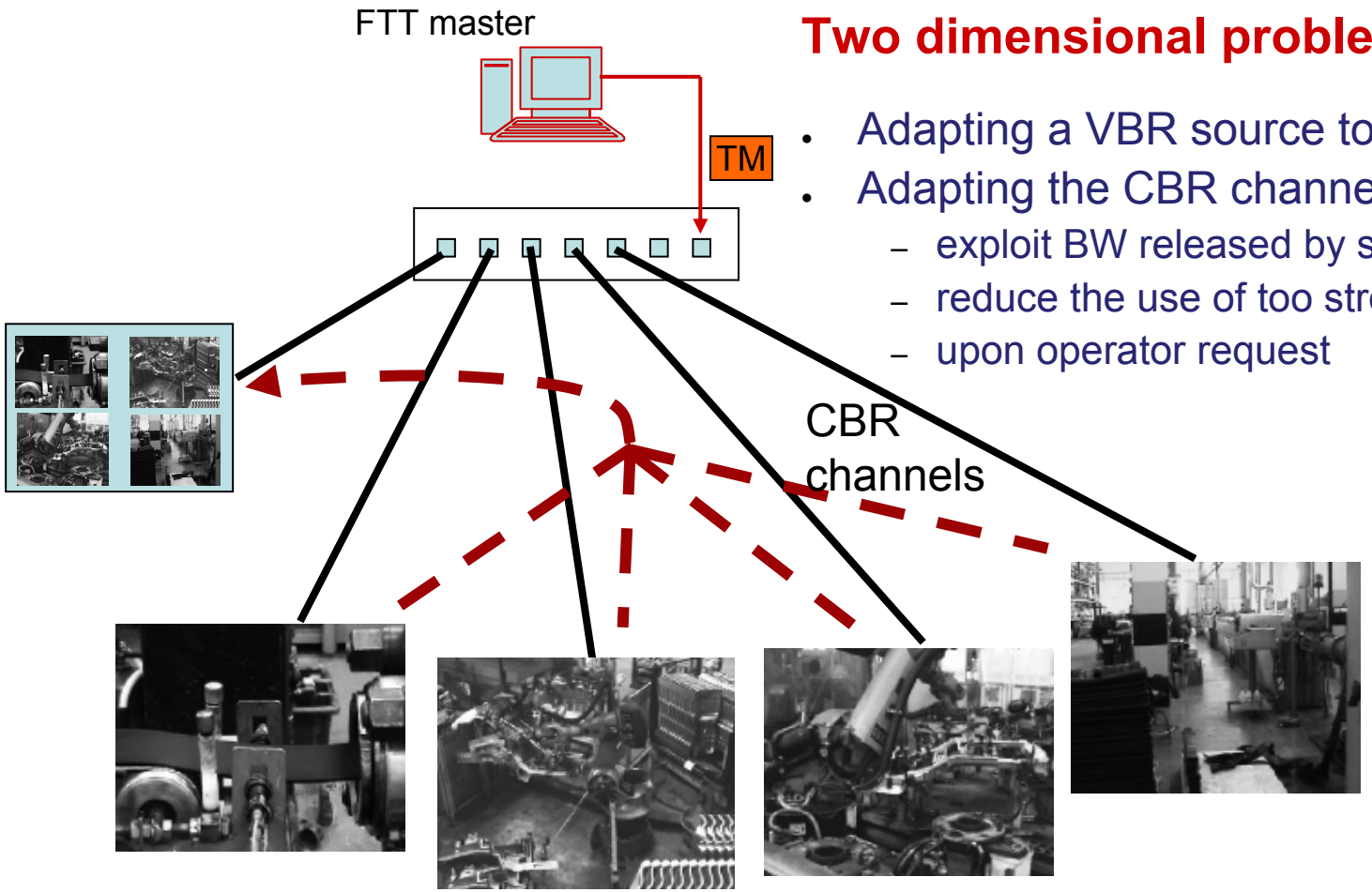
# Server-SE




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

1 short control packet together with 2 long video frames

# QoS adaptation with FTT-SE



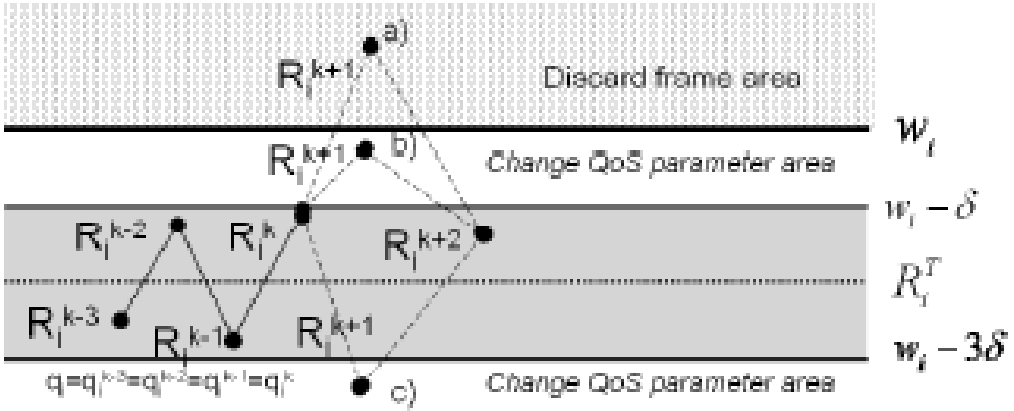

**Two dimensional problem:**

- 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

MJPEG encoders – VBR bit streams

# 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*




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

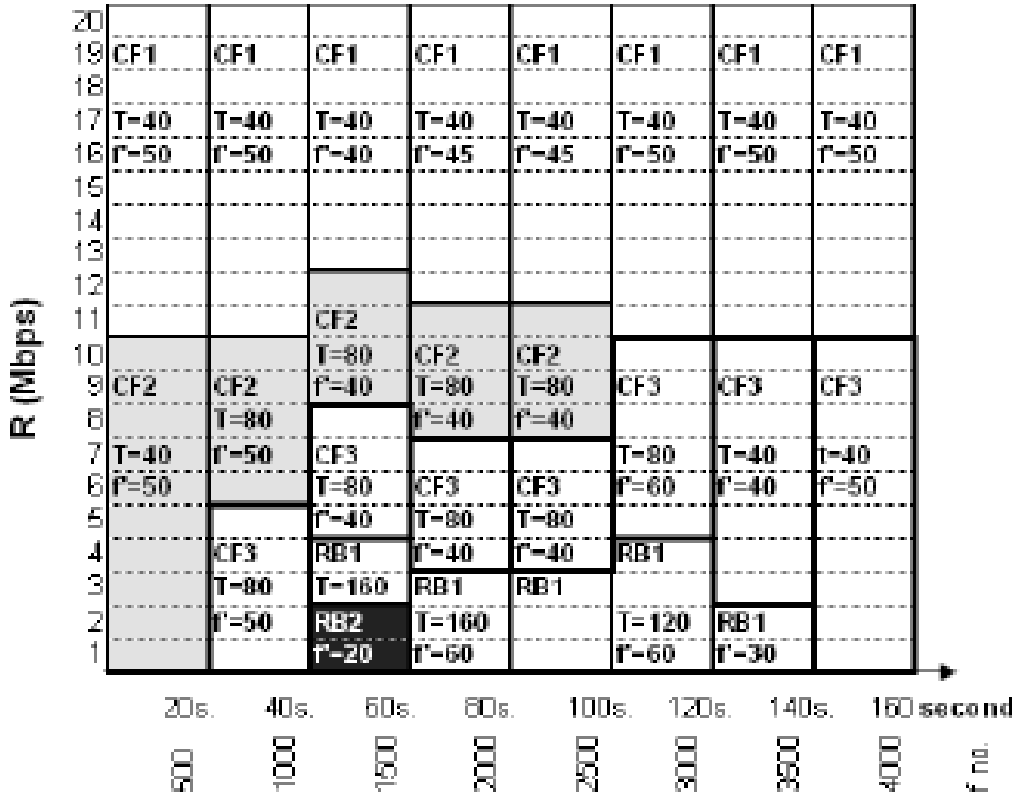
$$R(q) = \alpha + \frac{\beta}{q^{\lambda}}$$

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

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

# A distributed monitoring system

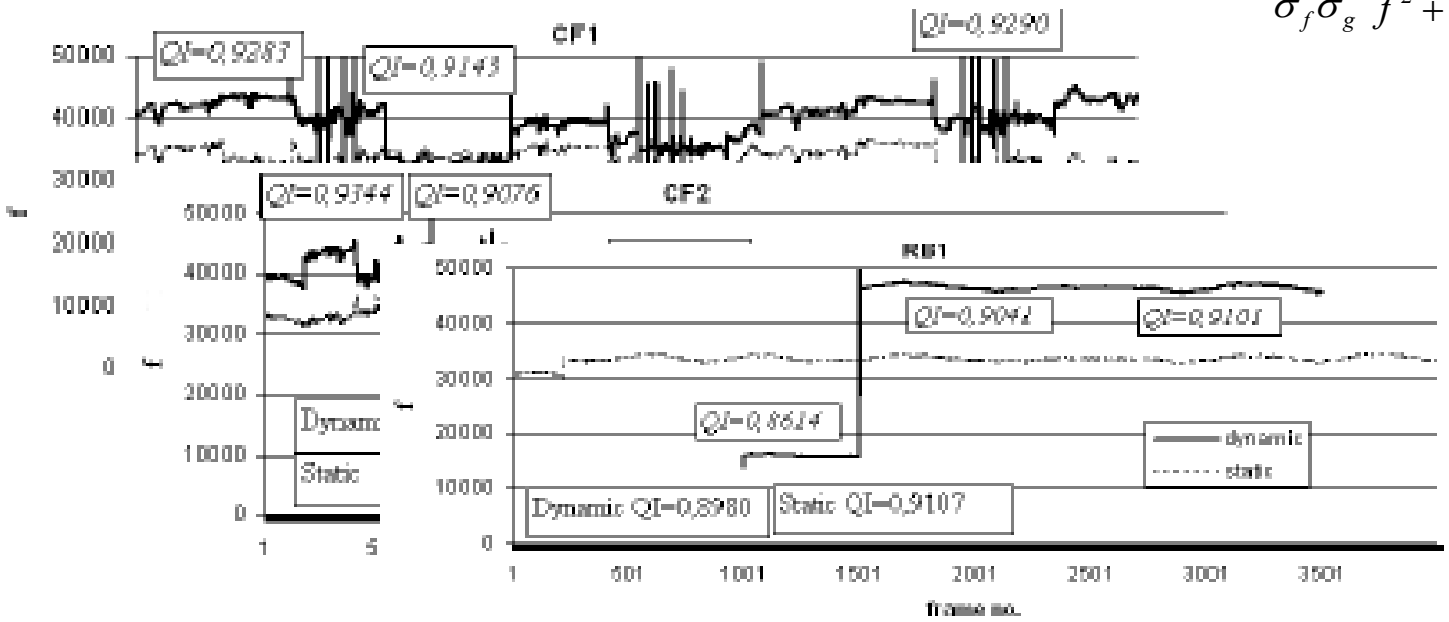
- **Adapting multiple CBR channels**
  - Streams are not always ON
  - Maximize total BW usage



# A distributed monitoring system

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

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



5 Mbit/s

V

1 Mbit/s

after 20s

1 Mbit/s

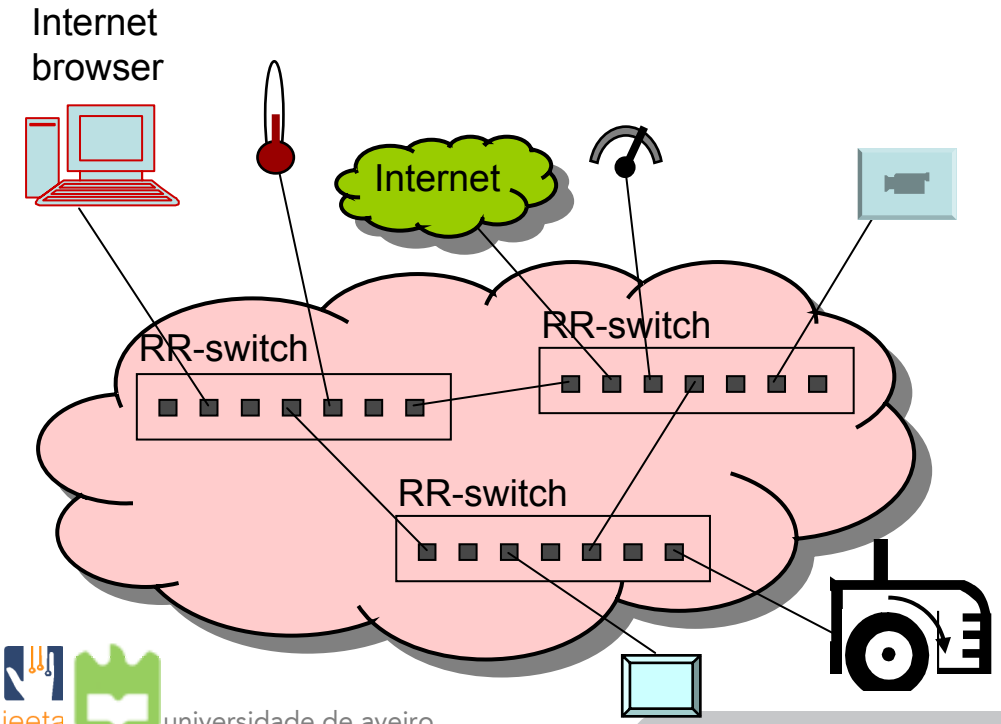
V

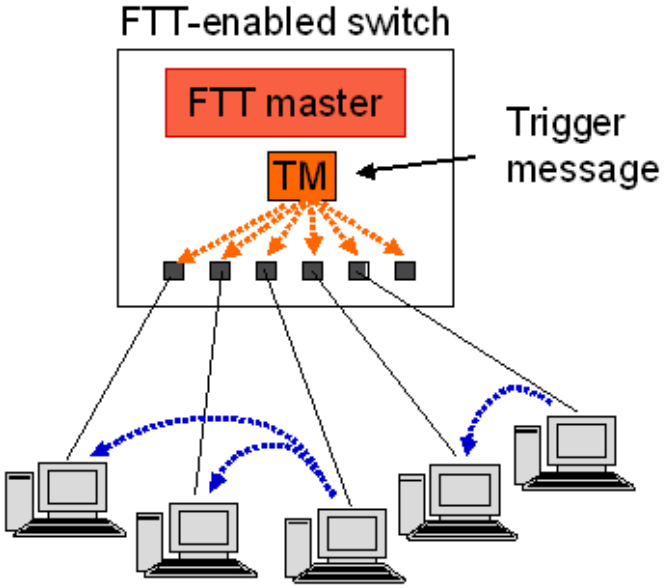
5 Mbit/s

# Resource-reservation switch

Providing **timeliness, flexibility and high robustness** in switched Ethernet networks

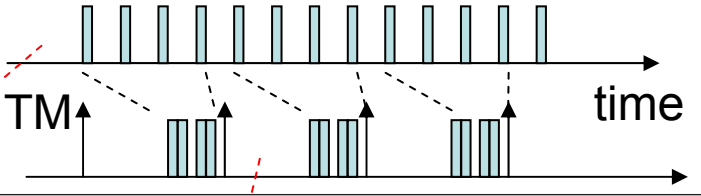
- ✓ Enforce negotiated channel characteristics (**policing**)
- ✓ Reject abusive negotiated traffic (**filtering**)
- ✓ Confine non-negotiated traffic to separate windows (**selection**)

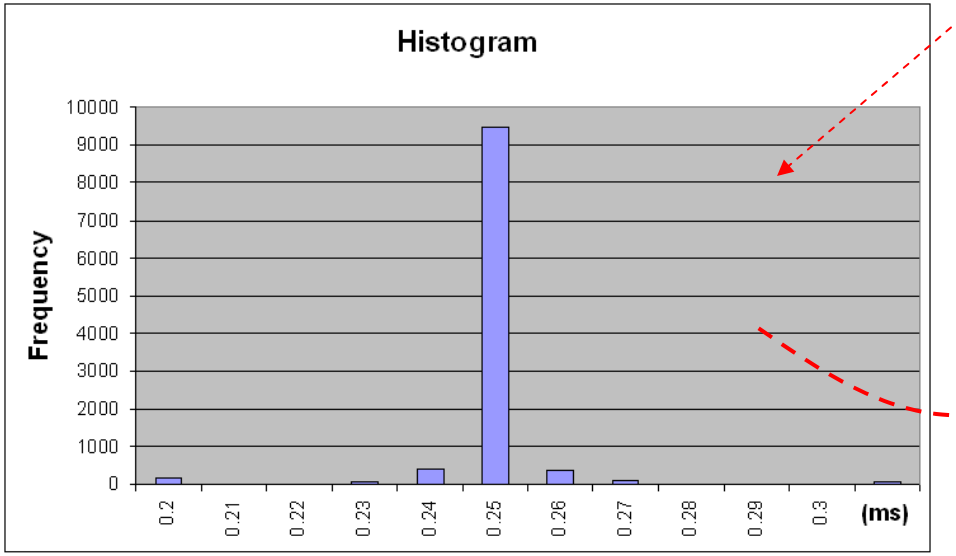


# Resource-reservation switch

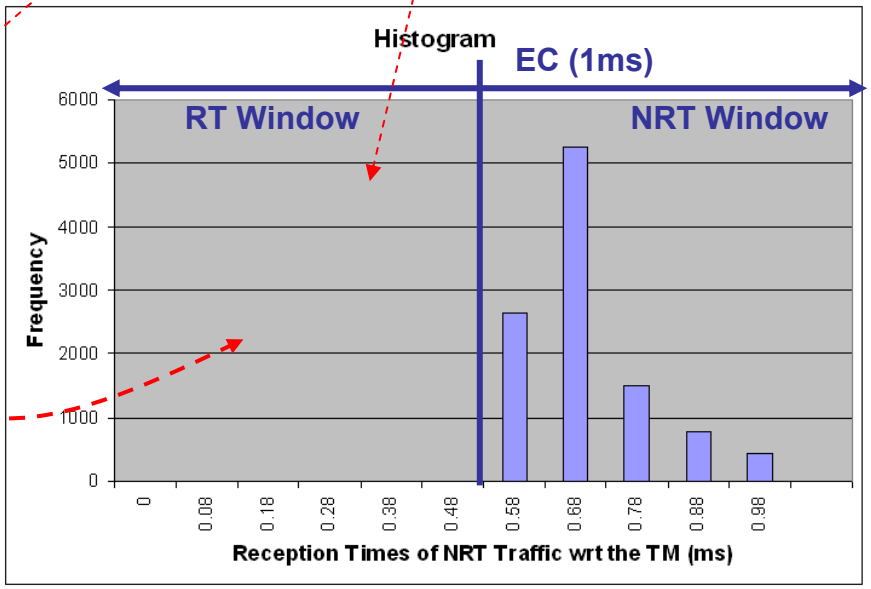
- **Aperiodic traffic confinement**




**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<br>$T\_TM_{max}$ = 1,0003ms<br>$T\_TM_{min}$ = 0,99998ms<br>STD_TM = 138ns |
|---|---|

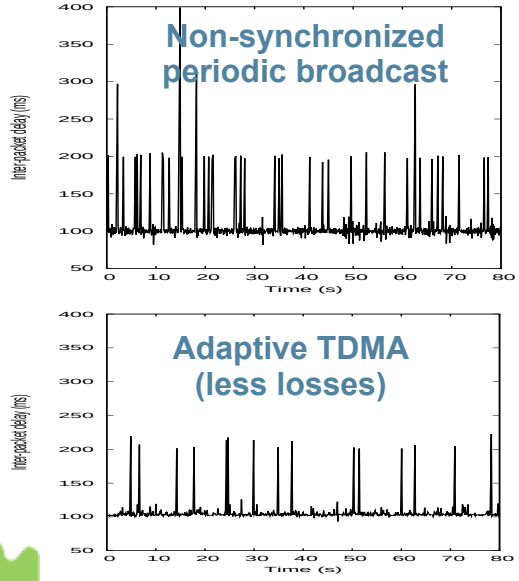
# Robust IEEE802.11 communication

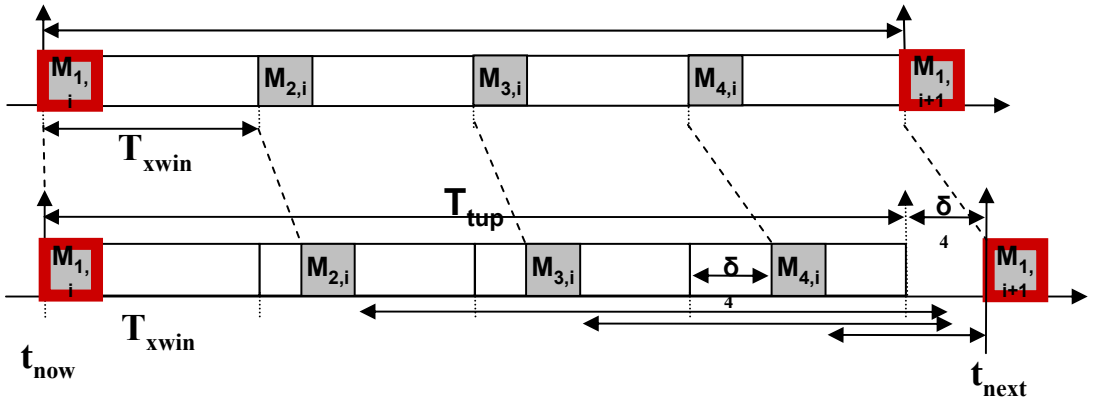
- Providing robust IEEE 802.11 communication for teams of autonomous mobile robots
  - **Real-time is *best-effort***
    - Open medium, uncontrolled environment / load, non-stationary interference...
  - **Adaptive techniques** can **help reducing** chances of **packet losses**

# Robust IEEE802.11 communication

## Adaptive TDMA

- Maximizes separation between transmissions in the team
- Synchronizes on receptions (no need for clock sync)
- Shifts phase of TDMA round to match periodic interference
- Time constraints → round period $T_{tup}$
- Fully distributed

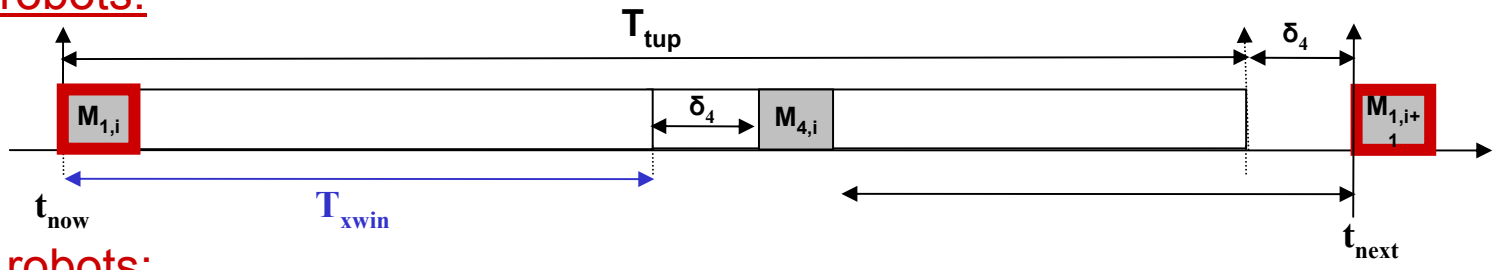


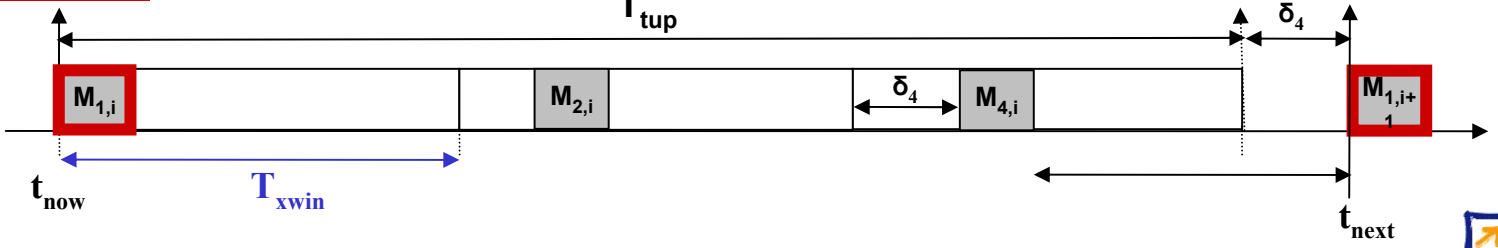
# Robust IEEE802.11 communication

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

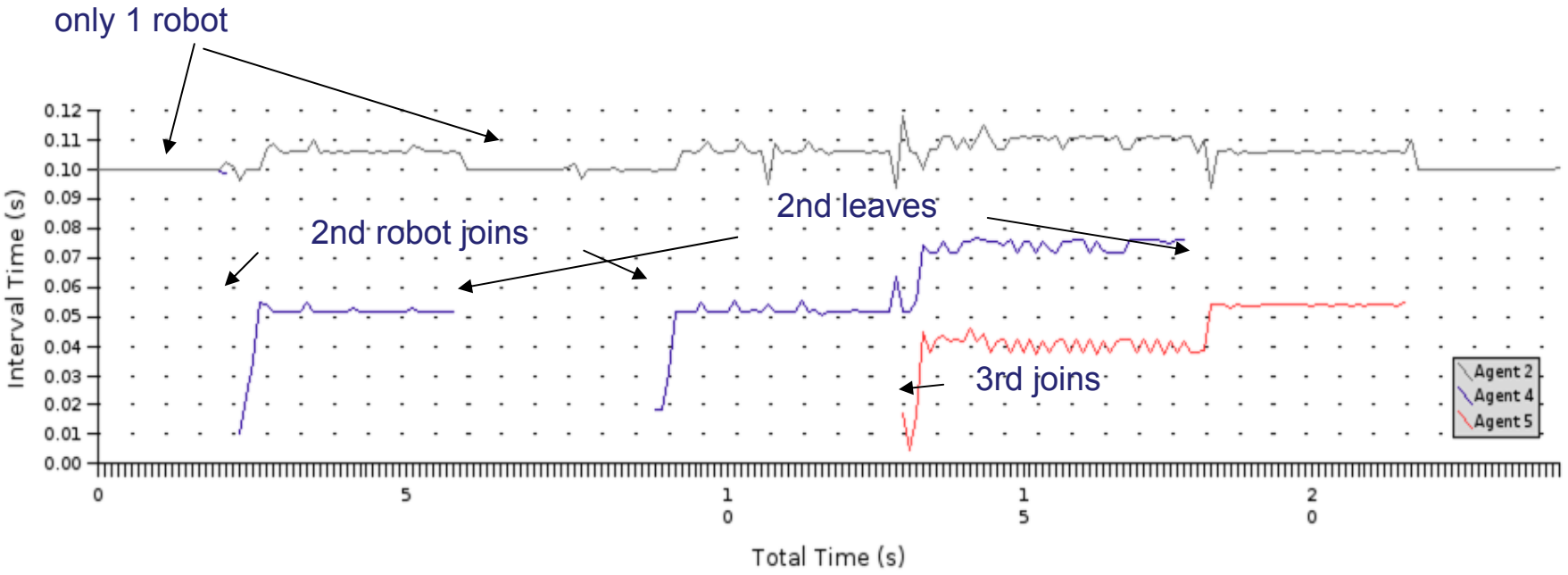
2 running robots:




3 running robots:



# Robust IEEE802.11 communication



# Conclusion

- **Adapting / reconfiguring** a distributed system on-line requires appropriate support from the **network**
- Two **main approaches** can be followed:
- Law enforcement
  - Such as the **FTT framework,** aiming at **controlled adaptation/reconfig.** (timeliness guarantees, traffic isolation…)
- Voluntary cooperation
  - **Wireless communication** for teams of autonomous agents, using **adaptive techniques** can help **reducing packet losses** in the presence of uncontrolled load (best effort way)
- ***A few general issues***
  - *How much control have we got over system resources?*
  - *Which guarantees do we need?*
  - *How much does flexibility cost? How much do we gain?*

# Announcement

*A students design competition*

- **CiberMouse@RTSS2008**
  - **http://www.ieeta.pt/lse/ciberRTSS08**
- *Control the team of 5 robots with **ad-hoc communication capabilities** to reach the victim in the least time*